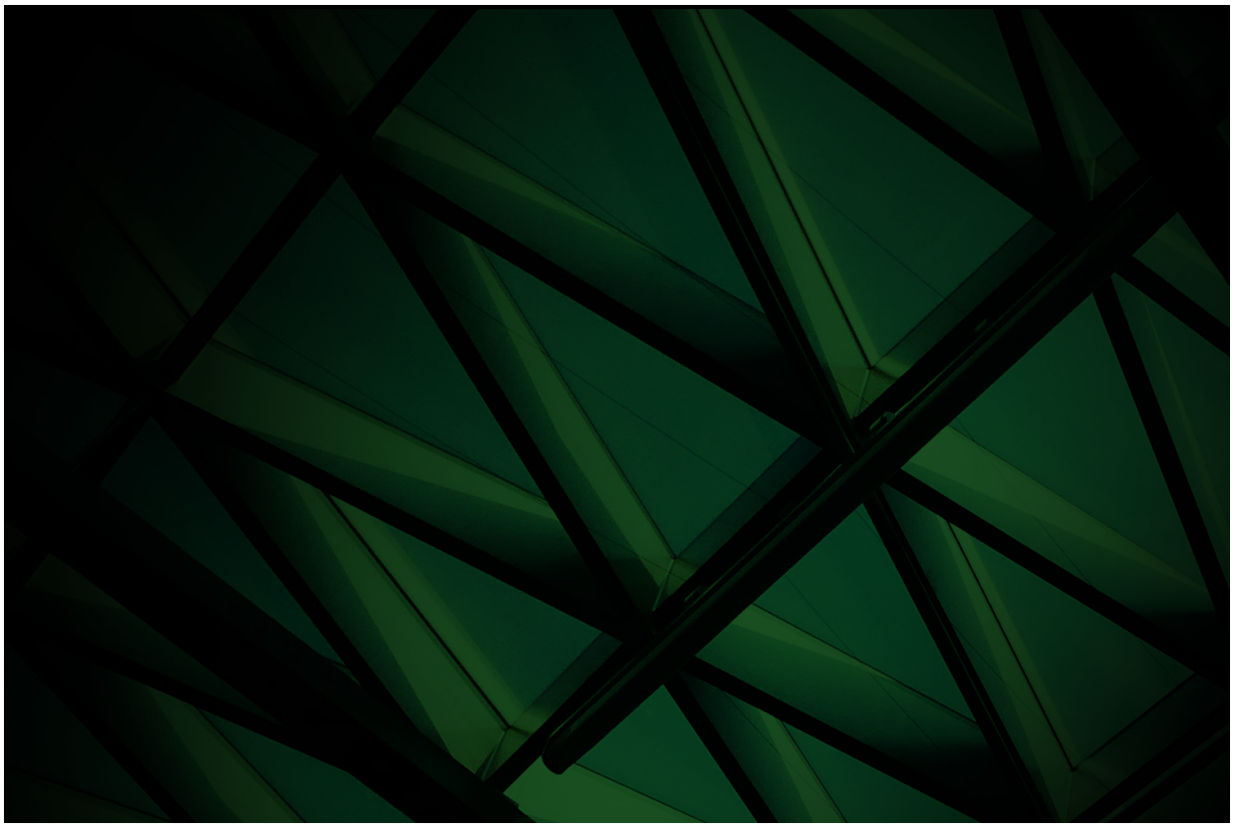


Installation and Configuration Guide

VERSION 2022.0.05

jade platform

Contents

Before You Begin

The *Installation and Configuration Guide* is intended as the main source of information when you are installing and configuring Jade.

Who Should Read this Guide

The main audience for the *Installation and Configuration Guide* is expected to be system administrators.

What's Included in this Guide

The *Installation and Configuration Guide* has three chapters and one appendix.

Chapter 1	Covers installing Jade
Chapter 2	Covers configuring Jade
Chapter 3	Covers Jade containers
Appendix A	Covers the Jade program and warning exit values

Conventions

The *Installation and Configuration Guide* uses consistent typographic conventions throughout.

Convention	Description
Arrow bullet (❯❯)	Step-by-step procedures. You can complete procedural instructions by using either the mouse or the keyboard.
Bold	Items that must be typed exactly as shown. For example, if instructed to type **foreach**, type all the bold characters exactly as they are printed. File, class, primitive type, method, and property names, menu commands, and dialog controls are also shown in bold type, as well as literal values stored, tested for, and sent by Jade instructions.
Italic	Parameter values or placeholders for information that must be provided; for example, if instructed to enter *class-name*, type the actual name of the class instead of the word or words shown in italic type. Italic type also signals a new term. An explanation accompanies the italicized type. Document titles and status and error messages are also shown in italic type.
Blue text	Enables you to click anywhere on the cross-reference text (the cursor symbol changes from an open hand to a hand with the index finger extended) to take you straight to that topic. For example, click on the "Work File Directory Location" cross-reference to display that topic.
Bracket symbols ([ ])	Indicate optional items.
Vertical bar (|)	Separates alternative items.
`Monospaced font`	Syntax, code examples, and error and status message text.

Convention	Description
ALL CAPITALS	Directory names, commands, and acronyms.
Small font	Keyboard shortcut keys.

Key combinations and key sequences appear as follows.

Convention	Description
Key1+Key2	Press and hold down the first key and then press the second key. For example, "press Shift+F2" means to press and hold down the Shift key and press the F2 key. Then release both keys.
Key1,Key2	Press and release the first key, then press and release the second key. For example, "press Alt+F,X" means to hold down the Alt key, press the F key, and then release both keys before pressing and releasing the X key.

Related Documentation

Other documents that are referred to in this guide, or that may be helpful, are listed in the following table, with an indication of the Jade operation or tasks to which they relate.

Title	Related to…
Database Administration Guide	Administering Jade databases
Development Environment Administration Guide	Administering Jade development environments
Development Environment User's Guide	Using the Jade Platform development environment
Developer's Reference	Developing or maintaining Jade applications
Encyclopaedia of Classes	System classes (Volumes 1 and 2), Window classes (Volume 3)
Encyclopaedia of Primitive Types	Primitive types and global constants
External interface Developer's Reference	Developing Jade applications using external interfaces
Jade Initialization File Reference	Maintaining Jade initialization file parameter values
Synchronized Database Service (SDS) Administration Guide	Administering Jade Synchronized Database Services (SDS), including Relational Population Services (RPS)
Object Manager Guide	Jade Object Manager administration
Jade Platform Differences Guide	Platform differences when running Jade applications
Report Writer User's Guide	Using the Jade Report Writer to develop and run reports
Runtime Application Guide	Administering deployed Jade runtime applications
Thin Client Guide	Administering Jade thin client environments
Web Application Guide	Implementing, monitoring, and configuring web applications

Chapter 1 Installing Jade under Windows

This chapter covers the following topics.

Operational Requirements

This section describes the Jade operational requirements and contains the following subsections.

Hardware Requirements

This section describes the hardware required for Jade development and Jade applications running under a Windows operating system.

Further requirements that must be met to satisfactorily provide for data recoverability are described in the *Environmental Considerations for Deploying Jade* white paper on the Jade website at https://www.jadeplatform.com/developer-centre/learn/whitepapers.

Notes These specifications represent minimum configurations. Configurations that are more powerful may be advisable, depending on the scale of your Jade applications and the performance requirements of your system.

Jade handles only the first nine monitors running on one workstation. Additional monitors are ignored.

Minimum Hardware Requirements for a Jade Database Server

The following subsections contain the recommended minimum hardware required to deploy Jade on a machine that hosts a Jade database server node.

Memory

For database servers, although Error Checking and Correcting (ECC) RAM is not a requirement, we recommend at least 2G bytes of ECC RAM.

Add 128M bytes for each Jade node.

Storage

For database servers, you require:

- Disk subsystem that guarantees that acknowledged writes are non-volatile (for details, see the *Environmental Considerations for Deploying Jade* white paper on the Jade website at https://www.jadeplatform.com/developer-centre/learn/whitepapers)
- Available disk space of four times the expected database size, allowing for growth
- RAID 1+0 configuration (for details, see also the *Environmental Considerations for Deploying Jade* white paper on the Jade website)

Other

For database servers:

- Server class hardware is essential
- Data backup components (for example, a backup disk drive, tape drive, or an optical read-write drive)

Minimum Hardware Requirements for an Application Server

The following subsections contain the recommended minimum hardware required to deploy Jade on machines hosting an application server.

Memory

For application servers, although Error Checking and Correcting (ECC) RAM is not a requirement, we recommend at least 2G bytes of ECC RAM.

Add 128M bytes for each Jade node.

Storage

When computing disk space requirements for application servers, allow for software installation size, transient object storage, and application external file requirements.

Other

For application servers:

- Server class hardware is recommended
- Data backup components (for example, a backup disk drive, tape drive, or an optical read-write drive) if not provided by another server (for example, by the database server)

Minimum Hardware Requirements for Standard Clients

The following subsections contain the recommended minimum hardware required to deploy Jade on a machine that hosts a Jade standard client.

Memory

For standard clients, you require 512M bytes Error Checking and Correcting (ECC) RAM.

Add 128M bytes for each Jade node.

Storage

When computing disk space requirements for standard clients, allow for software installation size, transient object storage, and application external file requirements.

Other

For standard clients:

- Screen resolution of 800x600 or higher is required

Minimum Hardware Requirements for Running Presentation Clients

The following subsections contain the recommended minimum hardware required to deploy Jade on a presentation (or *thin*) client node.

Memory

For presentation clients, you require 512M bytes RAM (recommended) or 256M bytes (minimum, depending on the operating system).

Storage

When computing disk space requirements for presentation clients, allow for software installation size, presentation client forms cache, and application external file requirements.

Other

For presentation clients:

- Screen resolution of 800x600 or higher is required

Software Requirements

This section describes the software required for Jade development and Jade applications.

Notes The database server will run only on a 64-bit version of Windows. Clients can be 32-bit or 64-bit.

Jade encodes Unicode characters using the wide-character-encoding routines provided by Microsoft Windows.

Jade 2022 requires the Microsoft Visual C++ 2015-2022 Redistributables to be installed. Download the latest version from Microsoft, or use the version that comes with Jade. You must install the x64 redistributables to run a 64-bit node, and the x86 redistributables to run a 32-bit node.

The minimum required Microsoft Visual C++ version is distributed with Jade, in the **\bin** folder, or you can download the latest available from Microsoft. The Jade installer will automatically install the redistributables when run, and a thin client download will also download and install the redistributables. You will need to install the updated redistributables on computers that host application or database servers yourself. Installation of the required C++ runtimes will be done as part of the normal Jade installation, if necessary. Installing this Microsoft redistributable package requires administration privileges.

Microsoft Windows Operating System

The supported Windows operating system versions are:

- Windows Server 2022 (from Jade 16.0.02.058, 18.0.01.109, 20.0.01.057, 20.0.02, and later)
- Windows Server 2019 (from Jade 2016.0.02, Jade 2018.0.01, and later)
- Windows Server 2016 (from Jade 7.1.09, Jade 2016.0.01, and later)
- Windows Server 2012
- Windows 11 (from Jade 16.0.02.058, 18.0.01.109, 20.0.01.057, 20.0.02, and later)
- Windows 10 (from Jade 2016.0.01, and later)

See also the notes in "Software Requirements", in the previous section.

Additional Software Requirements for Deploying Jade Web Applications

The additional software required to deploy Jade web applications is the minimum of one of the following.

- Microsoft Internet Information Server (IIS)
- WebSocket protocol requires IIS 8 or later
- Apache HTTP Server 2.4.10 for Microsoft Windows

In addition, Jade web services support SOAP version 1.1 and version 1.2.

ODBC Requirements for External Database Coexistence or Jade ODBC Driver Usage

The Microsoft Data Access Components (MDAC) installed with the Windows operating system are sufficient for running an external relational database or using the Jade ODBC standard or thin client driver.

Note The Jade ODBC drivers are available in 32-bit and 64-bit versions. If running on a 64-bit machine, the driver used must match the third-party tool being used; for example, it may be necessary to install 32-bit Jade ODBC drivers for use with 32-bit tools.

When using an external database on a 64-bit machine, the bit version of the Jade node executing the external database access must match the bit version of the external database ODBC driver that is being used.

If you are running Jade on a 64-bit machine under Windows in 64-bit mode and you are configuring a 32-bit ODBC driver, run the following program:

```
<\windows-directory>\SysWOW64\odbcad32.exe
```

This runs the 32-bit version of the Microsoft Data Source Administrator program rather than the 64-bit version.

For more details, see "Configuring a Jade ODBC Driver", in Chapter 2 of the *External Interface Developer's Reference*.

Jade Generic Messaging Requirements

The IBM WebSphere MQ interface in the Jade Generic Messaging module is built using the WMQ version 7.0.1.3 Client interface library.

The WMQ API can be used in both 64-bit and 32-bit client nodes.

.NET Requirements

To develop and compile .NET Framework applications for:

- .NET exposure, you require a *minimum* of Microsoft Visual Studio 2017 or higher.

  Note Projects must target .NET Framework 4.8.

- .NET (formally called .NET Core), you require a *minimum* of Microsoft Visual Studio 2022 or higher.

  Note Projects must target at least .NET 6.

For external component libraries, you require:

- .NET 4.x components (or .NET 3.x, if you are using components designed for .NET 3.x)

Postscript Printing Requirement

If using Postscript printing, your printer must support Postscript level 2 or greater.

RPS Node Requirements

With the Relational Population Service (RPS), the following 64-bit SQL Server ODBC drivers must be used for the SQL Server versions.

- Azure SQL Database (consult the Microsoft Azure documentation for the versions and features that are supported).

  Note For details about Jade initialization file parameters to configure a secondary RPS database for Azure, see "Configuring a Secondary RPS Database for Azure", in Chapter 2 of the *Synchronized Database Service (SDS) Administration Guide*.

- Azure SQL Managed Instance (consult the Microsoft Azure documentation for the versions and features that are supported).
- Microsoft SQL Server 2022 requires ODBC Driver 18 for SQL Server.
- Microsoft SQL Server 2019 requires ODBC Driver 17 for SQL Server or later (ODBC Driver 18 for SQL Server is recommended).
- Microsoft SQL Server 2016 requires ODBC Driver 13 for SQL Server or later (ODBC Driver 18 for SQL Server is recommended).
- Microsoft SQL Server 2014 requires SQL Server Native Client 11.0 or later.
- Microsoft SQL Server 2012 requires SQL Server Native Client 11.0 or later.
- Microsoft SQL Server 2008 requires SQL Server Native Client 11.0 or later.

The **sqlcmd** SQL Server utility must be installed on the RPS node to use the default (recommended) **sqlcmd** to execute SQL scripts.

Virtual Environments

Jade can be run in a virtual environment on all supported operating systems listed in the previous sections on Intel-compatible hardware.

Support is provided as per the published release policy for faults found in a virtual environment; however, a problem that is suspected to be related to an issue in the virtualization layer may result in requests for additional customer diagnostic actions. In some rare cases, fixes for issues that are specific to the virtualization layer may need to be negotiated with the customer.

Installing Your Jade Software

The Jade software is supplied from the Jade website.

Notes If you want to develop your own installation process for Windows, the Jade install and upgrade steps are documented in the **ReadmeInstallSteps.pdf** file in the documentation directory.

Installing any Microsoft redistributable package requires administration privileges.

When the installation is complete, the Jade set-up program informs you that the Jade set-up program was successfully completed and the **ReadMe.txt** file can now be viewed.

Note Example files are not part of the installation and must be downloaded and installed from the Jade website separately, if required.

Initiating the Installation from the Jade Setup Program

To initiate the installation of your Jade Platform software

1. Ensure that you have the appropriate privileges to install applications.
2. Double-click the executable program downloaded from the Jade website (https://www.jadeplatform.com/developer-centre/downloads).

The installation of your Jade software is then started and the initialization progress is displayed in the Setup progress dialog.

Jade Software Installation Process

This section describes actions that are performed by the Jade set-up program during the installation process.

Selecting the Type of Installation

The Installation Type folder is then displayed, to enable you to select the type of installation that you require. The types of installation that you can select are:

- **Fresh Copy**, which is selected by default, and installs a brand new copy of Jade.
- **Feature Upgrade**, which you can select if you want to upgrade an existing Jade database or binary files, or both database and binary files. For details, see the **ReadmeInstallSteps.pdf** document in the documentation directory.

Selecting the Type of Set Up

The Setup Type folder is then displayed, to enable you to select the type of set up that you require.

The types of set up that you can select are listed in the following table.

Type	Installs…
Development	Binary, database, Jade Report Writer, and general support files required to run single user and multiuser mode of the development and application runtime environments and Jade application server and presentation client files (64-bit only)
Application Runtime	Only the binary and general support files required to run the Jade application server
Presentation Client	Only the binary and general support files required to run the presentation client in Jade thin client mode
Jade Client	Only the binary files and general support files required to run a Jade client
Custom	User-selected Jade components (for the **Fresh Copy** installation type only)
SDS/RPS Database Server	Binary and database files for the Jade environment on SDS (**Feature Upgrade** installation type and 64-bit only)

By default, the **Development** option is selected for 64-bit, or **Jade Client** for 32-bit.

Selecting the Components to Install

When you specify a **Custom** installation, the Select Components folder is then displayed, to enable you to select the file components that you want to install. You can select the installation of the following components.

- Binary Files
- Include Files
- Library Files
- Presentation Client Files (64-bit, 32-bit)
- ReadMe File
- Report Writer Files (Jade Report Writer)
- System Files (Jade development database), 64-bit only
- WSDL Files

No components are selected for installation, by default.

The required disk space for all currently selected components is displayed at the bottom of the folder.

Specifying Your User Information

If you are installing a Jade database, the User Information folder enables you to specify the **Licence Name** (that is, your organization name, which is not to be confused with your company name that is displayed in large letters above the licence name and key) and your **Licence Key** specified on your licence for your Jade database.

An entry is required in both the **Licence Name** and **Licence Key** text boxes.

Notes To obtain your licence key, contact your local Jade authorized reseller.

The licence key has four text boxes. Enter the licence key *exactly* as it is specified on your Certificate of Authorisation (which may be an email message providing you with your licence name and key). Enter the first eight characters in the first text box, the next eight characters in the second text box, the next eight characters in the third text box, and the last eight characters in the fourth text box. (Ignore the "**-**" printed on the licence.)

Ensure that you type the licence name of your organization (which is case-sensitive) correctly, as this is validated against your licence key.

Jade Configurations

The Jade database can be run in one of the following configurations.

- Multiuser (the default)

  The multiuser configuration allows access to the database from multiple processes across multiple workstations. One workstation must perform the role of server, or remote, node. For more details, see "Multiuser Configuration", in the following subsection.

- Single user

  The single user configuration allows access to the database from one process only (for example, only one copy of **jade.exe**). For more details, see "Single User Configuration", later in this chapter.

For details about accessing Jade applications from IIS or the Apache HTTP Server, see "Connecting to Jade Applications from Internet Information Server (IIS)" and "Configuring JadeHttp for Remote Connections" or see "Connecting to Jade Applications from an Apache HTTP Server" and "Configuring Apache for Remote Connections", in Chapter 2.

Multiuser Configuration

Before running Jade in multiuser mode (the default), ensure that TCP/IP has been loaded on both your client workstations and the server (remote node) workstation. For details, see "Selecting Network Addresses", in Chapter 2, or "Using the Jade Remote Access Program", in Chapter 1 of the *Remote Access Program User's Guide*.

In the standard initial installation, the following parameters are set in the **jade**.**ini** file in the database directory.

```
[Jade]
server=multiUser                    // Default

[JadeServer]
NodeName=LocalHost
NetworkSpecification1=TcpIP,Enabled,6005

[JadeClient]
ServerNodeSpecifications=TcpIP,LocalHost,6005
```

This sets up the server and clients to run on the same installation node. (TcpIP is case-insensitive in Jade initialization file parameter values.)

For more details about the parameters in the Jade initialization file required to run Jade in a multiuser environment, see Chapter 1 of your *Jade Initialization File Reference*, particularly the "Jade Object Manager Client Section [JadeClient]" and "Jade Object Manager Server Section [JadeServer]" sections.

For details about using the server Uniform Resource Identifier (URI) string so that a client node can specify the target database server and transport details, see "Format of the Server URI String", in Chapter 2.

Running a Jade Server in Multiuser Mode

The default multiuser installation sets up the **Jade MultiUser \Jade Database Server** shortcut for the Jade server.

The following table lists examples of the properties required to run a Jade server.

Property	Example
Command line	jadrap.exe path=s:\jade\system ini=s:\jade\system\jade.ini app=Jade
Working directory	s:\jade\bin

Running a Jade Client in Multiuser Mode

The default multiuser installation sets up the **Jade MultiUser \Jade Client** shortcut for the Jade multiuser client.

Tip When running Jade in multiuser mode as a standard (fat) client, the Jade initialization file is usually specified on the command line. If a Jade initialization file is not specified, Jade attempts to use the **jade.ini** file in the server node **system** directory.

If this directory is not visible to the client node, the default values used may not be sufficient for your application to run.

The following table lists examples of the properties required to run the Jade Platform development environment in a Jade client.

Property	Example
Command line	jade.exe path=s:\jade\system ini=c:\jade\system\jade.ini app=Jade
Working directory	s:\jade\bin

Ensure that you specify the same path in the **path** argument of both the Jade Remote Access Program (**jadrap**) and Jade (**jade**.**exe**).

Running a Jade User Application in Multiuser Mode

The following table lists examples of the properties required to run a user application written in Jade on a client node.

Property	Example
Command line	jade.exe path=s:\jade\system schema=AccountsApp app=Accounts ini=c:\jade\system\jade.ini
Working directory	s:\jade\bin

Ensure that you specify the same path in the **path** argument of both the Jade Remote Access Program (**jadrap**) and Jade (**jade.exe**).

Running an Application Server in Jade Thin Client Mode

The default multiuser installation sets up the **Jade MultiUser \ Jade Application Server** shortcut for the Jade application server.

The following table lists examples of the properties required to run an application server in Jade thin client mode.

Property	Example
Command line	jadapp.exe appServerPort=1500 server=multiUser path=s:\jade\system ini=s:\jade\system\jade.ini
Working directory	s:\jade\bin

For more details, see "Invoking an Application Server", in Chapter 2 of the *Thin Client Guide*.

Running a Presentation Client in Jade Thin Client Mode

The default multiuser installation sets up the **Jade MultiUser \Jade Presentation Client** shortcut for the Jade presentation client.

The following table lists examples of the properties required to run a presentation client in Jade thin client mode.

Property	Example
Command line	D:\Jade\bin\jade.exe app=SortApp schema=SortTest appServer=JadeServer appServerPort=1500
Working directory	D:\jade\bin

If a Jade initialization file is not specified on the command line when you are running an application in thin client mode, Jade attempts to use the **jade.ini** file in the installation directory of **jade.exe** (that is, the **bin** directory) on the presentation client.

For more details, see "Invoking a Jade Presentation Client", in Chapter 2 of the *Thin Client Guide*.

Single User Configuration

Before you run Jade in single user mode, ensure that the **Server** parameter in the [Jade] section of the Jade initialization file is set to **singleUser**. Alternatively, you can specify **server=SingleUser** in the command line.

Running Jade in Single User Mode

The default installation sets up the **Jade** shortcut for a single user mode client. The following table lists examples of the properties required to run the Jade Platform development environment in single user mode.

Property	Example
Command line	jade.exe path=c:\jade\system ini=c:\jade\system\jade.ini app=Jade server=singleUser
Working directory	c:\jade\bin

Running a Jade User Application in Single User Mode

The following table lists examples of the properties required to run a user application written in Jade in single user mode.

Property	Example
Command line	jade.exe path=c:\jade\system ini=c:\jade\system\jade.ini schema=TPC_A app=BenchmarkApp server=SingleUser
Working directory	c:\jade\bin

Installing Multiple Jade Initialization Files

If you want multiple users to share binary files on your Local Area Network (LAN), install multiple copies of the Jade initialization file. For example, you can install multiple initialization files so that each user has his or her own initialization file.

» To install multiple Jade initialization files

1. Copy the Jade initialization file (**jade.ini**) into the appropriate user directory.
2. Create a Jade shortcut to meet your requirements, with the **ini** argument in the command line specifying the Jade initialization file of the user; for example:

```
c:\jade\bin\jade.exe ini=c:\current\myjade.ini path=c:\jade\system
schema=Faults server=singleUser app=Jade
```

The *working-directory-name* is usually the directory in which the Jade executable file (**jade.exe**) is located but you can specify another start-in directory when you install Jade, if required. The arguments can be in any order but must be specified once only.

For details about multiple Jade programs on the same host sharing a Jade initialization file, see "Sharing Jade Initialization Files", in the *Jade Initialization File Reference*.

Reregistering Jade with a New Licence

The **JadReg** or **jadregb** program installed with your Jade software enables you to reregister Jade with new licence information.

Note Jade licenses are not transferred automatically between databases in an SDE. It is your responsibility to apply new licenses to any existing databases in an SDE. In addition, to ensure proper operation, you must apply the primary license to every secondary.

For details about automating the registration of a Jade system by running the registration program in batch mode, see "Reregistering a Jade System in Batch Mode", in Chapter 2.

Chapter 2 Configuring Jade

This chapter covers the following topics.

Directory Locations

The ability to return locations other than the Jade HOME directory for program and user data requests is dependent on where the Jade binaries have been installed. If the Jade programs are installed in the **\Program Files** system-wide location for installed applications, alternative directory locations are implemented.

Because security restrictions can deny appropriate access to directories or the directories are a unique location for each user, the **Node** and **Process** classes provide methods that enable you to access other directory locations in the file system hierarchy. In addition, the [JadeEnvironment] section of the Jade initialization file provides the **JadeWorkDirectory**, **ProgramDataDirectory**, and **UserDataDirectory** parameters, which enable you to specify the location of Jade work files, program data, and user data, respectively.

In the methods summarized in the following subsections, a method on a specific:

- Node instance performs its action on the specified node instance, which does not have to be the current node. If the current node is required, use the **node** environmental object (system variable).
- Process instance performs its action on the owning node (that is, a **Process.node** instance) if the process is not associated with a presentation client. If the process has an associated presentation client, the action is performed on the presentation client. The presentation client does not have to be the current presentation client or a presentation client attached to the same application server.

For details, see Volume 1 and Volume 2 of the *Encyclopaedia of Classes* and "Jade Environment [JadeEnvironment] Section", in the *Jade Initialization File Reference*.

Installation Directory Location

The **getJadeInstallDirectory** method defined in the **Node** and **Process** classes returns a string containing the directory in which the executable of the current executing program is located; that is, the directory in which the Jade binaries are installed.

See also "Work File Directory Location", later in this chapter.

Home Directory Location

The **getJadeHomeDirectory** method defined in the **Node** and **Process** classes returns a string containing the parent directory of the installation directory.

Program Data Directory Location

The **getProgramDataDirectory** method defined in the **Node** and **Process** classes returns a string containing the program data directory, which is dependent on the:

- Value of the **ProgramDataDirectory** parameter in the [JadeEnvironment] section of the Jade initialization file. The parameter is read each time the **getProgramDataDirectory** method is called. When you call this method and the directory does not already exist, Jade creates it based on the value of the **ProgramDataDirectory** parameter.
- Location of the installation directory.

Files that should be placed under the returned location are entities that should be shared across multiple users of these binaries; for example, the **jommsg.log** file or shared dictionary spelling files that are updated.

If the installation directory is a subdirectory of the programmatically determined location of **\Program Files**, translation rules are applied. However, if the installation directory is not a subdirectory of the **\Program Files** location, the return value is the same as the Jade HOME directory.

The **ProgramDataDirectory** parameter in the [JadeEnvironment] section of the Jade initialization file can have one the following values.

- **<default>**, which is the default value

  The **<default>** value has the same meaning as the **<programdata>** value.
- **<homedir>**

  The value of the Jade HOME directory is returned.
- **<programdata>**

  The **\Program Files** portion of the HOME directory is replaced with the programmatically obtained location for the common application data directory. For example, a presentation client installed into **\Program Files\Jade Software Corporation\parsys** returns **\ProgramData\Jade Software Corporation\parsys**.
- A user-specified directory name, which returns the specified directory

The directory location is created, if necessary, and returned in the required format.

User Data Directory Location

The **getUserDataDirectory** method defined in the **Node** and **Process** classes returns a string containing the user data directory, which is dependent on the:

- Value of the **UserDataDirectory** parameter in the [JadeEnvironment] section of the Jade initialization file. The parameter is read each time the **getUserDataDirectory** method is called. When you call this method and the directory does not already exist, Jade creates it based on the value of the **UserDataDirectory** parameter.
- Location of the installation directory.

Files that should be placed under the returned location are entities that should be unique to each user of these binaries; for example, if a presentation client installation occurs on a Windows machine running Citrix or Terminal Services and all users run the same thin client binaries, any data created on the client file system should be stored under this location (that is, unique dictionaries for each user).

If the installation directory is a subdirectory of the programmatically determined location of **\Program Files**, translation rules are applied. However, if the installation directory is not a subdirectory of the **\Program Files** location, the return value is the same as the Jade HOME directory.

The **UserDataDirectory** parameter in the [JadeEnvironment] section of the Jade initialization file can have one the following values.

- **<default>**, which is the default value

  The **<default>** value has the same meaning as the **<userdata>** value.

- **<homedir>**

  The value of the Jade HOME directory is returned.

- **<userdata>**

  The **\Program Files** portion of the HOME directory is replaced with the programmatically determined location for the specific user application private data directory. For example, a presentation client installed into **\Program Files\Jade Software Corporation\parsys** and executed by user **wilbur** returns **\Users\wilbur\AppData\Local\Jade Software Corporation\parsys**.

- A user-specified directory name, which returns the specified directory

The directory location is created, if necessary, and returned in the required format.

Work File Directory Location

The **getJadeWorkDirectory** method defined in the **Node** and **Process** classes returns a string containing the Jade work directory, which is dependent on the:

- Value of the **JadeWorkDirectory** parameter in the [JadeEnvironment] section of the Jade initialization file. The parameter is read each time the **getJadeWorkDirectory** method is called and the directory is accessed. When you call this method and the directory does not already exist, Jade creates it based on the value of the **JadeWorkDirectory** parameter.
- Location of the Jade HOME directory.

By default, the **jade.exe** executable program file does not directly create or update files in the binary directory.

- The **JadeWorkDirectory** parameter in the [JadeEnvironment] section of the Jade initialization file determines the directory in which work files are created.

  By default, this directory is created at the same level as the Jade binary directory and is named **temp**. For example, the directory is named **c:\jade\temp** if the Jade installation directory is **c:\jade\bin**.

  The **JadeWorkDirectory** parameter can specify an absolute path or a relative path (relative to the Jade HOME directory, which is **c:\jade** in the above example).

- The presentation cache file is written into the directory defined by the value of the **JadeWorkDirectory** parameter unless the **FormCacheFile** parameter in the [JadeThinClient] section of the Jade initialization file specifies the location of the form cache file.
- The presentation client automatic download process lock files are created in the directory specified by the **JadeWorkDirectory** parameter.

- The automatic presentation client download **log** file is written to the location specified by the **LogDirectory** parameter in the [JadeLog] section of the Jade initialization file.

The only situations in which a file creation or update will occur within the Jade binary directory are as follows.

- If the Jade initialization file is positioned in the binary directory (the default action for a presentation client). You can avoid this by using the **ini** argument in the Jade presentation client command line to specify an alternative location on initiation of the presentation client.
- When the **jaddinst** executable program installs files downloaded by the Jade thin client automatic download facility.

Configuring Your Network Protocol

TCP/IP is required for multiuser configurations. TCP/IP must be installed on each workstation that is to participate as a node in the Jade environment. Your network manufacturer supplies TCP/IP and your network administrator supplies the TCP/IP address.

Jade also provides fast transport mechanisms for Jade modules on the same machine (for example, one that has multiple CPUs). For details, see "Hybrid Pipe Shared Memory (HPSM) Transport" and "JadeLocal Transport", later in this chapter.

Use the:

- **NetworkSpecification** parameter in the [JadeServer] section of the Jade initialization file to specify the transport types used by a Jade server.
- **ServerNodeSpecifications** parameter in the [JadeClient] section of the Jade initialization file to specify the transport type used by a Jade client to connect to the server.
- Server URI string in the **server** argument of a command line, to specify the target database and the client-server transport. For details, see "Format of the Server URI String", later in this chapter.

As Jade database servers and Jade application servers can often reside on the same machine, you can use a local intra-machine transport to significantly improve performance. You can also use these transport mechanisms to communicate from standard (fat) clients and Jade application servers to Jade servers if these processes run on the same physical machine, to significantly improve overall performance. Intra-machine local transport uses *shared memory*.

To detect whether a connection has been lost, a polling mechanism ensures that some minor communication is taking place regularly. See also "TransportIdlePollInterval" under "Jade Object Manager Server Section [JadeServer]", in the *Jade Initialization File Reference*.

If your database is running as a service but not with the **Local System** account, your Windows administrator must configure the user logon to add the following Windows privilege **Create global objects** (**SeCreateGlobalPrivilege** at the programming API level), which can be done directly to the user logon or to a group of which that user is a member. On a machine that is not part of a Windows domain, this can be done by accessing the **Local Security Policy** in the Administrative tools directory and adding the **Create global objects** policy under **Local Policies / User Rights Assignment** to the desired group or user. This allows Jade programs that need to connect via the **JadeLocal** or **HPSM** intra-machine transport to work across multiple Windows sessions or user logons.

With the **HPSM** transport, a database running with a standard account requires the extra privilege. Client nodes do not require the extra privilege.

With the **JadeLocal** and **HPSM** transports, if a standard user attempts to create a **Global\basename** value in either of these initialization file parameters, it fails because of insufficient privileges.

Selecting Network Addresses

Use the reserved range of Class C network addresses if you want to set up a small private network for demonstration purposes. (The reserved private Class C network addresses are in the range **192.168.0** through **192.168.255**, and are often used for broadcast messages.) If you therefore select network number **192.168.1**, for example, you can then assign all IP addresses for your host in the range **192.168.1.1** through **192.168.1.254**.

Ensure that the first three parts of this address are the same for all interfaces connected to the same LAN segment, and set the subnet mask to **255.255.255.0**.

Tip Choose a reserved private Class C IP address when one or more of the workstations also needs to connect to the Internet by using an Internet Service Provider (ISP). If you use these reserved addresses, they should not conflict with any valid Network Interface Controller (NIC)-assigned Internet addresses. Avoid using random numbers for IP addresses, as this usually fails.

When you have configured your network and IP addresses, check that workstations can see each other using the IP protocol. A simple test is to use **ping** from a command prompt, by entering one or both of the following command prompts from any workstation:

```
ping computer-name

ping remote-IP-address
```

If this does not provide a response, you will not be able to use the specified workstation name as the server node name in the **ServerNodeSpecifications** argument of the [JadeClient] section of your Jade initialization file.

If the IP is functioning correctly and the **ping *remote-IP-address*** command prompt provides a positive response, setting the **ServerNode** parameter to the server node IP address value in the **ServerNodeSpecifications** parameter in the [JadeClient] section of your Jade initialization file should work. Setting this **ServerNode** parameter to the workstation name of the server node may not always work even if the **ping *server-node-computer-name*** command prompt gives a valid response, as this alias to standard TCP/IP cannot always be relied on to work.

If you do not have Windows Internet Naming Services (WINS) or a Domain Name Server (DNS) server, you should set up a hosts file with IP address-to-name mappings and then use the names from there. If you have a small single-segment LAN and you do not know what Dynamic Host Configuration Protocol (DHCP)/DNS or WINS are, do not use them for your local networking. Only an experienced network administrator should set these up. (If you have Internet access, you will almost certainly have DNS configured.)

TCP/IP Transport

The TCP/IP transport must be used when the database server and client are on different computers. For connections on the same computer, the fast Hybrid Pipe Shared Memory (**HPSM**) transport is recommended, although TCP/IP or the **JadeLocal** transport can be used. See also "RPC, Database, and Node Configuration", later in this chapter.

The following Jade initialization file parameters are required to use the TCP/IP transport.

```
[JadeServer]
NetworkSpecification<n>=transport-type,enabled|disabled,
                        listener-port[,interface]

[JadeClient]
ServerNodeSpecifications=transport-type,remote-host,remote-port
                          [,local-interface[,local-port]]
```

Alternatively, use the **server** argument in a command line to specify the server Universal Resource Identifier (URI) target database and the client-server transport, instead of the **server** and **path** arguments in a command line and the **ServerNodeSpecifications** parameter in the [JadeClient] section of the Jade initialization file. For details, see "Format of the Server URI String", later in this chapter.

The values (which are case-insensitive) for the *transport-type* variable in the **NetworkSpecification** parameter are:

- **TcpIP**, which is a synonym for **TcpIPV4**.
- **TcpIPv4**, which provides IP version 4 connections only.
- **TcpIPv6**, which provides IP version 6 connections only.

If you want the server to support both network protocols, you must define a network specification for each protocol.

You can specify the *interface* value as a host name or an IP address. If you use an IP address, the address must be in an appropriate format for the specified *transport-type* value. Specify the local interface name or IP address if you want to select a specific network adapter in a database server node that has more than one network adapter installed; for example, to enable an administrator to ensure connections from clients connect on the fastest interface or to allow easier security when used in conjunction with a firewall or router access list. (Jade defaults to all network adapters in the node.)

The values for the *transport-type* variable in the **ServerNodeSpecifications** parameter are:

- **TcpIP**, which is a synonym for **TcpIPAny**.
- **TcpIPAny**, which connects using IP version 4 or 6.
- **TcpIPv4**, which connects using IP version 4.
- **TcpIPv6**, which connects using IP version 6.

You can specify the *remote-host* and *local-interface* values as a host name or an IP address. If you use an IP address, the address must be in an appropriate format for the specified *transport type* value. If you specify a host name, all DNS-provided addresses will be attempted.

For the **TcpIPAny** transport type, the client will first attempt to connect via IP version 6 and then IP version 4 protocol on the provided IP addresses. Each connection failure will be logged, and the next available combination tried.

Hybrid Pipe Shared Memory (HPSM) Transport

Hybrid Pipe Shared Memory (**HPSM**) provides a fast transport mechanism between Jade nodes on the same computer to utilize the multiple CPUs and *shared memory*, to significantly improve overall performance. (An older shared-memory transport, called **JadeLocal**, is documented in the following section.)

The Remote Procedure Call (RPC) **jomsrvr2.dll** includes a Hybrid Pipe Shared Memory (HPSM) transport, which has lower overhead than the **JadeLocal** (shared memory) transport and it is a better fit with the server thread model, as it results in fewer thread context switches per request.

Note The HPSM transport can be used only between nodes residing in the same machine, because it uses shared memory.

The following Jade initialization file parameters are required to use the HPSM transport.

```
[JadeServer]
NetworkSpecification1=HPSM,enabled|disabled,[Global\]base-name
```

```
[JadeClient]
ServerNodeSpecifications=HPSM,[Global\]base-name
```

The following is an example of the required Jade initialization file definitions to use the HPSM transport.

```
[JadeServer]
NetworkSpecification1=HPSM,Enabled,TestCRM-HPSM

[JadeClient]
ServerNodeSpecifications=HPSM,TestCRM-HPSM
```

The *base-name* value can have an optional **Local** or **Global** prefix.

When the **Local** prefix tag is specified (or no prefix tag is specified), the value of the *base-name* variable is created in the session namespace. To establish a connection, the Jade database server and the Jade client must be running under the same user logon and in the same Windows session. This would not be the case, for example, if you were running an interactive desktop application that attempted to connect to the Jade database server.

When the **Global** prefix tag is specified, the *base-name* variable is created as a global object that is visible across sessions and logons.

Note Use the **Global** prefix tag, in the **NetworkSpecification** and the **ServerNodeSpecifications** parameter to enable an HPSM connection to the Jade Database service across Windows sessions.

The **HPSM** transport has lower overhead than the **JadeLocal** transport, and is a better fit with the server thread model, as it has fewer thread context switches per request. For details about the server thread model, see "Server Thread Model", later in this chapter.

On a client node using HPSM, the **Node** class **networkAddress** method returns **"proc*NNNN*"**, where the ***NNNN*** value is the decimal number of the process at the other end of the connection.

Alternatively, use the **server** argument in a command line to specify the server Universal Resource Identifier (URI) target database and the client-server transport, instead of the **server** and **path** arguments in a command line and the **ServerNodeSpecifications** parameter in the [JadeClient] section of the Jade initialization file. For details, see "Format of the Server URI String", later in this chapter.

JadeLocal Transport

The **JadeLocal** transport between the Jade database and standard clients is implemented by the use of shared memory, global events, and semaphores.

Note The **JadeLocal** transport should be used when a node has one or two processes only.

The following Jade initialization file parameters are required to use the **JadeLocal** transport.

```
[JadeServer]
NetworkSpecification<n>=JadeLocal,enabled|disabled,[Global\]base-name

[JadeClient]
ServerNodeSpecifications=JadeLocal,[Global\]base-name
```

The *base-name* value can have an optional **Local** or **Global** prefix, as described for the HPSM transport, earlier in this chapter.

If the prefix is absent, it defaults to **Local**, which is consistent with running with the least-privileges mode. When running as a standard user, the value of the *base-name* variable is created in the **Local** or session namespace, which means that all Jade programs must be running as the same user logon and also in the same Windows session, to be able to connect to this RPC transport. For example, if the database is installed as a service, all application servers and standard clients wanting to connect to this database via **JadeLocal** transport must also be running as services and under the same user logon.

Note Local intra-machine transport is used only when the transport type is defined as **JadeLocal** or **HPSM** on both the server and client and the network state on the server is defined as **Enabled**.

Alternatively, use the **server** argument in a command line to specify the server Universal Resource Identifier (URI) target database and the client-server transport, instead of the **server** and **path** arguments in a command line and the **ServerNodeSpecifications** parameter in the [JadeClient] section of the Jade initialization file. For details, see "Format of the Server URI String", in the following section.

Format of the Server URI String

Use the **server** argument in a command line to specify the server Universal Resource Identifier (URI) target database and the client-server transport, instead of the **server** and **path** arguments in a command line and the **ServerNodeSpecifications** parameter in the [JadeClient] section of the Jade initialization file.

The syntax of the server URI command line argument is as follows.

```
server=scheme://target-address/environment-UUID[/server-UUID][?arguments]
```

The *scheme* and *target-address* values specify the client-server transport definition. The environment identity UUID (in display form) specifies the database to which the client node is connecting. This is optionally followed by the database server identity UUID. For details about the database identities, see "Database Identities", in Chapter 3 of the *Database Administration Guide*.

Specify both environment and server identities when you want the client node to connect to a specific SDS secondary database.

The following subsections explain how you can specify the target address for each URI scheme.

File Scheme (Single User)

The format of the single user **File** scheme is as follows.

```
File://database-directory
```

The *database-directory* value specifies the name of the directory where the database server finds the database control file (**_control.dat**). Use this format when running a single user client node; for example:

```
File://c:\jade\system
```

This example is equivalent to **server=SingleUser** and **path=c:\jade\system** in the command line.

TcpIP, TcpIPv4, and TcpIPv6 Schemes

The format of the target address in the **TcpIP**, **TcpIPv4**, and **TcpIPv6** schemes is as follows.

```
host:port-number
```

You can specify the host using a:

- Simple host name; for example, **george**, **server129**, or **localhost**.
- Fully qualified domain name (FQDN); for example, **wilbur.example.com**.
- Dotted decimal IPv4 address; for example, **127.0.0.1**.
- Quadded hexadecimal IPv6 address enclosed in square brackets; for example, **[fe80::bdf1:cbe:6902:7deb]**.

The **TcpIPv4** scheme uses the IPv4 protocol, so quadded hexadecimal IPv6 addresses are not allowed.

The **TcpIPv6** scheme uses the IPv6 protocol, so dotted decimal IPv4 addresses are not allowed.

The *port-number* value is a decimal integer in the range 1 through 65535.

When the host is a simple host name or an FQDN, resolution always returns **TcpIPv6** addresses before **TcpIPv4** addresses. When a specific protocol version is not specified, the first returned address is used.

The following examples show target addresses for the **TcpIP**, **TcpIPv4**, and **TcpIPv6** schemes.

```
server=TcpIP://localhost:6005/48cf13df-bf6d-df11-87e2-2e5925024153

server=TcpIPv4://host.company.com:6005/48cf13df-bf6d-df11-87e2-
2e5925024153?localHostname=aHostName

server=TcpIPv6://[fe81::6fe:7fff:fe87:bd20]:6005/48cf13df-bf6d-df11-87e2-
2e5925024153?localPort=54321

server=TcpIP://127.0.0.1:6005/48cf13df-bf6d-df11-87e2-2e5925024153/48cf13df-bf6d-
df11-87e2-2e5925024153
```

By default, the environment identity specified in the **server** command line argument of a client shortcut must match the identity of the database server, as shown in the following example.

```
server=TcpIp://localhost:6005/48cf13df-bf6d-df11-87e2-2e5925024153
```

If zeroes can be specified for the environment identity in the **server** command line argument of a client shortcut, as shown in the following example, set the value of the **AcceptZeroEnvironmentUUID** parameter in the [JadeServer] section of the Jade initialization file to **true**.

```
server=TcpIp://localhost:6005/00000000-0000-0000-0000-000000000000
```

Caution The **AcceptZeroEnvironmentUUID** parameter should not be set to **true** in production databases.

The following optional parameters enable you to specify the client node source address and port.

- LocalInterface=*host*

  The *host* value is a simple host name, a fully qualified domain name, or an appropriate IP address. You can use this to select a specific source address when the client machine has more than one network interface.

- LocalPort=*port-number*

  The *port-number* value is a decimal integer in the range **1** through **65535**. You can use this when the default random source port number cannot be used; for example, when the client and server machines are separated by a firewall, which limits the client source port numbers.

Hybrid Pipe Shared Memory (HPSM) Scheme

In the Hybrid Pipe Shared Memory (**HPSM**) scheme, shared memory is used to establish a connection so the client node must be on the same machine as the server.

The format of the target address is as follows.

```
localhost:base-name
```

The host must be **localhost**. The *base-name* contains up to 60 ASCII graphic characters excluding the forward slash (*/*), percent **(%**), and comma (**,**) characters. When the database server is installed as a Windows service, the *base-name* should be prefixed with **Global**.

The following example shows a target address for the **HPSM** scheme.

```
server=HPSM://localhost:SRCJade-HPSM/48cf13df-bf6d-df11-87e2-2e5925024153
```

JadeLocal Scheme (Shared Memory)

In the **JadeLocal** scheme, shared memory is used to establish a connection so the client node must be on the same machine as the server.

The format of the target address is as follows.

```
localhost:base-name
```

The host must be **localhost**. The *base-name* contains up to 60 ASCII graphic characters excluding the forward slash (*/*), percent (**%**), and comma (**,**) characters. When the database server is installed as a Windows service, the *base-name* should be prefixed with **Global** and the *base-name* value name in the range 1 through 60 characters.

The following example shows a target address for the **JadeLocal** scheme.

```
server=JadeLocal://localhost:SRCJade-SHM/48cf13df-bf6d-df11-87e2-2e5925024153
```

Server Thread Model

The server thread model uses two thread pools, as follows.

- The first pool (short threads) handles request types that have a short execution time (for example, a **getObject** call).
- The second pool (long threads) handles the request types that are expected to have a relatively long execution time.

Both thread pools use a Windows I/O completion port to distribute requests among its threads, thereby reducing the thread context switches in the server process.

The thread pools are managed with parameters in the [JadeServer] section of the Jade initialization file, as shown in the following example.

```
[JadeServer]
TransportIdlePollInterval=120000
MinShortThreads=10
MaxShortThreads=100
ConcurrentShortThreads=0
MinLongThreads=5
MaxLongThreads=50
ConcurrentLongThreads=0
```

All transports perform a keep-alive check, whose frequency is controlled by the value of the **TransportIdlePollInterval** parameter.

The number of threads in each pool grows dynamically, starting with the values of the **MinShortThreads** and **MinLongThreads** parameters, up to the values of the **MaxShortThreads** and **MaxLongThreads** parameters, depending on the request arrival rate.

The values of the **ConcurrentShortThreads** and **ConcurrentLongThreads** parameters control the maximum number of threads that the operating system can allow to concurrently process I/O completion packets for each I/O completion port. If the value of these parameters is the recommended value of zero (**0**), the system allows as many concurrently running threads as there are cores in the system. (For more details, see the Microsoft MSDN **CreateIoCompletionPort** function documentation.)

End-to-End SSL Encryption

Jade uses the Microsoft Secure Channel (Schannel) Security Package to provide SSL/TLS capability.

To use SSL, Jade integrates with the Windows Certificate Store to access the X.509 certificates used to establish a secure connection. Jade looks *only* in the local machine personal certificate store (the **CERT_SYSTEM_STORE_LOCAL_MACHINE\ MY** store, to be exact). This enables you to store certificates securely and utilize industry-standard tools to maintain them.

Note Jade will *not* take on the role of maintaining either the certificate store or any certificates.

Currently, **JadeSsl** (Jade's Schannel **SSL/TLS** integration) is offered only on the connections listed below. Every TCP/IP connection that Jade makes or allows you to make will eventually be secured by **JadeSsl**, with the connection of the thin client to the application server (currently secured by OpenSSL) being replaced last. When this occurs, OpenSSL will no longer be distributed with Jade.

Specialized Settings

JadeSsl lets the operating system make the decisions about the specifics of the security to offer; that is, which protocols, cipher suites, and so on. If you require fine-grained control of these settings, you can control them using the information provided by Microsoft in the following documentation: Transport Layer Security (TLS) registry settings (https://learn.microsoft.com/en-us/windows-server/security/tls/tls-registry-settings).

Configuration

Jade provides Secure Sockets Layer (SSL) encryption between:

- **jadehttp.dll** (IIS) and the application server (REST and SOAP web service providers)
- SDS primary and secondary nodes
- Connections with Microsoft Exchange servers using the Opportunistic Transport Layer Security (OTLS) from the **TcpIpConnection** class (applies to version 2022.0.01 and higher)
- Application server to database server (applies to version 2020.0.02 and higher)
- Standard client to database server (applies to version 2020.0.02 and higher)
- .NET client to database server (applies to version 2020.0.02 and higher)
- ODBC standard client to database server (applies to version 2020.0.02 and higher)

- ODBC thin client, whose configuration is controlled with the **OdbcThinClient** parameter in the [Connections] section of the Jade initialization file enables you to specify the name and path of the SSL configuration file (applies to version 2022.0.01 and higher)

The Jade ODBC Thin Client Setup dialog enables you to specify the name and path of the SSL configuration file.

» To use Jade SSL, put the following in your jade.ini or jadehttp.ini file

```
[JadeSsl]
ConfigurationStore=[path of the Jade SSL configuration file]
```

The Jade SSL configuration file is an initialization file-like language. It does not support two-level section names as the **jade.ini** file does. Because almost all Jade instances you control will be able to use the same configuration, the SSL configuration is designed to go in a separate file to which any Jade instance can point. This way, there is less configuration to manage for each Jade instance. The **ConfigurationStore** parameter can also point to the initialization file in which it is defined, although in this case the restriction on two-level section names still applies.

The Jade SSL configuration file has the following format.

```
[Connections]
Default=[configuration section n]
JadeHttpToClient=[configuration section n+1]
SDS=[configuration section n]
ClientToRap=[configuration section n]
OdbcThinClient=[configuration section n]
```

The [Connections] section contains the names of each type of connection that Jade will make. The special **Default** connection type is used for connections that are not specified in the list. Both sides of the connection will look at the same connection name, but do not have to look in the same physical file on the device.

The complete current list of internal SSL names is:

- ClientToRap
- SDS
- JadeHttpToClient
- OdbcThinClient
- Default

The [Connections] section contains the following parameter values that can be defined in a specific connection-type section.

```
[configuration section n]
UseEncryption=[true|false|opportunistically]
RequireClientAuthentication=[true|false]
ServerCertificateToSendName=[Any|certificate name]
ClientCertificateToSendName=[Any|None|certificate name]
AcceptableClientCertificateToReceiveNames=[Any|a certificate name,another
certificate name]
```

The name of the connection configuration section can be anything you like but must match one of the values in the [Connections] section for Jade to find it. Jade will then read that configuration for the type of connection it is creating. This provides you with a way to configure your system with the least amount of configuration to maintain.

If the value of the **UseEncryption** parameter is **false** (the default value), other parameter values in this section are not used.

If the value of the **RequireClientAuthentication** parameter is **true**, the server requires the client to send a certificate to authenticate itself. The default value of this parameter is **false**.

Note The terms *client* and *server* correspond to roles taken in the connection. The server role is taken by the side that accepts incoming connections. For example, in an SDS connection, the primary takes the server role; in a presentation client connection, the application server takes the server role.

The **ServerCertificateToSendName** parameter specifies the name of the certificate the server will send to the client to authenticate itself. You must specify this parameter. The default value of **Any** indicates that Jade will select the first suitable certificate from the certificate store; that is, a trusted server authentication certificate with the name of the device.

If you are not using client authentication, only the server must send a certificate. Therefore, only the computer the server is on must have a certificate.

The **ClientCertificateToSendName** parameter specifies the name of the certificate the client will send to the server to authenticate itself. This parameter is required only if the server requires it. The **Any** value indicates that Jade will select the first suitable certificate from the certificate store; that is, a trusted client authentication certificate. The default value of **None** indicates that Jade will not attempt to send a client certificate. In this case, if the server requires a certificate, the connection will fail.

The **AcceptableClientCertificateToReceiveNames** parameter is a comma-separated list of certificate names the server will accept if client authentication is required. If you specify the default value of **Any**, the server will accept any trusted client authentication certificate. If you want to restrict this further (for example, by service name), the server will look at this list. We therefore introduce the distinction between *trusted* and *acceptable* certificates. Certificates can be *trusted* or *acceptable*. All acceptable certificates must be trusted, but not all trusted certificates must be acceptable.

Note Any user who runs Jade requires read access to the private key of the certificate.

The following are configuration examples.

```
[Connections]
Default=Config1
JadeHttpToClient=Config2
SDS=Config3
ClientToRap=Config4

[Config1]
;This configuration does not use encryption
UseEncryption=false

[Config2]
;This configuration enables encryption and looks for a default certificate
;to send. The client will not send a certificate unless asked. This is the
;minimal configuration to enable SSL. If Jade can't find a default
;certificate, the connection will fail. Use SslConfigurationTool.exe to
;verify this configuration before starting Jade. This configuration may
;cease working if you are relying on certificates you don't control, but is
;good for test clusters where you don't want to specify the name of the
;computer but do have certificates installed.
UseEncryption=true

[Config3]
;This configuration is the same as [Config2], except that a specific certificate
;name will be sent to the client. This is more stable than the above [Config2].
UseEncryption=true
```

```
RequireClientAuthentication=false
ServerCertificateToSendName=subdomain.domain.tld

[Config4]
;This configuration requires that the client send a certificate with the names
;subdomainClientCertificateName or subdomainClientCertificateNameSpecial.
;This configuration says that the client will only attempt to send certificates
;with the name subdomainClientCertificateName.
UseEncryption=true
RequireClientAuthentication=true
ServerCertificateToSendName=subdomain.domain.tld
ClientCertificateToSendName=subdomainClientCertificateName
AcceptableClientCertificateToReceiveNames=
subdomainClientCertificateName,subdomainClientCertificateNameSpecial

[Config5]
;This configuration is the partner for the [Config4] configuration, but is only
;read by clients. Clients that use this configuration will only send certificates
;with the name subdomainClientCertificateNameSpecial.
UseEncryption=true
ClientCertificateToSendName=subdomainClientCertificateNameSpecial
```

The following is a **jade.ini** Jade initialization file configuration example.

```
[JadeSsl]
ConfigurationStore=[path of the Jade SSL configuration file]
```

The following is an example of the SSL configuration file.

```
[Connections]
Default=[configuration section n]
JadeHttpToClient=[configuration section n+1]
SDS=[configuration section n]
ClientToRap=[configuration section n]

[configuration section n]
UseEncryption=[true|false|opportunistically]
RequireClientAuthentication=[true|false]
ServerCertificateToSendName=[Any|certificate name]
ClientCertificateToSendName=[Any|None|certificate name]
AcceptableClientCertificateToReceiveNames=[Any|a certificate name,another
certificate name]
```

SslConfigurationTool.exe

The industry knows that if encryption is hard, people don't use it. One of our goals has therefore been to make configuring the system as simple as possible. To this end, we provide the **SslConfigurationTool.exe** utility, which reads your Jade initialization file and the SSL configuration store to which it points, and attempts to validate your configuration by setting up a secure connection using the settings it has found. You can instruct the SSL configuration tool to take either the server or client role in a connection (allowing you to test across machines), or both roles simultaneously (for intra-machine connections or other testing).

The command line syntax is as follows (this application does not support overriding configuration store options on the command line).

```
SslConfigurationTool Action=[ValidateConfiguration|
                             SelectCertificateFromUI|
                             ListCertificates|
                             TryForDefaultConfiguration]
```

All errors and warnings (and successes!) are logged to the **SslConfigurationTool.log** file and is created in the same folder.

When **Action=ValidateConfiguration** is specified:

- [JadeIniFile|SslConfigFile]=[*path to file*]

  If **JadeIniFile** is specified, the tool reads that initialization file to look for the JadeSsl section and the **ConfigurationStore** option, and tries to open that SSL Configuration file. This is the most useful argument, because it most closely simulates the way Jade will behave.

  If **SslConfigFile** is specified, the tool looks directly at that SSL Configuration file.

- ConnectionName=[*connection defined in the SSL Configuration Store*]
- ConnectionSide=[ClientSide|ServerSide|BothSides]
- ServerPortNumber=[*port number to listen on as server or connect to as client*]
- ServerHostname=[*host name to connect to as client*]

The **ValidateConfiguration** action is the main use case for this tool, and it attempts to set up a secure connection as Jade would. The connection type to set up is defined by the **ConnectionName** option, and corresponds to the **Connections** options in the SSL Configuration file. The tool then attempts to set up a secure connection as a server or a client, or as both ends, depending on the value of the **ConnectionSide** option. The tool uses the **ServerPortNumber** and **ServerHostname** values to attempt to set up the connection. You can omit one or both of these options, as the tool has default values of port **27657** and the fully qualified domain name of the computer. To make things easy, the server side of the connection ignores the **ServerHostname** option and always listens for all incoming TCP/IP connections on the specified port.

When the **SelectCertificateFromUI** action is specified, the tool attempts to make a secure connection using a certificate that you select from a dialog. The name of the certificate will be logged in a way that you can enter straight into the SSL Configuration options, which is useful because there are sometimes many names on a certificate. You can optionally specify the following command line arguments.

- ServerPortNumber=[*port number to listen on as server or connect to as client*]
- ServerHostname=[*host name to connect to as client*]

When the **ListCertificates** action is specified, the tool lists the available certificates and their trust status. Although it is more useful to look at certificates using the Microsoft Management Console, this action logs the names of the certificates.

When the **TryForDefaultConfiguration** action is specified, the tool tries to establish a secure connection using the **"Any" ServerCertificateToSendName** policy, and logs the name of the selected certificate. Client authentication is not attempted. You can optionally specify the following command line arguments.

- ServerPortNumber=[*port number to listen on as server or connect to as client*]
- ServerHostname=[*host name to connect to as client*]

Example

The following command line example tells the tool to test the configuration linked to in the file **F:\workspace\end-to-end-ssl\jade-db\samplejade.ini** as if it were starting up a web-enabled application on the same machine as that on which IIS is running, and use port **6547**.

```
SslConfigurationTool.exe
JadeIniFile="F:\workspace\end-to-end-ssl\jade-db\samplejade.ini"
```

```
Action=ValidateConfiguration ConnectionName=JadeHttpToClient
ConnectionSide=BothSides ServerPortNumber=6547
```

Applies to Version: 2020.0.01 and higher

Opportunistic Transport Layer Security (OTLS) from the TcpIpConnection Class

From Jade 2022.0.01, Jade provides support for Opportunistic Transport Layer Security (OTLS) from the **TcpIpConnection** class, which allows encrypted connections with Microsoft Exchange servers.

» To use OTLS

1. Put settings like the following into your SSL configuration file.

```
[Connections]
connection-name=section-name
[section-name]
UseEncryption=opportunistically
```

Note The **UseEncryption** parameter in the [Connections] section of the SSL configuration file has been changed from **UseEncryption=[true|false]** to **UseEncryption=[true|false|opportunistically]**, to make it impossible to mistake that you are not using always-on encryption.

You can configure only a user-defined connection name to use OTLS. If an internal **JadeSsl** connection is configured to use OTLS, an error will be returned. The **Default** name is also rejected, because it can represent all of the others.

2. Create a **TcpIpConnection** object (or a pair of **TcpIpConnection** objects).
3. Call **TcpIpConnection**::**setConnectionTypeName***(connection-name)* on those objects.
4. Connect the connection.
5. Send any amount of data over the connection.
6. Call **TcpIpConnection**::**makeConnectionSecure** to secure the connection with encryption.

Setting the Connection Type Name

In earlier releases, all web applications that make use of the **MultiWorkerTcp** subsystem were hard-coded to use the "**JadeHttpToClient**" connection name. This was appropriate when all web applications sat in behind IIS, but this is not as useful for web service applications (that is, Direct REST and Direct SOAP web service applications) that connect directly to incoming HTTPS traffic.

The [WebOptions] section of the Jade initialization file provides the **ConnectionTypeName** parameter that specifies the name of the connection type, to control the SSL configurations used by web-based applications. This parameter enables application-specific configuration of secure communications.

The **JadeMultiWorkerTcpTransport** class provides the **setConnectionTypeName** method that allows users to control which Secure Sockets Layer (SSL) configuration is applied to the web-based application using the **JadeMultiWorkerTcpTransport** class.

Tip For details about SSL end-to-end encryption, see "End-to-End SSL Encryption", earlier in this chapter.

If the **ConnectionTypeName** parameter does not have a specified connection type name, the value of the **JadeMultiWorkerTcpTransport** class **setConnectionTypeName** method defaults to **JadeHttpToClient**.

The following parameter and value example specifies the connection type name for all web-based applications on the node.

```
ConnectionTypeName=Milly
```

However, you can specify the setting in the format <***application-name***>**_ConnectionTypeName** to affect only a specific application; for example, to apply the connection type name to a REST application called **RestServer**, specify the following.

```
RestServer_ConnectionTypeName=Molly
```

The connection type names specified in the [WebOptions] section of the Jade initialization file could be as follows.

```
[WebOptions]
ConnectionTypeName=Milly
RestServer_ConnectionTypeName=Molly
AnotherApp_ConnectionTypeName=Mandy
```

Note The application name is case-sensitive, and Jade does not support spaces in application names.

Applies to Version: 2022.0.05 and higher

Connecting to Jade Applications from Internet Information Server (IIS)

Your connections from the Internet Information Server (IIS) to Jade applications are via a TCP/IP connection. (For more details, see "Configuring JadeHttp for Remote Connections", later in this chapter.)

To implement a TCP/IP connection from IIS to Jade applications, define a section within the **jadehttp.ini** file for each application, with the **ApplicationType** parameter and unique multiple **TcpPort[*n*]**, **MinInUse[*n*]**, **MaxInUse[*n*]**, **TcpConnection[*n*]**, **CloseDelay[*n*]**, and **ConnectionGroup[*n*]** parameters within each [*application-name*] section. Set the **ApplicationType** parameter in each [*application-name*] section to **RestServices**, **WebEnabledForms**, **WebServices**, or **HtmlDocuments**. The following is an example of these parameters.

```
[HtmlCompany]
ApplicationType = HtmlDocuments
TcpConnection   = Host1a
TcpPort         = 22000
MinInUse        = 2
MaxInUse        = 10
CloseDelay      = 600
TcpConnection2  = Host2a
TcpPort2        = 22001
MinInUse2       = 2
MaxInUse2       = 10
CloseDelay2     = 600
```

Connecting to Jade Applications using the WebSocket Protocol

Jade supports the WebSocket communications protocol, which enables interaction between a web client (such as a browser) and a Jade server-side application, facilitating real-time data transfer from and to the server. This is made possible by providing a standardized way for the Jade server-side application to send content to the client without being first requested by the client, and allowing messages to be passed back and forth while keeping the connection open. In this way, a two-way ongoing conversation can take place between the client and server-side application.

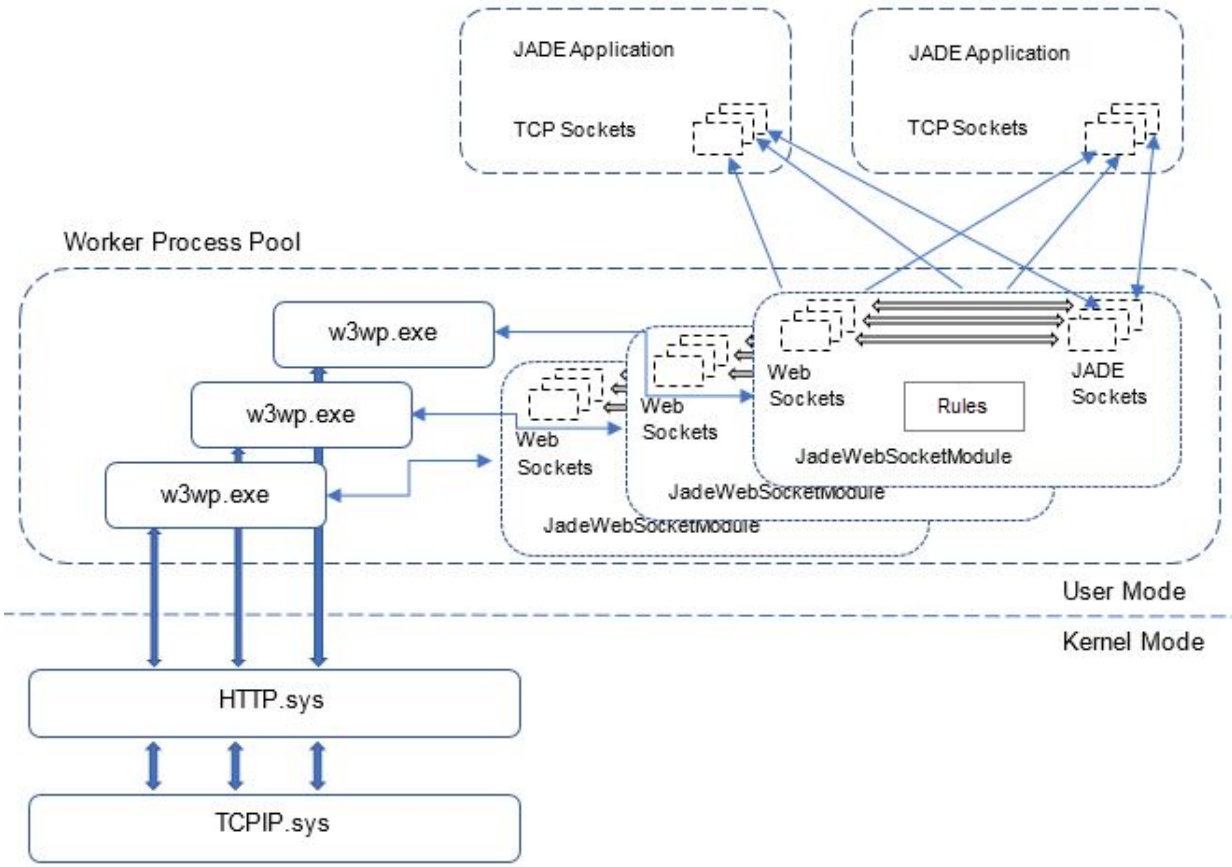
The protocol consists of an opening handshake followed by basic message framing, layered over TCP. This provides browser-based applications that need two-way communication with servers to do so without relying on opening multiple HTTP connections.

IIS has to be correctly configured to handle incoming WebSocket connections and forward requests that will be handled by Jade to the **JadeWebSocketModule** (that is, the **JadeWebSockets_IIS.dll** native IIS module).

A **JadeWebSocketModule** instance loaded into an IIS worker process maps WebSocket connections to Jade application instances using URL pattern-matching rules that yield a host (network name or IP address) and TCP port number.

Note The WebSocket protocol is not supported by Jade applications connected to an Apache HTTP Server.

A simple message protocol is used to multiplex the WebSocket sessions and their messages over each Jade application TCP connection, as shown in the following diagram.



As IIS is typically configured to run multiple worker processes, multiple instances of the **JadeWebSocketModule** can be active at the same time, so Jade applications must support multiple TCP connections.

For details about:

- Configuring the WebSocket protocol, see "Configuring the WebSocket Module", later in this chapter. See also the See also the **JadeWebSocket**, **JadeWebSocketServer**, and **WebSocketException** classes in Volume 2 of the *Encyclopaedia of Classes*.
- IIS WebSocket protocol support, refer to the Microsoft website.

Applies to Version: 2018.0.01 and higher

Connecting to Jade Applications from an Apache HTTP Server

Your connections from the Apache HTTP Server to Jade applications can be via a TCP/IP connection only. The design and configuration of the **mod_jadehttp** module fully conforms to Apache HTTP Server 2.4 module usage and configuration practices.

You can obtain more information about the Apache Software Foundation from the Apache website.

Jade supports the Apache HTTP Server by supplying **mod_ jadehttp**. Only **JadeInternetTCPIPConnection** class TCP/IP communications are supported for all versions of **mod_ jadehttp**, regardless of the operating system; that is, TCP/IP connections only are used for connections to Jade applications running from the Apache HTTP Server. For details, see "Using the mod_jadehttp Module", in the following subsection.

The **mod_ jadehttp** library module supplied by Jade for the Apache HTTP Server implements support for the following Multi-Processing Modules (MPMs). The selection of the default MPM is selected based on the operating system on which the Apache server is running; that is, **mpm_winnt** is the default value for Microsoft-based operating systems.

If you build the Apache server from source code, you can build and select whichever MPM you want to use, on the operating system of your choice.

Some relevant MPMs with respect to Jade and the **mod_ jadehttp** library are listed in the following table.

Multi-Processing Module (MPM)	Implements a …
mpm_winnt	Multiple-threaded single process web server
prefork	Non-threaded, pre-forking web server
worker	Hybrid multiple-threaded multiple-process web server

The **mod_ jadehttp** module ensures that the same destination IP address and port number combination are used for a specific logical connection, by using the hidden fields that both the **jadehttp** and the **mod_ jadehttp** libraries insert into the HTML data.

» To install the mod_ jadehttp module

- Copy the **mod_ jadehttp.so** file into the **modules** directory of your Apache HTTP Server.

Notes The name **mod_ jadehttp.so** applies to all operating systems.

Ensure that you use the correct operating system binary.

Apache and the HTTP generally use UTF-8 as the encoding scheme for Unicode data on the Web. As **mod_jadehttp** currently does not allow for this and it passes the data directly to Jade, only ANSI data can be read from or written to the **JadeInternetTCPIPConnection** object.

For details about configuring the Apache HTTP Server, see "Configuring Apache for Remote Connections", later in this chapter.

Using the mod_ jadehttp Module

Jade includes **mod_ jadehttp32.so** and **mod_ jadehttp64.so**, for the 32-bit and 64-bit version of Apache 2.4.10, respectively.

To use the mod_ jadehttp module

1. Ensure that you are running Apache 2.4.10 or a higher 2.4.*n* version.
2. Determine whether 32-bit or 64-bit Apache is installed. (If the operating system installation is a 64-bit version, it is most likely that Apache is also the 64-bit version.)
3. Load the correct **mod_ jadehttp** module into Apache. Note that path names in following examples may differ from those in your environment.

   For 32-bit versions of Apache:

   ```
   LoadModule jadehttp_module modules/mod_jadehttp32.so
   ```

   For 64-bit versions of Apache:

   ```
   LoadModule jadehttp_module modules/mod_jadehttp64.so
   ```

Configuring Your Jade Software

The Jade initialization file enables you to configure Jade to your requirements. For details, see "Jade Initialization File", in the *Jade Initialization File Reference*.

The physical database layer or database engine of the Jade Object Manager can use the information stored in the [PersistentDb] and [TransientDb] sections of the Jade initialization file to configure itself to meet defined performance or resource constraints, or user preferences.

Note In a multiple-database environment, each database can have its own unique Jade initialization file and configuration parameters.

You can also:

- Set options to specify your user configuration in the Jade command line, for use when running a Jade application from outside the development environment. For details, see "Jade Configurations", in Chapter 1, and "Format of the Server URI String", earlier in this chapter.
- Specify your installation global preferences that apply to all Jade Browser and Painter windows for your Jade Platform development environment work sessions in the installed Jade release. For details, see "Specifying Your Administration Options", later in this chapter.

- Specify options in the **jadehttp.ini** file for use when accessing Jade applications from the Internet Information Server (IIS). For details, see "Configuring JadeHttp for Remote Connections", later in this chapter.
- Specify options in the **JadeWebSockets_IIS.ini** file for use when accessing Jade applications from the WebSocket protocol. For details, see "Configuring the WebSocket Protocol", later in this chapter.
- Specify options in the **mod_ jadehttp** module, for use when accessing Jade applications from the Apache HTTP Server. For details, see "Configuring Apache for Remote Connections", later in this chapter.
- Scale the dynamic connection worker pool. For details, see "Scaling Dynamic Worker Pool Connections", later in this chapter.

Configuring the Internet Protocol Version

Jade supports Internet Protocol Version 4 (IPv4) and Internet Protocol Version 6 (IPv6).

These transport protocols are used in the following areas of Jade.

- Remote Procedure Call (RPC), between database server and nodes
- Thin client, between the application server and presentation clients
- Synchronized Database Service (SDS), between SDS nodes
- Web, between the node and the Web server add-in (that is, **jadehttp.dll** or **mod_ jadehttp.so**)
- User code in your application that uses the **TcpIpConnection** class
- ODBC, between the thin client and server

The following Jade initialization file parameter values refer to networking.

- [JadeServer] section and **NetworkSpecification** parameter
- [ConnectionParams] section **NetworkSpecification** and **ServerNodeSpecifications** parameters
- [JadeClient] section **ProxyHost** and **ServerNodeSpecifications** parameters
- [JadeAppServer] section **AppServer** parameter
- [JadeThinClient] section **AppServer** and **SSLProxyHost** parameters
- [JadeOdbcServer] section **ListenerHostName** and **ListenPort** parameters

The system administrator should review the following host name and IP address values of the following properties for potential changes, to enable the IPv6 feature. Jade developers who expect a deployed system to support the IPv6 feature should review their code that accesses these properties, as these properties can now return different values and break assumptions that you may have previously made. The most-common change is that IP addresses will have a *dot quad number* format (for example, 192.168.254.254) and be a maximum of 16 characters. For details, see "Limits and Defaults", later in this chapter.

- **TcpIpConnection**::**localInterface**
- **TcpIpConnection**::**localIpAddress**
- **TcpIpConnection**::**remoteIpAddress**

- **TcpIpConnection**::**remoteName**
- **TcpIpConnection**::**protocolFamily**
- **Connection**::**name**
- **JadeTcpIpProxy**::**host**
- **JadeMultiWorkerTcpTransport**::**listenHostname**
- **JadeMultiWorkerTcpConnection**::**localAddress**
- **JadeMultiWorkerTcpConnection**::**remoteAddress**
- **JadeMultiWorkerTcpConnection**::**getFullName**
- **JadeMultiWorkerTcpConnection**::**getLocalHostname**
- **JadeMultiWorkerTcpConnection**::**getRemoteHostname**
- **Process**::**networkAddress**
- **Node**::**getComputerName**
- **Node**::**networkAddress**

RPC, Database, and Node Configuration

This section covers the area of behavior and configuration between the database server (including SDS environments) and client nodes.

Configuring the Database Server

Configure the database server by specifying the following values for the **NetworkSpecification** parameter in the [JadeServer] section of the Jade initialization file, as follows.

```
NetworkSpecification<specification-number> = transport-
type,enabled|disabled,listener-port[,interface]
```

The values for the *transport-type* variable relevant to this feature can be **TcpIP**, **TcpIPv4**, or **TcpIPv6**. The *transport-type* literal value is case-insensitive.

The **TcpIP** value is a synonym for **TcpIPV4**. **TcpIPv4** provides IP version 4 connections only. **TcpIPv6** provides IP version 6 connections only. If you want the server to support both network protocols, you must define a network specification for each protocol.

You can specify the *interface* value as a host name or an IP address. If you use an IP address, the address must be in an appropriate format for the specified *transport-type* value.

Configuring the Client Node

Configure the client node by:

- Specifying the following values for the **ServerNodeSpecifications** parameter in the [JadeClient] section of the Jade initialization file.

```
ServerNodeSpecifications = transport-type,remote-host,remote-port
[,local-interface[,local-port]]
```

The values for *transport-type* variable relevant to this feature can be **TcpIP**, **TcpIPv4**, **TcpIPv6**, or **TcpIPAny**. The *transport-type* literal value is case-insensitive. The **TcpIP** value is a synonym for **TcpIPAny**. **TcpIPv4** provides IP version 4 connections only. **TcpIPv6** provides IP version 6 connections only. The **TcpIPAny** value supports IP version 6 or IP version 4 connections.

You can specify the *remote-host* and *local-interface* values as a host name or an IP address. If you use an IP address, the address must be in an appropriate format for the specified *transport type* value. If you specify a host name, all DNS-provided addresses will be attempted. For the **TcpIPAny** transport type, the client will first attempt to connect via IP version 6 and then IP version 4 protocol on the provided IP addresses. Each connection failure will be logged, and the next available combination tried.

- Using the **server** argument in a command line to specify the server Universal Resource Identifier (URI) target database and the client-server transport, instead of the **server** and **path** arguments in a command line and the **ServerNodeSpecifications** parameter in the [JadeClient] section of the Jade initialization file. For details, see "Format of the Server URI String", earlier in this chapter.

Configuring SDS Connections

Configure SDS connections by specifying the following values for the **NetworkSpecification** and **ServerNodeSpecifications** parameters in the [ConnectionParams] section of the Jade initialization file.

```
NetworkSpecification<specification-number> = transport-
type,enabled|disabled,listener-port[,interface]

ServerNodeSpecifications = transport-type,remote-host,remote-port[,local-
interface[,local-port]]
```

Both the server side (specified in the **NetworkSpecification** parameter) and the client side (specified in the **ServerNodeSpecifications** parameter) follow the same behavior and rules documented for the database server and client node RPC configurations in the previous sections.

Application Server and Thin Client Configuration

This section covers the area of behavior and configuration between the application server and the presentation clients.

Configuring the Application Server

Configure the application server by specifying the following values for the **AppServer** parameter in the [JadeAppServer] section of the Jade initialization file, as follows.

```
AppServer = [transport-type,]interface
```

The values for the *transport-type* variable relevant to this feature can be **TcpIP**, **TcpIPv4**, **TcpIPv6**, or **TcpIPAny**. The *transport-type* literal value is case-insensitive.

If you do not specify a *transport-type*, the default value is **TcpIP**, which is a synonym for **TcpIPV4**. **TcpIPv4** provides IP version 4 connections only. **TcpIPv6** provides IP version 6 connections only. **TcpIPAny** supports IP version 6 or IP version 4 connections. Depending on your requirements, you can configure a single application server to use **TcpIPAny**, or you can set up two application servers so that one supports **TcpIPV4** and the other **TcpIPV6**.

You can specify the *interface* value as a host name or an IP address. If you use an IP address, the address must be in an appropriate format for the selected *transport-type* value.

Because of the way that Jade parses the command line, enclose the parameter values in quote marks if you specify **ApplicationServer="*transport-type*,*interface*"** on the command line.

Configuring a Presentation Client

Configure a presentation client by specifying the following values for the **AppServer** and **SSLProxyHost** parameters in the [JadeThinClient] section of the Jade initialization file, as follows.

```
AppServer = [transport-type,]interface

SSLProxyHost = interface
```

The values for the *transport-type* variable of the **AppServer** parameter relevant to this feature can be **TcpIP**, **TcpIPv4**, **TcpIPv6**, or **TcpIPAny**. The *transport-type* literal value is case-insensitive.

If you do not specify a *transport-type*, the default value is **TcpIP**, which is a synonym for **TcpIPAny**. **TcpIPv4** provides IP version 4 connections only. **TcpIPv6** provides IP version 6 connections only. **TcpIPAny** supports IP version 6 or IP version 4 connections.

You can specify the *interface* value as a host name or an IP address. If you use an IP address, the address must be in an appropriate format for the selected *transport-type* value. If you specify a host name, all DNS-provided addresses will be attempted. For the **TcpIPAny** transport type, the presentation client will first attempt to connect via IP version 6 and then IP version 4 protocol on the provided IP addresses. Each connection failure will be logged, and the next available combination tried.

Because of the way that Jade parses the command line, enclose the argument values in quote marks if you specify **ApplicationServer="*transport-type*,*interface*"** on the command line.

If SSL has been enabled and the presentation client needs to connect to the application server via a proxy host, the presentation client will attempt to connect to the proxy server using the same protocol (*transport-type*) as that specified in the **ApplicationServer** parameter configuration. How the proxy host handles TCP/IP version 6 is dependent on the proxy host implementation.

The areas to check and to test with the proxy host are as follows.

1. Will it accept an IP version 6 connection?
2. What protocol will it use between the proxy and the application server?

TcpIpConnection Class

This section covers the area of behavior and configuration in using Jade RootSchema-based networking classes.

The **TcpIpConnection** class provides the virtual **protocolFamily** property, which is an Integer value. The values that can be assigned to this property are the **TcpIpConnection** class constants listed in the following table.

Class Constant	Integer Value
ProtocolFamilyTcpIPv4	0
ProtocolFamilyTcpIPv6	1
ProtocolFamilyTcpIPAny	-1

If you do not change your code, your existing code runs using TCP/IP version 4 only.

Note The **JadeTcpIpProxy** class works only with the TCP/IP version 4 protocol.

Other User Networking Areas

This section contains other areas of your network that may be affected by this feature.

The **TcpConnection** parameter in the **jadehttp.ini** file specifies a host or IP address to which to connect.

If you specify a host name, all DNS-provided address will be attempted. Both the **TcpIPv6** and **TcpIPv4** protocols will be attempted on the provided IP addresses. Each connection failure will be logged, and the next available combination tried.

Limits and Defaults

Your user code must allow for longer network addresses and host names.

A fully qualified hostname can be up to 255 characters, with each label being up to 63 characters (for details, see the RFC2181 clarification to the DNS specification). However, the underlying C++ code allows for 1,025, so allow for a host name of 1,024, to make your code safe. A fully defined IPv6 address, including port number, can be up to 65 characters.

The following is taken from a Microsoft header file, and agrees with other Internet descriptions.

```
// The totals are derived from the following data:
//  15: Ipv4 address
//  45: Ipv6 address including embedded Ipv4 address
//  11: Scope Id
//   2: Brackets around Ipv6 address when port is present
//   6: Port (including colon)
//   1: Terminating null byte
//
#define INET_ADDRSTRLEN  22
#define INET6_ADDRSTRLEN 65
```

Configuring the WebSocket Module

Configure the **JadeWebSocketModule** using the **JadeWebSockets_IIS.ini** initialization file that has the same syntax as other Jade initialization files.

Specify logging-related configuration parameters in the [JadeWebSocket Logging] section and WebSocket routing configuration rules in the [JadeWebSocket Rules] section.

Note The **JadeWebSockets_IIS.ini** file must exist and it must contain at least one rule.

For details about configuring the WebSocket module, see:

- Location of the WebSocket Initialization File
- [JadeWebSocket Logging] Section
- [JadeWebSocket Rules] Section
- Configuring the WebSocket Protocol in IISMgr

Applies to Version: 2018.0.01 and higher

Location of the WebSocket Initialization File

The **JadeWebSockets_IIS.ini** file is located in the same way web services find the **jadehttp.ini** file; that is, the native module **...\jade\bin\JadeWebSockets_IIS.dll** looks for the initialization file **...\jade\bin_JadeWebSockets_IIS\ini\JadeWebSockets_IIS.ini**.

The log files are located in the same way web services locate the jadehttp.log files; that is, the native module **...\jade\bin\JadeWebSockets_IIS.dll** creates log files in the **...\jade\bin_JadeWebSockets_IIS\logs** directory.

Applies to Version: 2018.0.01 and higher

[JadeWebSocket Logging] Section

The [JadeWebSocket Logging] section of the **JadeWebSockets_IIS.ini** file controls logging by the **JadeWebSockets_IIS.dll** library file.

Applies to Version: 2018.0.01 and higher

The [JadeWebSocket Logging] section can contain the following parameters.

Trace

Value Type Boolean

Default False

Purpose

The **Trace** parameter controls the logging of **JadeWebSockets_IIS.dll** library activity.

No message content is ever logged. When the value of this parameter is **false**, logging is suppressed completely.

When the value of this parameter is set to **true**, logged messages acknowledge only that a message has been received or sent, because it is not possible to distinguish what is sensitive data and what is not. Setting this parameter to **true** does *not* include any of the text sent or received from the client, as this text could contain personal information, passwords, credit card details, and so on.

Note You cannot use this parameter to inspect and debug data passing through the **JadeWebSockets_IIS** library.

Applies to Version: 2018.0.01 and higher

TraceFile

Value Type String *(file-path)*

Default Default

Purpose

The **TraceFile** parameter specifies an optional path and a file name that overrides the default **jadeWebSockets_IIS<*timestamp*>.log** file name so that log files can be placed in other directories; for example, **c:\temp\testWebSocks.log** creates **testWebSocks<*timestamp*>.log** files in the **c:\temp** folder.

If you define a value for the **TraceFile** parameter, the default log file in the default location is neither created nor used.

Applies to Version: 2018.0.01 and higher

TraceFileSize

Value Type Integer (bytes)

Default 1000000

Purpose

The **TraceFileSize** parameter specifies the maximum file size before switching logs.

Applies to Version: 2018.0.01 and higher

[JadeWebSocket Rules] Section

The [JadeWebSocket Rules] section of the **JadeWebSockets_IIS.ini** file can contain one or more rules to dispatch incoming WebSocket connections, based on their URL, to Jade application instances.

Each rule assigns a URL pattern to a specific host and port.

Note The [JadeWebSocket Rules] section must contain at least one **Rule** parameter.

Applies to Version: 2018.0.01 and higher

Rule

Value Type String (*Rule=URL-pattern*,*host*,*port*)

Purpose

The ***Rule*** parameter specifies the rule to dispatch incoming WebSocket connections, based on its URL, to Jade application instances.

The ***Rule*** value can be any name, as it does not influence the dispatching of incoming connections.

The *URL-pattern* value is a C++/ECMAScript regular expression. To activate this rule, the request URL must match the pattern.

The *host* value is a network-resolvable host name or IP address for the machine that is hosting the Jade application.

The *port* value is the TCP port being handled by the Jade application.

Caution If you specify a port (for example, from a localhost port 80 request) and you did not include the port in the rule, the request is ignored

Rules are tried in the order in which they appear in the **JadeWebSockets_IIS.ini** file; that is, earlier-defined rules take precedence over later rules.

The following is an example of the [JadeWebSocket Rules] section.

```
[JadeWebSocket Rules]
; http or https URL ending with -1.ws is forwarded to port 5555 on the local host
ws5555='https?://.*-1.ws',localhost,5555

; forward all other http or https websocket URLs to localhost port 4444
default='https?://.*',localhost,4444
```

Applies to Version: 2018.0.01 and higher

Configuring the WebSocket Protocol in IIS

To configure the WebSocket protocol in IIS and enable the **JadeWebSocketModule** for the website, run the **Internet Information Services (IIS) Manager** from the right panel of the **Administrative Tools**, accessed from the Control Panel.

Although these instructions use the default website, you can create a separate website with one or more application pools.

To install the JadeWebSocketModule

1. In the **Connections** panel at the left, select the web server; for example, **CNWCHCS86**.

   The middle panel then displays *server-name* Home; for example, **CNWCHCS86 Home**.

2. Make sure that the middle panel is in **Features View** rather than **Content View**.

   Switch between views by using the context menu of the web server in the left panel or the icons at the bottom of the middle panel.

3. Double-click **Modules** in the middle panel. The **Modules** page is then displayed.

4. Select **Configure Native Modules** from the context menu in the middle panel and then click the **Register** button in the Configure Native Modules dialog.

5. In the Register Native Module dialog, use **JadeWebSocketModule** for the name and then locate the **jadeWebSockets_IIS.dll** in the Jade **bin** directory for the path.

6. Click **OK** to register the DLL and then enable the **JadeWebSocketModule** by checking the **JadeWebSocketModule** check box in the Configure Native Modules dialog. The **JadeWebSocketModule** is then displayed in the list on the **Modules** page.

7. IIS loads modules for requests in a specified order. Note that the **WebSocketModule** must be loaded before the **JadeWebSocketModule**.

   a. Select **View Ordered List** from the **Actions** panel. The modules are then listed in the order in which they are loaded.

   b. Select the **WebSocketModule** in the list and then select **Move Up** or **Move Down** from the **Actions** panel until the **WebSocketModule** is displayed before the **JadeWebSocketModule** in the module order.

To enable the JadeWebSocketModule for the website

1. In the **Connections** panel at the left, select the website under your web server; for example, **Default Web Site**.

   The middle panel then displays *web-site-name* Home; for example, **Default Web Site Home**.

2. Double-click **Handler Mappings** in the middle panel.

3. Select **Add Module Mapping** from the context menu in the middle panel or click **Add Module Mapping** in the **Actions** panel at the right.

4. Complete the module mapping in the Add Module Mapping dialog, as follows.

   a. In the **Request path** text box, enter the URL path you want to be handled by the module; for example, ***.ws**.

   b. In the **Module** drop-down list, select the **JadeWebSocketModule** that you configured in the previous instruction in this topic that installed the **JadeWebSocketModule**.

c. Leave the **Executable** control blank; it is not needed.

d. In the **Name** text box, enter **JadeWebSocketHandler**.

5. Click **Request Restrictions** and then work through the tabs as follows.

a. **Mapping** - disable (that is, uncheck) the **Invoke handler only if request is mapped to** check box.

b. **Verbs** - select **All verbs**.

c. **Access** - select **Script**.

Applies to Version: 2018.0.01 and higher

Configuring JadeHttp for Remote Connections

The **jadehttp** library module can handle multiple TCP/IP address connections to the same application, which allows the web server to connect to multiple copies of the same Jade application on multiple hosts. For details, see "[*application-name*] Section", later in this chapter. To handle the transfer of large files from a user's browser to Jade for **JadeInternetTCPIPConnection** class connections, data is read in small pieces and written into a temporary file as it goes.

If the **Firewall** parameter in the [Jadehttp Files] section of the JadeHttp initialization file or the **Firewall** configuration directive in the Jade **mod_ jadehttp** module and the **Firewall** parameter in the [WebOptions] section of the Jade initialization file are set to **true**, the contents of the temporary file is sent (in pieces) to Jade when the entire data stream has been received. Jade then reads those pieces and writes the file contents as it goes (therefore avoiding large memory allocations). This is transparent if the **Firewall** parameter or directive is set to **false** or if it is set to **true** and one of the following applies.

- The **FileTransferDirectory** parameter in the [Jadehttp Files] section of the **jadehttp.ini** file or the **FileTransferDirectory** configuration directive in the Jade **mod_ jadehttp** module is set to **default** or to a valid directory (used to hold temporary file contents as the data is read from the Internet).
- The standard **readPipeCallback** method of the **JadeInternetTCPIPConnection** class is used to read the data.

Note The firewall must be enabled at both ends of the connection (that is, if you are uploading images from another machine for a Jade web-enabled form over a TCP/IP connection, the **Firewall** parameter in the [Jadehttp Files] section of the JadeHttp initialization file or the **Firewall** configuration directive in the Jade **mod_ jadehttp** module and the **Firewall** parameter in the [WebOptions] section of the Jade initialization file must both be set to **true**).

The **jadehttp.ini** file is located in the virtual directory (that is, the IIS server location). Each virtual directory has its own **jadehttp.ini** file. (For details about the paths, see "[Jadehttp Files] Section", later in this chapter.)

The JadeHttp initialization file or Jade **mod_ jadehttp** module is read-only when the first request for an application is received. If you update a parameter value in this file, you must therefore stop the Internet service and then restart it.

Firewall for the Jade Internet Environment

As the **JadeHttp** library or the Jade **mod_ jadehttp** module can communicate via a TCP/IP connection to a different workstation running the Jade application, you can ensure that the Jade environment is entirely behind the firewall. In the default implementation of the Jade Internet environment, the Jade application writes image files directly into the virtual directory for subsequent web browser access.

Files transferred by a web browser to the **JadeHttp** library are written into a directory for access by the Jade application.

Use of a firewall requires that there are no network directory services running between the web browser and Jade environments.

The only available links are TCP/IP connections. Jade applications do not write files directly into the virtual directory outside the firewall. Instead, the files must be passed via the TCP/IP connection to the **JadeHttp** library, which creates the file in the virtual directory. Similarly, files arriving at the **JadeHttp** library from a web browser are passed to the Jade application via the TCP/IP connection.

The Jade application writes these files into the Jade web application directory.

Files Created by the Jade Application

Use the **Firewall** parameter in the [Jadehttp Files] section of the JadeHttp initialization file or the **Firewall** configuration directive in the Jade **mod_ jadehttp** module to control whether Jade uses this facility for its automatic HyperText Markup Language (HTML) generation. When this parameter or directive is set to the default value of **false**, files are written directly to the virtual directory by Jade.

When you set this parameter or directive to **true** and the **Firewall** parameter in the [WebOptions] section of the Jade initialization file is also set to **true**, files are transferred to the **JadeHttp** library via the TCP/IP connection before being passed to the virtual directory.

The **JadeWebServiceProvider** and **WebSession** classes provide a **createVirtualDirectoryFile** method, which enables you to pass files created by a Jade application to the **JadeHttp** library.

The **createVirtualDirectoryFile** method has the following signature:

```
createVirtualDirectoryFile(fileName:     String;
                           fileContents: Binary;
                           retain:       Boolean): Integer;
```

The **createVirtualDirectoryFile** method parameters are listed in the following table.

Parameter	Description
fileName	Name of the file to be created in the virtual directory
fileContents	Binary holding the file contents
retain	Creates read-only files when set to **true** or standard files when set to **false**

The **JadeHttp** library creates the specified file in the directory (the virtual directory visible to web browsers) in which the library is running. This method returns zero (**0**) if the method successfully formats a request to the **JadeHttp** library or it returns the non-zero Windows error code indicating the failure to create the file.

The image files must be passed before the final reply to the web request is returned.

This process is transparent to you if your application is using the standard Jade generated Internet facility. However, if your application logic does additional file generation of its own, you must call the **JadeWebServiceProvider** or **WebSession** class **createVirtualDirectoryFile** method.

For web service applications, the virtual directory parameter *must* be specified and the **fileName** parameter contains only the file name. When an application transfers files to the **jadehttp** library by using the **createVirtualDirectoryFile** method for non-web service applications, the directory that is used is determined as follows.

- If the file name specified in the **fileName** parameter includes a directory, the file is written into that directory.
- If the file name does not include a directory and the application's section of the **jadehttp.ini** file or the Jade **mod_ jadehttp** module contains a directory value for the **VirtualDirectory** parameter or **PhysicalDirectory** directive, respectively, the files are written into that directory.

If neither the file name nor the [*application-name*] section of the **jadehttp.ini** file or the directive **ApplicationType** in the Jade **mod_ jadehttp** module contains a directory, the files are written into the same directory as the **jadehttp** library module. (This enables you to control directory permissions so that the directory containing the **jadehttp** library module does not need to have read and write access.)

Use the **JadeWebServiceProvider** or **WebSession** class **isVDFilePresent** method to return whether the requested file specified in the **fileName** parameter of the method is present on the web server side of the firewall when using the Jade web interface via the **jadehttp** library file. This method sends a message to the **jadehttp** library file to perform this action. If the specified file name does not have a directory part, the current virtual directory defined in the **VirtualDirectory** parameter in the **JadeHttp.ini** file or the **PhysicalDirectory** directive in the Jade **mod_ jadehttp** module for the application is used. The file specified in the **fileName** parameter of the **isVDFilePresent** method is used if the file name has a directory part.

Call the **JadeWebServiceProvider** or **WebSession** class **deleteVirtualDirectoryFile** method to delete files that are in the virtual directory (that is, the directory specified by the value of the **VirtualDirectory** parameter in the **JadeHttp.ini** file or the **PhysicalDirectory** directive in the Jade **mod_ jadehttp** module).

The **deleteVirtualDirectoryFile** method parameters are listed in the following table.

Parameter	Description
fileName	Name of the file to be deleted from the virtual directory
deleteIfReadOnly	Deletes files marked as read-only when set to **true**

This method returns zero (**0**) if the file deletion is successful or it returns a non-zero error code if the deletion fails.

Notes If your applications are not using the standard Jade generated Internet facility, you need to set the **Firewall** parameter in the [Jadehttp Files] section of the **jadehttp.ini** file or the **Firewall** configuration directive in the Jade **mod_ jadehttp** module and the [WebOptions] section of the Jade initialization file to **true** and call the **JadeWebServiceProvider** or **WebSession** class **createVirtualDirectoryFile** method *only* if you require firewall separation. If you do not require firewall separation, Jade creates image files directly into the virtual directory and bypasses the **JadeHttp** library.

The file cleanup process that is started when the **jadehttp.ini** file **Firewall** parameter or the Jade **mod_ jadehttp** module **Firewall** configuration directive is set to **true** deletes only files that are *not* read-only and which are of type .**jpg**, .**png**, or .**gif**. You should therefore make all other files in this directory that you want to retain read-only, by setting the **retain** parameter to **true**.

Files Transferred from the Web Browser

Use the **Firewall** parameter in the [Jadehttp Files] section of the **jadehttp.ini** file or the **Firewall** configuration directive in the Jade **mod_ jadehttp** module to control whether firewall separation is used when transferring files to Jade from web browsers. When the **Firewall** parameter or directive is set to **false**, the default mechanism is used.

When you set the **Firewall** parameter or directive to **true** and the **Firewall** parameter in the [WebOptions] section of the Jade initialization file is also set to **true**, files are transferred to the Jade application over the TCP/IP connection and each file transferred from a web browser is sent to the Jade application as a separate message. Your Jade application must distinguish this message from a normal web browser request and store the transferred file.

The **JadeHttp** library reads data in small pieces and writes it into a temporary file as it goes. The contents of the temporary file are sent (in pieces) when the entire data stream is received. Jade then reads those pieces and writes the file contents as it goes, which avoids large memory allocations. For details, see "Configuring JadeHttp for Remote Connections", earlier in this section.

For details about the location of files, see "Controlling the Location of Files Uploaded via a Web Application", later in this chapter.

The message sent through the firewall is formatted as follows.

File Component	Description
Header	**TRF**, followed by a null
File name	Name of the file that is to be created, followed by a null
File contents	Binary image

The handling of this message is placed in the **Connection** class **readPipeCallback** method before the rest of the general processing. (Note that no reply should be sent for this message.) When the Jade application receives the web browser request, it stores the file in the specified web application directory.

Note There is no cleanup method invoked if your user logic does not remove the files. Your application should delete these files when processing on them is complete.

When a file is transferred to Jade from a web browser by using the default connection mechanism, the text for the text box that generated the transfer is changed by the **JadeHttp** library to:

```
<original-file-name>;<temporary-file-name-and-path>
```

If you have set the **Firewall** parameter in the [Jadehttp Files] section in the **jadehttp.ini** file or the **Firewall** configuration directive in the Jade **mod_ jadehttp** module and the **Firewall** parameter in the [WebOptions] section of the Jade initialization file to **true**, the text is set to:

```
<original-file-name>;<temporary-file-name>
```

Your application logic that accesses the file must append the web application directory to the temporary file name to form the actual path. (To get the name of the web application directory, call the **Application** class **webApplicationDirectory** method, which returns a string containing the directory name.)

Internal Housekeeping of the Virtual Directory

When Jade sends a file to the **JadeHttp** library, any file that is not to be retained is created as a standard file of type .**jpg**, .**png**, or .**gif** when the **jadehttp.ini** file **Firewall** parameter or the Jade **mod_ jadehttp** module **Firewall** configuration directive, the **Firewall** parameter in the [WebOptions] section of the Jade initialization file, and the **retain** parameter in the **JadeWebServiceProvider** or **WebSession** class **createVirtualDirectoryFile** method are set to **true**. Files that are to be retained are created as read-only files.

Call the **JadeWebServiceProvider** or **WebSession** class **deleteVirtualDirectoryFile** method to delete virtual directory files (that is, the directory specified by the value of the **VirtualDirectory** parameter in the **jadehttp.ini** file or the **PhysicalDirectory** configuration directive in the Jade **mod_ jadehttp** module).

By default, Jade initiates a separate thread to clean up files in the virtual directory when **JadeHttp** is started up if the **Firewall** parameter in the [Jadehttp Files] section of the JadeHttp initialization file or the **Firewall** configuration directive in the Jade **mod_ jadehttp** module is set to **true**. Every hour this thread removes any standard files of type .**jpg**, .**png**, or .**gif** that are more than 12 hours old, by default. (As the **JadeHttp** library has no life of its own until a message is received, this thread is initiated only after the first web browser message is received.)

Alternatively, set the value of the **PurgeDirectoryRule** parameter in the [*application-name*] section of the **jadehttp.ini** file or the **PurgeDirectoryRule** configuration directive in the Jade **mod_ jadehttp** module to **AllWritable** if you want to purge all files in the virtual directory that are not read-only. The default value of **Default** for this parameter removes any standard files of type .**jpg**, .**png**, or .**gif** that are more than 12 hours old. If you want to specify the length of time since a read-only file was last modified before it is purged from the defined virtual directory for that application, specify the required value in the **PurgeFileAge** parameter.

Controlling the Location of Files Uploaded via a Web Application

If your Jade application accepts file input in text boxes on a web page (by using the **webInputType** property of a **TextBox** control with the **Web_InputType_File** setting to upload a file from a Web session), you can use the **FileTransferDirectory** parameter in the [Jadehttp Files] section of the **jadehttp.ini** file to specify the directory to which the file is written.

This parameter controls the directory in which any files transferred using the HTML **InputType=file** option are placed. (This parameter applies only when the **Firewall** parameter in the [Jadehttp Files] section of the **jadehttp.ini** file or the **Firewall** configuration directive in the Jade **mod_ jadehttp** module is set to **false**.) By default, any transferred files are placed in the same directory as the **jadehttp** library file or the Jade **mod_ jadehttp** module.

The format of the **text** property value of the **TextBox** control is as follows.

```
<source-file-name>;<destination-file-path><destination-file-name>
```

The *source-file-name* value is the name (excluding the path) of the originating file on the client workstation from which the file was loaded (that is, the workstation that is running the web browser). A semicolon character (**;**) separates this and the *destination-file-path* and *destination-file-name* values, which are the full path to which the file is written (uploaded) and the name of that file; for example:

```
UsefulStuff.doc;d:\jade\bin\txf188.tmp
```

In this example, accessing the **txf188.tmp** file in the specified directory opens a document file that contains the information in the **UsefulStuff.doc** file uploaded via the web browser.

To provide increased security for applications running in HTML thin client mode, a text file input by using the **TextBox** class **webInputType** property with a value of **Web_InputType_File** must be processed in the event that resulted in the file upload occurring (for example, in the **click** event method of a **Completed** button). In addition, set the value of the **Trace** parameter in the [Jadehttp Logging] section of the **jadehttp.ini** file to **false** to suppress logging completely. When the value of this parameter is set to **true**, logged messages acknowledge only that a message has been received or sent, because it is not possible to distinguish what is sensitive data and what is not. Setting this parameter to **true** does *not* include any of the text sent or received from the client, as this text could contain personal information, passwords, credit card details, and so on.

Caution To prevent malicious use of files uploaded to web-enabled applications, the files are removed as soon as the event that resulted in their upload has completed. You should therefore process the file immediately or move it into a directory that is not available from the web if you require that file for future processing.

[*application-name*] Section

To ensure that an application specified in a web browser cannot cause an attachment to a non-Jade environment, you must specify the name of the web application to which users can connect, by defining a unique [*application-name*] section for each web-enabled Jade application.

IIS attempts to attach only to the applications whose names are specified as section names in the **jadehttp.ini** file.

Note The **jadehttp** library module can handle connections for up to 50 different hosts (that is, you can define up to 50 unique **TcpPort[*n*]**, **MinInUse[*n*]**, **MaxInUse[*n*]**, **TcpConnection[*n*]**, **CloseDelay[*n*]**, and **ConnectionGroup[*n*]** parameters within each [*application-name*] section).

The [*application-name*] section of the **jadehttp.ini** file contains parameters that define TCP/IP connections that are used instead of the default named pipe connections. For more details and the application requirements for TCP/IP connections, see "Connecting to Jade Applications from Internet Information Server (IIS)", earlier in this chapter.

To specify that TCP/IP connections are used to connect to Jade release applications from IIS, define a section with a unique name within the **jadehttp.ini** file for each application, with the **ApplicationType** parameter and unique multiple **TcpPort[*n*]**, **MinInUse[*n*]**, **MaxInUse[*n*]**, **TcpConnection[*n*]**, **CloseDelay[*n*]**, and **ConnectionGroup[*n*]** parameters within each [*application-name*] section. Set the **ApplicationType** directive in each [*application-name*] section to **RestServices**, **WebEnabledForms**, **WebServices**, or **HtmlDocuments**.

The [*application-name*] section can contain the following parameters.

ApplicationType

Value Type String

Default Not specified

Purpose

The **ApplicationType** parameter, which is required in all [*application-name*] sections of the **jadehttp.ini** file, specifies the type of application that will be supported and how the interfaces to Jade function. The valid parameter values are listed in the following table.

Parameter Value	Description
RestServices	Representational State Transfer (REST) lightweight web service that embraces HyperText Transfer Protocol (HTTP) through Extensible Markup Language (XML) or JavaScript Object Notation (JSON) messages instead of SOAP.
WebEnabledForms	Web-enabled forms. This application type results in all requests from a user being directed to the same Jade application copy during the user's session. **JadeHttp** no longer inserts the tags into the HTML output; the Jade software now performs this function. The Jade side uses the multi-worker TCP facility to perform routing and queuing.
WebServices	Web service operation, where user requests are sent to any available connection. The Jade side uses the multi-worker TCP facility to perform routing and queuing.
HtmlDocuments	HTML documents operation where user requests are sent to any available connection. The Jade side uses the multi-worker TCP facility to perform routing and queuing.

CloseDelay[*n*]

Value Type Integer *(number-of-seconds)*

Default Not specified

Purpose

When the current number of in-use connections for the TCP/IP specified in the **TcpPort[*n*]** parameter exceeds the value of the **MinInUse[*n*]** parameter, the **CloseDelay[*n*]** parameter indicates how many seconds before the extra connections are closed after they become inactive.

This value is used only when the value of the **ApplicationType** parameter is set to **RestServices**, **WebEnabledForms**, **WebServices**, or **HtmlDocuments**.

ConnectionGroup[*n*]

Value Type String

Default Not specified

Purpose

The **ConnectionGroup[*n*]** parameter specifies the name of the connection group associated with the value of the **TcpPort[*n*]** parameter (that is, the TCP/IP port to be used for the connection). The connection group is the unique name that is used by applications of type **WebEnabledForms** to identify the TCP/IP port (the value of the **TcpPort[*n*]** parameter) to which a user request is to be sent.

Note

If you do not specify this parameter with a unique connection group name, this process does not function correctly.

MaxInUse[*n*]

Value Type Integer

Default Not specified

Purpose

The **MaxInUse[*n*]** parameter specifies the maximum number of connections that can be opened for the corresponding value of the **TcpPort[*n*]** parameter only when the value of the **ApplicationType** parameter is set to **RestServices**, **WebEnabledForms**, **WebServices**, or **HtmlDocuments**.

The **MaxInUse** parameter with no specified suffix is the limit for a **ConnectionGroup** associated with an entry with the **TcpConnection** parameter; it is *not* a global limit.

The **MaxInUse[*n*]** parameter defines the limit of outstanding requests for a single connection group; that is, the sum of queued and active requests. The total number of outstanding requests for a web application is the sum of the **MaxInUse[*n*]** suffix values for all of the connection groups specified for that application.

Note This parameter is the only **jadehttp.dll** configuration setting that affects the dynamic worker pool scaling. If this value is too small, it will prevent the spawning of new REST service applications. The value must be large enough to allow enough connections between IIS and Jade so the queue depth remains greater than the **QueueDepthLimit** parameter value for the period specified in the **QueueDepthLimitTimeout** parameter defined in the [WebOptions] section of the **jade.ini** file.

In general, most REST server usage patterns would be well-served by a **MaxInUse** parameter value that is 10 times larger than the queue depth limit, with a value not less than **50**.

The **MaxInUse** parameter is the only configuration setting the affects how often the system will respond with a “Server Unavailable” 503 response. In a direct HTTP configuration, all requests are queued. This parameter controls how many connections (sockets) to keep open when they are not in use. Given the speed at which sockets can be created, this value has no real effect on system performance, and it is entirely reasonable to set it its minimum value of **1**.

For more details, see "Scaling Dynamic Worker Pool Connections", in Chapter 2 of the *Installation and Configuration Guide*.

Note

The value of the **MaxInUse** parameter is forced to be equal to or greater than the value of the **MinInUse** parameter.

MaxMessageSize

Value Type Integer

Default 1000000 bytes

Purpose

The **MaxMessageSize** parameter specifies the maximum size allowed for a single-part web message (that is, a single stream of data with minimal formatting). If the size of a single-part message exceeds the specified value of this parameter, the request is rejected and the following message is displayed to the user.

```
The size of the input data exceeds the limit permitted for this application and
has been rejected.
```

Note

Multiple-part formatted web messages are not limited in size (each message is made up of a series of sections that are specifically formatted).

MessageTimeout

Value Type Integer *(seconds)*

Default 300

Purpose

The **MessageTimeout** parameter specifies the maximum number of seconds that the **JadeHttp.dll** library waits for a reply from a Jade system before sending a failure message to the requesting web browser.

Hints

This parameter enables you to test the effects of unexpectedly long request processing without having to wait the default five minutes for each test.

When web-enabled applications are all busy, you can control the time that a web browser window shows no action before returning the *Service unavailable* response.

MinInUse[*n*]

Value Type Integer

Default Not specified

Purpose

The **MinInUse[*n*]** parameter specifies the minimum number of connections that can be opened for the corresponding value of the **TcpPort[*n*]** parameter only when the value of the **ApplicationType** parameter is set to **RestServices**, **WebEnabledForms**, **WebServices**, or **HtmlDocuments**.

The default minimum number of connections is **1**.

MinMessageSize

Value Type Integer

Default 10 bytes

Purpose

The **MinMessageSize** parameter specifies the minimum size allowed for a web message received from Jade using the **WebSession** class **reply** method to send HTML string web requests back to the client node.

The minimum value is **1** byte, the maximum value **1024** bytes, and the default value **10** bytes.

This value is read the first time the application specified in the [*application-name*] section is accessed after **jadehttp.dll** has been loaded by Internet Information Server (IIS).

PurgeDirectoryRule

Value Type String

Default Default

Purpose

The **PurgeDirectoryRule** parameter specifies when files that are not read-only in the virtual directory are purged. The default value of **Default** for this parameter removes any standard files of type .**jpg**, .**png**, or .**gif** that are more than 12 hours old. Specify a value of **AllWritable** if you want to purge all files in the virtual directory that are not read-only.

PurgeFileAge

Value Type *time-unit[D|H|M]*

Default 12H

Purpose

The **PurgeFileAge** parameter specifies the length of time since a read-only file was last modified before it is purged from the defined virtual directory for that application; for example, **PurgeFileAge=7M**.

The parameter value is a time-unit numeric, optionally followed by a modifier **D** (days), **H** (hours), or **M** (minutes). The **D** multiplier multiplies by 86,400; the **H** multiplier multiplies by 3,600; the **M** multiplier multiplies by 60; and all other time-unit values are treated as seconds. Specify zero (**0**) if you want to turn off the purging of the physical directory.

The minimum time-unit value that you can specify is **30** seconds; the maximum value is **365D** (days). Values outside this range are forced to their respective limits.

This parameter applies only if the **VirtualDirectory** parameter has a specified value, and only files that match the **PurgeDirectoryRule** parameter criteria are considered (that is, all non-read-only files or by default, standard files of type .**jpg**, .**png**, or .**gif** that are more than the specified age).

TcpConnection[*n*]

Value Type Integer *(tcp-address)*

Default Not specified

Purpose

The **TcpConnection[*n*]** parameter specifies the unique valid TCP/IP address of the port to connect to on the server (remote) node, to enable web client nodes to connect to the specified server node across the network through a TCP/IP connection.

TcpPort[*n*]

Value Type Integer *(tcp-port-number)*

Default Not specified

Purpose

The **TcpPort[*n*]** parameter specifies the TCP/IP port number to be used for the connection specified in the corresponding **TcpConnection[*n*]** parameter for applications of type **WebEnabledForms**, **WebServices**, and **HtmlDocuments**.

UseNewStyleMultipart

Value Type Boolean

Default False

Purpose

The **UseNewStyleMultipart** parameter specifies whether Jade overrides the previous behavior of automatically converting the **multipart/form-data** body into **url/form-encoded**, where each bit of form data is separated by an ampersand character (**&**) and the boundaries, **Content-Disposition**, and **Content-Type** information are removed. In addition, by default data is not sent through to the Jade REST method as parameters, so it relies on user code to extract it from the body (for example, by reimplementing the **JadeRestService** class **processRequest** method).

Set this parameter to **true** if you want to override the default handling when the **multipart/form-data** content type is used for a **POST** request, which is typically used when uploading the contents of files that may be very large and have special characters or be in binary format. Jade leaves the HTTP body as it is, and instead extracts the data from the parts in the body, passing them as parameters to the Jade REST method. For details about multipart/form-data encoding, see "Multipart Form Data Encoding Server-Side Support", in Chapter 11 of the *Developer's Reference*.

Applies to Version: 2022.0.01 and higher

VirtualDirectory

Value Type String *(virtual-directory-name)*

Default Not specified

Purpose

The **VirtualDirectory** parameter enables you to specify a string containing the name of the virtual directory for the HTML-enabled application.

Hints

The value specified for this parameter is used if the **fileName** parameter value of the **JadeWebServiceProvider** or **WebSession** class **createVirtualDirectoryFile** method does not include a directory. If neither this parameter nor the file name in the **createVirtualDirectoryFile** method contains a directory, the files are written into the same directory as the **jadehttp** library module.

This parameter enables you to control virtual directory permissions so that the directory containing the **jadehttp** library module does not need to have read and write access.

For details about retaining files passed to the **jadehttp** library, see "Firewall for the Jade Internet Environment", earlier in this chapter. For details about returning whether a specified file is present on the IIS side of the firewall when using the Jade HTML thin client interface via the **jadehttp** library file, see "Image Files Created by the Jade Application", earlier in this chapter.

Sample [*application-name*] Section

The following are examples of an [*application-name*] section in the **jadehttp.ini** file for application types specified by the **ApplicationType** parameter.

```
[Company]
ApplicationType = WebEnabledForms
```

```
TcpConnection   = Host1a
TcpPort         = 20000
ConnectionGroup = CompanyForms
MinInUse        = 1
MaxInUse        = 5
CloseDelay      = 600

[RestCo]
ApplicationType = RestServices
TcpConnection   = Host1a
TcpPort         = 27000
MinInUse        = 5
MaxInUse        = 20
CloseDelay      = 600

[WebCompany]
ApplicationType = WebServices
TcpConnection   = Host1a
TcpPort         = 21000
MinInUse        = 5
MaxInUse        = 15
CloseDelay      = 600
TcpConnection2  = Host2a
TcpPort2        = 21001
MinInUse2       = 5
MaxInUse2       = 15
CloseDelay2     = 600

[HtmlCompany]
ApplicationType = HtmlDocuments
TcpConnection   = Host1a
TcpPort         = 22000
MinInUse        = 2
MaxInUse        = 10
CloseDelay      = 600
TcpConnection2  = Host2a
TcpPort2        = 22001
MinInUse2       = 2
MaxInUse2       = 10
CloseDelay2     = 600
```

[Jadehttp Files] Section

The [Jadehttp Files] section of the **jadehttp.ini** file contains parameters that control firewall security and the location of files uploaded from a web-enabled application.

Caution To prevent malicious use of files uploaded to web-enabled applications, the files are removed as soon as the event that resulted in their upload has completed. You should therefore process the file immediately or move it into a directory that is not available from the web if you require that file for future processing.

The **jadehttp** library derives a path for the **jadehttp.ini**, **jadehttp.log**, and transfer files, ensuring that they are completely secure. The directories are derived from the **jadehttp** library directory for these files, as follows.

1. **JadeHttp** attempts to create the required directories if they do not exist if security permits this. These directories, based on the example **c:\jade\bin\jadehttp.dll** file, are listed in the following table.

Directory	Description	Example
dll-path_dll-name\ini\	For the initialization file and the default error ***dll-name*.htm** file	c:\jade\bin_jadehttp\ini\jadehttp.ini
dll-path_dll-name\logs\	For all *dll-name* logs	c:\jade\bin_ jadehttp\logs\jadehttp.log
dll-path_dll-name\transfer\	For the location of any temporary files created during file transfer	c:\jade\bin_jadehttp\transfer\...

2. If the initialization file option to specify the log path is set, the logs directory is not created or used.
3. If the initialization file option to specify the transfer path is set, the transfer directory is not created or used.
4. As the dynamic link library can be renamed, its name is included. The library path therefore does not need to contain any other files, and users cannot access any of the files listed in the previous table.

For details about implementing a TCP/IP connection from the IIS to Jade applications, see "Connecting to Jade Applications from Internet Information Server (IIS)" and "Configuring JadeHttp for Remote Connections", earlier in this chapter.

The [Jadehttp Files] section can contain the following parameters.

FileTransferDirectory

Value Type String *(disk-path)*

Default Not specified

Purpose

The **FileTransferDirectory** parameter specifies the directory to which files are uploaded during web sessions from Jade applications that accept file input in text boxes on web pages (by using the **webInputType** property of a **TextBox** control with the **Web_InputType_File** setting to upload a file from a web session).

Note This parameter applies only when the **Firewall** parameter in the [Jadehttp Files] section of the **jadehttp.ini** file is set to **false** or it is set to **true** and the **FileTransferDirectory** parameter is set to **default** or to a valid directory. (For details, see "Configuring JadeHttp for Remote Connections", earlier in this section.)

This parameter controls the directory in which any files transferred by using the HTML **InputType=file** option are placed. (The file transfer directory must be a valid directory that is relative to the virtual directory, or IIS server.)

If you define a value for the **FileTransferDirectory** parameter, the transfer directory in the same directory as the **jadehttp** library is neither created nor used. The file transfer directory path cannot be greater than 260 characters. For more details, including the format of the **text** property value of the **TextBox** control, see "Controlling the Location of Files Uploaded via a Web Application", earlier in this chapter.

Firewall

Value Type Boolean

Default False

Purpose

The **Firewall** parameter controls whether firewall separation is used when transferring files to Jade from web browsers. When this parameter is set to the default value of **false**, files are written to the transfer directory directly by the **JadeHttp** library.

The firewall must be enabled at both ends of the connection (that is, if you are uploading images from another machine for a Jade web-enabled form over a TCP/IP connection, the **Firewall** parameter in the [Jadehttp Files] section of the JadeHttp initialization file and the **Firewall** parameter in the [WebOptions] section of the Jade initialization file must both be set to **true**).

For details about the handling of data from the user's browser through to Jade, see "Configuring JadeHttp for Remote Connections", earlier in this section.

Set this parameter to **true** if you want files transferred to the Jade application over the TCP/IP connection. Each file transferred from a web browser is sent to the Jade application as a separate message. Your Jade application must distinguish this message from a normal web browser request and store the transferred file. For more details, including the format of messages sent through the firewall, see "Firewall for the Jade Internet Environment", earlier in this chapter.

Sample [Jadehttp Files] Section

The following is an example of a [Jadehttp Files] section in the **jadehttp.ini** file.

```
[Jadehttp Files]
FileTransferDirectory = c:\jade\bin_jadehttp\transfer
Firewall              = true
```

[Jadehttp Logging] Section

The [Jadehttp Logging] section of the **jadehttp.ini** file controls logging by the **jadehttp** library file. For details about the messages that are logged when tracing is set, see "Message Logging", in the *Web Application Guide*.

The [Jadehttp Logging] section can contain the following parameters.

Trace

Value Type Boolean

Default False

Purpose

The **Trace** parameter controls **jadehttp** library file logging of message meta data. When this parameter is set to the default value of **false**, logging is suppressed completely.

When the value of this parameter is set to **true**, logged messages acknowledge only that a message has been received or sent, because it is not possible to distinguish what is sensitive data and what is not. Setting this parameter to **true** does *not* include any of the text sent or received from the client, as this text could contain personal information, passwords, credit card details, and so on.

Note You cannot use this parameter to inspect and debug data passing through the **jadehttp** library.

TraceFile

Value Type String *(file-path)*

Default Default

Purpose

The **TraceFile** parameter specifies an optional path and a file name that overrides the default **jadehttp.log** file name so that log files can be placed in other directories.

If you define a value for the **TraceFile** parameter, the log file in the same directory as the **jadehttp** library is neither created nor used. The file transfer directory path cannot be greater than 260 characters.

TraceFileSize

Value Type Integer (bytes)

Default 1000000

Purpose

The **TraceFileSize** parameter specifies the maximum file size before switching logs.

Sample [Jadehttp Logging] Section

The following is an example of a [Jadehttp Logging] section in the **jadehttp.ini** file.

```
[Jadehttp Logging]
Trace     = true
TraceFile = c:\jade\bin_jadehttp\logs\jadehttp.log
```

Configuring Apache for Remote Connections

TCP/IP connections only are used for connections to Jade applications running from the Apache HTTP Server. This section covers the following topics.

- Apache Configuration Directives
- Apache Configuration Examples
- Apache Considerations

For details about installing the **mod_ jadehttp** library module supplied by Jade for the Apache HTTP Server and the MPMs that it implements, see "Connecting to Jade Applications from an Apache HTTP Server", earlier in this chapter.

Apache Configuration Directives

The configuration directives in this section are available to configure the Jade **mod_ jadehttp** module. It uses the standard Apache configuration syntax, rather than implementing configuration via an initialization file.

Set the value of the **JadeHttp_Trace** directive to **true** to ensure the security of your data.

The concept and intent of these directives is the same as the **jadehttp.ini** file parameters, but the syntax differs. For details about the **jadehttp.ini** file parameters, see "Configuring JadeHttp for Remote Connections", earlier in this chapter.

Most of the directives specific to **mod_ jadehttp** have the same meaning as their IIS counterparts (described in "Controlling the Location of Files Uploaded via a Web Application" under "Configuring JadeHttp for Remote Connections", earlier in this chapter). The Jade **mod_ jadehttp** module configuration directives are as follows.

- SetHandler
- ApplicationType
- Application
- FaultDocument

- FileTransferDirectory
- Firewall
- JadeHttp_Trace
- JadeServer
- JadeTimeout
- LocalInterface
- MaxMessageSize
- MinMessageSize
- PhysicalDirectory
- PurgeDirectoryFrequency
- PurgeDirectoryRule
- PurgeFileAge
- TcpConnection[*n*]
- VirtualDirectory

For details, see the following subsections.

To specify that TCP/IP connections are used to connect to Jade applications from Apache, you must define a **<Location>** directive within **jadehttp.conf** for each application, with an **ApplicationType** directive and unique multiple **TcpConnection[*n*]** directives between each **Application** directive.

The following **mod_ jadehttp** directives apply between the core Apache **<Location>** and **</Location>** directives. Apache processes the **<Location>** directives in the order in which they are found in the configuration file, following any **<Files>** and **<Directory>** sections and after .**htaccess** files have been read and processed.

```
<Location /jade>
    SetHandler jadehttp-handler
    ApplicationType WebServices
    Application WebApplication
    TcpConnection PC1 21000 5 15 300
</Location>
```

This allows **mod_ jadehttp** directives to be placed in higher locations and their values to become the defaults for lower locations unless you explicitly override them.

In the following example, the handler and **FaultDocument** directive specified in **<Location /jade>** also apply to **<Location /jade/subloc>**.

```
<Location /jade>
    SetHandler jadehttp-handler
    FaultDocument PostTooLarge "http://server/ohdarn.html"
</Location>
<Location /jade/subloc>
    ApplicationType WebEnabledForms
```

```
    Application WebApplication
    TcpConnection PC1 21000 5 10 200
</Location>
```

SetHandler

The standard **SetHandler** Apache directive, whose characteristics are listed in the following table, selects the **mod_ jadehttp** module to handle this location.

Characteristic	Value
Action	Forces all matching files to be processed by a handler
Syntax	SetHandler *handler-name*
Context	server config, virtual host, directory, .htaccess
Module	Core
Examples	SetHandler jadehttp-info and SetHandler jadehttp-handler

The **jadehttp-handler**, which is the primary handler inside the **mod_ jadehttp** module, formats and passes data to Jade and returns the results to the web browser. The **jadehttp-info** handler reports internal information about the configuration and current status of the **mod_ jadehttp** module back to the web browser. This handler is similar to the standard Apache **mod_info** module that provides a comprehensive overview of the server configuration. Access to locations that use this handler should be restricted to trusted sites.

ApplicationType

The **ApplicationType** directive specifies the type of application that will be supported and how the interfaces to Jade function.

Note You must define this directive *before* the **TcpConnection[*n*]** directive within each **Application** directive.

The application type characteristics are listed in the following table.

Characteristic	Value
Action	Specifies the Jade application type
Syntax	ApplicationType *application-type*
Context	Location
Handler	jadehttp-handler
Module	mod_ jadehttp
Example	ApplicationType RestServices

The valid application type values are listed in the following table.

Value	Description
RestServices	Representational State Transfer (REST) lightweight web service that embraces HyperText Transfer Protocol (HTTP) through Extensible Markup Language (XML) or JavaScript Object Notation (JSON) messages instead of SOAP.

Value	Description
WebEnabledForms	Web-enabled forms. This application type results in all requests from a user being directed to the same Jade application copy during the user's session. **JadeHttp** no longer inserts the tags into the HTML output; the Jade software now performs this function. The Jade side uses the multi-worker TCP facility to perform routing and queuing. **Note** For the **WebEnabledForms** application type, you must specify the **MinInUse**, **MaxInUse**, **CloseDelay**, and **ConnectionGroup** values for the **TcpConnection[*n*]** directive.
WebServices	Web service operation, where user requests are sent to any available connection. The Jade side uses the multi-worker TCP facility to perform routing and queuing.
HtmlDocuments	HTML documents operation where user requests are sent to any available connection. The Jade side uses the multi-worker TCP facility to perform routing and queuing.

Application

The **Application** directive, whose characteristics are listed in the following table, specifies the name of the web-enabled Jade application.

Characteristic	Value
Action	Specifies the Jade application to use
Syntax	Application *application-name*
Context	Location
Handler	jadehttp-handler
Module	mod_ jadehttp
Example	Application ErewhonShop

This directive is similar to the functionality of the [*application-name*] section of the **jadehttp** library module for IIS, described earlier in this chapter.

FaultDocument

The **FaultDocument** directive, whose characteristics are listed in the following table, specifies what the **mod_ jadehttp** module does when it encounters an error.

Characteristic	Value
Action	Specifies the URL or path name for redirection when an error occurs
Syntax	FaultDocument *cause redirect-url*
Context	Location
Handler	jadehttp-handler, jadehttp-info
Module	mod_jadehttp
Example 1	FaultDocument TcpConnectFailed http://secondary.example.com:80/page.html
Example 2	FaultDocument PostTooLarge bigpost.html

You can specify the *cause* value in numeric or text form. Valid values for the *cause* are listed in the following table.

Numeric Name	Text Name
600	TcpConnectFailed
601	PostTooLarge
602	ServiceFailed
603	DataReadFailed

You must specify a valid URL or path name for the *redirect-url* value, which can be on the local web server or on a remote web server. If you do not specify a *redirect-url* value and an error occurs, an error page is raised internally in the **mod_ jadehttp** module.

FileTransferDirectory

The **FileTransferDirectory** directive, whose characteristics are listed in the following table, specifies the name of a local directory to use when files are uploaded from the web browser to the Jade application.

Characteristic	Value
Action	Specifies the directory to use when files are uploaded from the web browser
Syntax	FileTransferDirectory *local-directory-name*
Context	Location
Handler	jadehttp-handler
Module	mod_jadehttp
Example	FileTransferDirectory /var/tmp/appname

The **FileTransferDirectory** directive is similar to the **FileTransferDirectory** parameter in the [Jadehttp Files] section of the **jadehttp.ini** file for IIS servers, described earlier in this chapter.

As the **mod_ jadehttp** module must put the transfer files somewhere before transferring them to Jade, this directive specifies the directory to which the file or files are written.

Firewall

The **Firewall** directive, whose characteristics are listed in the following table, specifies whether there is a firewall between the **mod_ jadehttp** module and the Jade application.

Characteristic	Value
Action	Indicates if there is a firewall between **mod_jadehttp** and Jade
Syntax	Firewall on|off
Context	Location
Handler	jadehttp-handler
Module	mod_jadehttp
Example	Firewall on

The **Firewall** directive is similar to the **Firewall** parameter in the [Jadehttp Files] section of the **jadehttp.ini** file for IIS servers, described earlier in this chapter. If a firewall is present, file transfers between Jade and the web server must be done by using the **mod_ jadehttp** module.

The firewall must be enabled at both ends of the connection (that is, if you are uploading images from another machine for a Jade web-enabled form over a TCP/IP connection, the **Firewall** configuration directive in the Jade **mod_ jadehttp** module must be set to **on** and the **Firewall** parameter in the [WebOptions] section of the Jade initialization file must be set to **true**).

When transferred, the files will be placed in the directory specified in the **PhysicalDirectory** directive.

JadeHttp_Trace

The **JadeHttp_Trace** directive, whose characteristics are listed in the following table, controls where any **mod_ jadehttp** tracing is written.

Characteristic	Value
Action	Specifies the log file to which Jade trace messages are sent
Syntax	JadeHttp_Trace *bit-mask* [*file-name [size[K|M]]]*
	JadeHttp_Trace *bit-mask* "| *rotatelogs-program rotatelogs-arguments*"
Context	Server config, virtual host, Location
Module	mod_jadehttp
Examples	JadeHttp_Trace 0x000000ff logs/jade%Y%m%d%H%M%S.log 10M
	JadeHttp_Trace 0x000000ff "|/usr/sbin/rotatelogs2 -l /var/logs/apache2/jadehttp_%Y%m%d%H%M.log 10M"

The **JadeHttp_Trace** directive is similar to the **Trace**, **TraceFile**, and [**TraceFileSize**] parameters in the [Jadehttp Logging] section of the **jadehttp.ini** file, documented earlier in this chapter.

If this parameter is not specified, logging is suppressed completely. Specifying this parameter ensures the security of your data. When the value of this parameter is specified, logged messages acknowledge only that a message has been received or sent, because it is not possible to distinguish what is sensitive data and what is not. When this parameter is specified, messages logged to the **jadehttp.log** file do not include any of the text sent or received from the client, as this text could contain personal information, passwords, credit card details, and so on.

Note You cannot use this parameter to inspect and debug data passing through the **jadehttp** library.

If the *file-name* is relative, it is created relative to the **ServerRoot** location. If you do not specify the *file-name*, it defaults to **"logs/jadehttp%Y%m%d%H%M%S.log"**.

If you do not specify the *size*, it defaults to **5M** bytes. To disable **mod_ jadehttp** from attempting to rotate log files, specify a *size* of zero (**0**). The *size* value can have a multiplier appended to it. The **K** multiplier multiplies by **1,024**, the **M** multiplier multiplies by **1,048,576** bytes, and all other values are treated as bytes.

If the *file-name* is in a directory that an Apache child **httpd** process does not have permission to access, the rotation of logs is disabled in **mod_ jadehttp**. The file name can include **%** modifiers, which specify values from the current local time (based on the Apache **rotatelogs2** program, whose documentation is available on the Oracle website, for example), permitted values are **AaBbcdHIjMmSUWwXxYyZ%**.2.4.10

If no **%** modifiers are present, **"%Y%m%d%H%M%S"** is appended at the end of the file name (before the file extension, if it exists). This adds year, month, day, hour, minute, and second values to the file name.

A new log file is created if the file name changes (due to the changing time and selected modifier values) and if the file exceeds the specified size.

Note Multiple log files could be actively written to concurrently, because of the way that the Apache pre-fork MPM works.

Alternatively, you can use an external program to get log files rotated; for example, the one that is normally provided with the **rotatelogs2** Apache distribution. This works well when log files are placed in a directory where the Apache child process does not have permission to create files.

The use of an external program behaves the same way as the Apache piped logs feature, which is described in the Apache documentation. It involves specifying the rotate program and arguments inside double quote characters (**""**), with the first character of the string being the pipe (**|**) character.

Note The name of the **rotatelogs** program can differ, depending on your Apache distribution and operating system.

The bit set values for the *bit-mask* value are listed in the following table.

Bit Set	Description
0x00000001	When a redirect occurs
0x00000002	When a special command is issued
0x00000004	When file purges occur
0x00000008	When **mod_ jadehttp** sends connection information to Jade

JadeServer

The **JadeServer** directive, whose characteristics are listed in the following table, is reserved for future implementation.

Characteristic	Value
Action	Specifies the Jade server configuration
Syntax	JadeServer *jade-version byte-order character-size*
Context	Location
Handler	jadehttp-handler
Module	mod_jadehttp
Default	JadeServer 7.1 *localEndian* oneByte
Example	JadeServer 7 Intel utf-16

This directive allows a single **mod_ jadehttp** module to connect to Jade, regardless of the combination of web server architecture and Jade server architecture, allowing **mod_ jadehttp** to adapt to the installed release of the Jade server.

The *jade-version* value allows for the protocol to between Jade and the **mod_ jadehttp** module to be changed. The version number can be in the range 1 through 4, with each number separated with numbers inclusive separated with the dot operator (**.**) notation; for example, **7.1.05.007**.

The *byte-order* value allows for binary numbers and wide characters to be converted to the native byte ordering of the Jade server, which may be different from the byte ordering of the web server.

In the default example in the previous table, the ***localEndian*** value defaults to **littleEndian** when the **mod_ jadehttp** module is running on Intel-based hosts and **bigEndian** when on PowerPC-based hosts. The entered value is not case-sensitive.

The primary names and synonyms for the *byte-order* value are listed in the following table.

Primary Name	Synonym
littleEndian	Intel
bigEndian	PowerPC

The *character-size* value indicates the size of a Jade character, which is ANSI or Unicode, depending on your Jade installation. If the character size is Unicode where the Jade server is running, the physical size of a Jade character can be one of 1, 2, or 4 bytes in size. This allows **mod_ jadehttp** to send Jade characters of the correct size and type to the Jade server.

The primary names and synonyms for the entered value, which is not case-sensitive, are listed in the following table.

Primary Name	Synonym
oneByte	ANSI
twoByte	Unicode, Utf16, or Utf-16
fourByte	Utf32 or Utf-32

JadeTimeout

The **JadeTimeout** directive, whose characteristics are listed in the following table, specifies the maximum number of seconds that **mod_ jadehttp** waits for a reply from a Jade system before sending a failure message to the requesting web browser.

Characteristic	Value
Action	Specifies the maximum number of seconds to wait for a reply from Jade
Syntax	JadeTimeout *number-of-seconds* (the default value is 45 seconds)
Context	Location
Handler	jadehttp-handler
Module	mod_jadehttp
Default	JadeTimeout 300 (that is, 5 minutes)
Example	JadeTimeout 30

A default value of zero (**0**) indicates that there is not timeout. The maximum timeout value is **1800** (that is, 30 minutes).

Hints

This directive enables you to test the effects of unexpectedly long request processing without having to wait the default five minutes for each test.

When web-enabled applications are all busy, you can control the time that a web browser window shows no action before returning the *Service unavailable* response.

LocalInterface

The **LocalInterface** directive, whose characteristics are listed in the following table, specifies the local network interface and optionally the port number to use when connecting from the **mod_ jadehttp** module to the Jade web-enabled application.

Characteristic	Value
Action	Specifies the local network interface and optionally the port number
Syntax	LocalInterface *local-network-interface-handler-name*
Context	Location
Handler	jadehttp-handler
Module	mod_jadehttp
Default	LocalInterface 0.0.0.0
Example	LocalInterface 10.1.1.1

If your web server host has multiple network interface cards, you can specify which one **mod_jadehttp** uses when initiating connections to the web-enabled Jade application.

The default value of **0.0.0.0** allows the operating system to choose the network interface card.

You would normally require this directive only when there are specific network security or routing issues to be addressed.

Caution Although you can also specify the local outgoing port number, you should take great care when doing so and only when you have a specific requirement. Specifying a **LocalInterface** port number limits you to a maximum of one TCP/IP connection to Jade for each Apache **<Location>** directive. In addition, if the web server terminates abnormally and the port is not properly closed, it may take several minutes before this port becomes available again for use.

MaxMessageSize

The **MaxMessageSize** directive, whose characteristics are listed in the following table, specifies the largest size of a **POST** URL before an error is raised.

Characteristic	Value
Action	Specifies the maximum size of a **POST** string
Syntax	MaxMessageSize *size[M|K]*
Context	Location
Handler	jadehttp-handler
Module	mod_jadehttp
Default	MaxMessageSize 1M
Example	MaxMessageSize 50K

You can append a multiplier to the *size* value. The **M** multiplier multiplies by **1,048,756**, the **K** multiplier multiplies by **1,024**, and all other values are treated as bytes.

The minimum value is zero (**0**) and the maximum value is **128M**.

MinMessageSize

The **MinMessageSize** directive, whose characteristics are listed in the following table, specifies the minimum size allowed for a web message received from Jade using the **WebSession** class **reply** method to send HTML string web requests back to the client node.

Characteristic	Value
Action	Specifies the minimum number of bytes allowed for a web message from Jade
Syntax	MinMessageSize *message-size-in-bytes*
Context	Location
Handler	jadehttp-handler
Module	mod_jadehttp
Default	MinMessageSize 10
Example	MinMessageSize 100

The minimum value is **1** byte and the maximum value is **1024** bytes. This value is read once, when the Apache web server starts.

PhysicalDirectory

The **PhysicalDirectory** directive specifies the name of a local directory to use when files are transferred from Jade to the web server by using the **JadeWebServiceProvider** or **WebSession** class **createVirtualDirectoryFile** method without specifying a directory in the **fileName** parameter. This directory is also used as the location for transferring files when the **Firewall** directive is set to on. In this situation, it behaves in a similar way to the **FileTransferDirectory** parameter in the [Jadehttp Files] section of the jadehttp.ini file for IIS servers, described earlier in this chapter.

The characteristics of the **PhysicalDirectory** directive are listed in the following table.

Characteristic	Value
Action	Specifies the directory to use as a physical directory
Syntax	PhysicalDirectory *local-directory*
Context	Location
Handler	jadehttp-handler
Module	mod_jadehttp
Example	PhysicalDirectory /var/tmp/appname/

Note Apache directives require separators to be defined using a forward slash (*/*) character.

The following is an example of the use of the **PhysicalDirectory** directive.

```
<Directory /var/spool/www/jade/images/>
    order allow,deny
    allow from all
</Directory>

Alias /jade/images/ /var/spool/www/jade/images/

<location /jade/webform>
```

```
        SetHandler jadehttp_handler

        PhysicalDirectory /var/spool/www/jade/images/

    </location>
```

In this example, the physical directory on disk **/var/spool/www/jade/images/** needs to be the same in the three places. The actual value can change on a site-specific basis.

PurgeDirectoryFrequency

The **PurgeDirectoryFrequency** directive, whose characteristics are listed in the following table, specifies how often the physical directory is checked to see if files require purging.

Characteristic	Value
Action	Specifies the frequency at which the physical directory is purged
Syntax	PurgeDirectoryFrequency *time-unit[H|M]*
Context	Location
Handler	jadehttp-handler
Module	mod_jadehttp
Default	PurgeDirectoryFrequency 1H
Example	PurgeDirectoryFrequency 5H

You can append a multiplier to the *time-unit* value. The **H** multiplier multiplies by **3,600**, the **M** multiplier multiplies by **60**, and all other *time-unit* values are treated as seconds.

Specify zero (**0**) for the *time-unit* value to turn off the purging of the physical directory.

The minimum that you can specify for the *time-unit* value is **5** minutes and the maximum is **24H** (hours).

Values outside this range are forced to their respective limits.

PurgeDirectoryRule

The **PurgeDirectoryRule** directive, whose characteristics are listed in the following table, specifies when files in the virtual directory that are not read-only are purged.

Characteristic	Value
Action	Specifies when non-read-only files in the virtual directory are purged
Syntax	PurgeDirectoryRule Default|AllWritable
Context	Location
Handler	jadehttp-handler
Module	mod_jadehttp
Default	PurgeFileAge Default
Example	PurgeFileAge AllWritable

The default value of **Default** for this directive removes any standard files of type .**jpg**, .**png**, or .**gif** that are more than 12 hours old.

Specify a value of **AllWritable** if you want to purge all files in the virtual directory that are not read-only.

PurgeFileAge

The **PurgeFileAge** directive, whose characteristics are listed in the following table, specifies the length of time since a file was last modified before it is purged from the physical directory.

Characteristic	Value
Action	Specifies the minimum modified age before the files are purged
Syntax	PurgeFileAge *time-unit[D|H|M]*
Context	Location
Handler	jadehttp-handler
Module	mod_jadehttp
Default	PurgeFileAge 12H
Example	PurgeFileAge 1D

You can append a multiplier to the *time-unit* value. The **D** multiplier multiplies by 86,400, the **H** multiplier multiplies by **3,600**, the **M** multiplier multiplies by **60**, and all other *time-unit* values are treated as seconds. Specify zero (**0**) for the *time-unit* value to turn off the purging of the physical directory.

The minimum that you can specify for the *time-unit* value is **30** seconds and the maximum is **35D** (days). Values outside this range are forced to their respective limits.

See also the **JadeWebServiceProvider** or **WebSession** class **deleteVirtualDirectoryFile** method.

TcpConnection[*n*]

The **TcpConnection[*n*]** directive can have one of two types of syntax, depending on the value of the **ApplicationType** directive. The optional **[*n*]** value can be a unique number in the range **1** through **9**. If you do not specify the optional *n* group suffix, Jade treats the value as **1**.

You can specify the *host-name* value as a host name or as an IP address. If you specify a host name, it must resolve to a single IP address. If you append the optional *n* value to the **TcpConnection[*n*]** directive, it indicates the group to which the specified host name port number range is assigned.

If the value of the **ApplicationType** directive is **RestServices**, **WebEnabledForms**, **WebServices**, or **HtmlDocuments**, the characteristics listed in the following table specify the TCP/IP connection details between the **mod_ jadehttp** module and the Jade Web application.

Characteristic	Value
Action	Specifies the TCP/IP connection to the Jade application
Syntax	TcpConnection[*n*] *host-name port* [MinInUse [MaxInUse [CloseDelay [ConnectionGroup]]]] **Note** For the **WebEnabledForms** application type, you must specify the **MinInUse**, **MaxInUse**, **CloseDelay**, and **ConnectionGroup** values.
Context	Location
Handler	jadehttp-handler

Characteristic	Value
Module	mod_jadehttp
Example 1	TcpConnection server1 6008
Example 2	TcpConnection2 10.1.1.19 51030 3 15 300
Example 3	TcpConnection3 10.1.1.19 51031 1 10 30 Forms

For details about the optional **MinInUse**, **MaxInUse**, **CloseDelay**, and **ConnectionGroup** values, see the following subsections.

MinInUse

The optional **MinInUse** value specifies the minimum number of connections that can be opened for the **TcpConnection[*n*]** directive when the value of the **ApplicationType** directive is set to **RestServices**, **WebServices**, or **HtmlDocuments**.

This value must be specified when the **ApplicationType** directive is set to **WebEnabledForms**.

The default minimum number of connections is **1**.

MaxInUse

The optional **MaxInUse** value specifies the maximum number of connections that can be opened for the **TcpConnection[*n*]** directive when the value of the **ApplicationType** directive is set to **RestServices**, **WebServices**, or **HtmlDocuments**.

This value must be specified when the **ApplicationType** directive is set to **WebEnabledForms**.

The default maximum number of connections is the value of **MinInUse**.

Note The **MaxInUse** value is forced to be equal to or greater than the **MinInUse** value.

CloseDelay

When the current number of in-use connections for the **TcpConnection[*n*]** directive exceeds the **MinInUse** value, you can specify the optional **CloseDelay** value to indicates how many seconds before the extra connections are closed after they become inactive.

This value must be specified when the **ApplicationType** directive is set to **WebEnabledForms**.

The close delay defaults to zero (**0**) seconds before inactive connections are closed, indicating that the connection is never closed. The maximum value is **300** seconds.

ConnectionGroup

The optional **ConnectionGroup** value specifies the name of the connection group associated with the **TcpConnection[*n*]** directive; that is, the unique name that is used by applications of type **WebEnabledForms** to identify the TCP/IP port to which a user request is to be sent.

Although there is no default connection group, you must specify a unique value when the value of the **ApplicationType** directive is set to **WebEnabledForms**.

VirtualDirectory

The **VirtualDirectory** directive, whose characteristics are listed in the following table, is an easier way to specify the same value as the **PhysicalDirectory** directive.

The **VirtualDirectory** directive is mainly of use when the same **jadehttp.conf** file is designed to run on multiple hosts.

Characteristic	Value
Action	Specifies the TCP/IP connection to the Jade application
Syntax	VirtualDirectory *directory-name*
Context	Location
Handler	jadehttp-handler
Module	mod_jadehttp
Example	VirtualDirectory /var/web/myWebApp/files

The following is an example of the use of the **VirtualDirectory** directive.

```
<IfModule mpm_winnt.c>
    <Directory "c:/temp/">
        ...
    </Directory>
    Alias /jade/images/ "c:/temp/"
</IfModule>

<IfModule !mpm_winnt.c>
    <Directory "/tmp/">
        ...
    </Directory>
    Alias /jade/images/ "/tmp/"
</IfModule>
<location /jade/webform>
    SetHandler jadehttp_handler

    VirtualDirectory /jade/images/

</location>
```

In this example, depending on where this **jadehttp.conf** file is used, the physical directory to which transferred files are written is different. The **VirtualDirectory** directive uses the URL location of **/jade/images/** to determine the actual directory name to use. For details about retaining files passed to **mod_ jadehttp**, see "Firewall for the Jade Internet Environment", earlier in this chapter.

Apache Configuration Examples

In the configuration examples in this section, the following has assumed to have been added to the **httpd.conf** file, which is the default main configuration file for Apache.

The primary Apache configuration file is normally called **http.conf**. You can place **mod_ jadehttp** directives in this file or in a separate file.

The recommended method is to create a separate **jadehttp.conf** file that contains the required **mod_ jadehttp** directives, and include this file in the primary **http.conf** file as follows.

```
LoadModule jadehttp_module modules/mod_jadehttp.so
Include conf/jadehttp.conf
```

Each of the examples in the following subsections has a different **jadehttp.conf** file.

Minimal Configuration Example

The following is a minimal example of the **jadehttp.conf** file that enables you to connect to the Jade example **Erewhon** system and access the web shop.

```
<IfModule mod_jadehttp.c>
<Directory "/pictures">
    Order deny,allow
    Allow from All
</Directory>

Alias /jadeImages/ "/pictures"

    <Location /jade-info>
        SetHandler jadehttp-info
        order deny,allow
        allow from localhost
    </Location>
    <Location /Erewhon>
        SetHandler jadehttp-handler
        ApplicationType WebEnabledForms
        Application WebShop
        TcpConnection jadeserver 6007 3 15 300 Forms
        VirtualDirectory /jadeImages/
    </Location>
</IfModule>
```

From your web browser, use the **http://webserver/Erewhon** URL to connect to the example **Erewhon** system.

To find out status information about what is happening inside the **mod_ jadehttp** module, use the **http://localhost/jade-info** URL on the workstation running the Apache web server.

Extended Configuration Example

Add the following to the **jadehttp.conf** file, referred to in the **httpd.conf** file under "Apache Configuration Examples", earlier in this chapter (or you can include it in the **httpd.conf** file itself).

```
<IfModule mod_jadehttp.c>

    <Location /jade>
        SetHandler jadehttp-handler
        FaultDocument TcpConnectFailed "/messages/CanNotConnect.html"
    </Location>

    <Location /jade/info>
        SetHandler jadehttp-info
    </Location>

    <Location /jade/Erewhon>
        ApplicationType RestServices
        Application RestTestApp
        TcpConnection cnwjdc1a 6007 1 10
    </Location>

    <Location /jade/Erewhon>
        ApplicationType WebServices
        Application Erewhon
```

```
        TcpConnection cnwjdc1a 6007 1 10
    </Location>

    <Location /jade/erewhon/user>
        ApplicationType WebEnabledForms
        Application UserApp
        TcpConnection cnwjdc1a 6007 3 10 30 Group1
    </Location>

    <Location /jade/erewhon/admin>
        ApplicationType WebServices
        Application AdminApp
        TcpConnection cnwjdc1a 6018
    </Location>

</IfModule>
```

Extended Configuration Example with Additional Apache Directives

Add the following to the **jadehttp.conf** file, referred to in the **httpd.conf** file under "Apache Configuration Examples", earlier in this chapter (or you can include it in the **httpd.conf** file itself).

```
<IfModule mod_jadehttp.c>

    <Location /jade-info>
        JadeHttp_Trace 0xffff Logs/jadehttp.log 5M
        SetHandler jadehttp-handler
        FaultDocument TcpConnectFailed "/messages/CanNotConnect.html"
        ErrorDocument 503 "http://localhost:8080/messages/Custom503.html"
    </Location>

    <Location /jade/info>
        SetHandler jadehttp-info
        Order deny,allow
        Deny from all
        Allow from 127.0.0.0/8 10.1.1.100
    </Location>

    <Location /jade/BankOfJade>
        JadeHttp_Trace 0xff
        ApplicationType HtmlDocuments
        Application BankApp
        ConnectionGroup FinanceDepartment
        TcpConnection cnwjdc1a 6007
        Order deny,allow
        Deny from all
        Allow from 192.168.0.0/16
    </Location>

    <Location>
        ApplicationType WebEnabledForms
        Application Company
        ConnectionGroup Customers
        TcpConnection Host1a 2000 1 5 300 CompanyForms
        Order allow,deny
        Allow from all
    </Location>
```

```
    <Location /jade/erewhon/user>
        ApplicationType WebServices
        Application WebCompany
        TcpConnection Host1a 21000 5 15 300
        TcpConnection2 Host2a 21001 5 15 300
        Order deny,allow
        Deny from all
        Allow from 143.96.0.0/16
    </Location>

<Location /jade/erewhon/user>
        ApplicationType RestServices
        Application RestTestApp
        TcpConnection Host1a 26000 2 10 300
        TcpConnection2 Host2a 26001 2 10 300
        Order allow,deny
        Allow from all
    </Location>

    <Location /jade/erewhon/user>
        ApplicationType HtmlDocuments
        Application HtmlCompany
        TcpConnection Host1a 22000 2 10 300
        TcpConnection2 Host2a 22001 2 10 300
        Order allow,deny
        Allow from all
    </Location>

    <Location /jade/erewhon/admin>
        ApplicationType WebServices
        Application WebCompany
        TcpConnection Host1a 21000 5 15 300
        TcpConnection2 Host2a 21001 5 15 300
        Order deny,allow
        Deny from all
        Allow from 127.0.0.0/8
    </Location>

</IfModule>
```

Apache Considerations

When using the Apache HTTP Server to connect to your Jade applications from the Internet, consider the following.

- The Jade web design considerations documented under "Connecting to Jade Applications from an Apache HTTP Server", earlier in this chapter.
- Only **JadeInternetTCPIPConnection** class TCP/IP communications are supported. This applies to all versions of **mod_ jadehttp**, regardless of the operating system.
- The security implications for **mod_ jadehttp** are those of the IIS **jadehttp.dll**. Access to Jade data from the web server can be controlled by Apache security directives.
- Apache and the HTTP generally use UTF-8 as the encoding scheme for Unicode data on the web. As **mod_ jadehttp** does not allow for this and passes the data directly to Jade, only ANSI data can be read or written to the **JadeInternetTCPIPConnection** object.

- As the **[JadeServer]** configuration directive that indicates details about the Jade application to which **mod_jadehttp** is connected is not yet implemented, it is assumed that the value of this directive is as follows.

```
JadeServer 7.1 littleEndian Ansi
```

Scaling Dynamic Worker Pool Connections

Jade configuration settings affect the operation of the dynamic worker pool used by web server applications such as REST servers and SOAP servers.

You can configure a pool of workers to be dynamically scaled in response to changes in the incoming communication load. When the incoming rate of requests exceeds the response capacity of the server, the Jade application detects this situation and adds processing capacity by spawning another worker application.

An incoming request is added to the request queue if there are no workers available to process the request. If the number of queued requests continuously remains above the queue depth threshold for the configured period, an internal **JadeMultiWorkerTcpManager** class, which manages TCP connections to a Jade application, adds a new worker to the pool if the number of current workers is less than the configured maximum number of workers. Each *worker* in the pool is a complete single-threaded Jade application, running on the same Jade node as the original Jade application that spawned it.

If a worker process is idle for more than the configured timeout period, it is removed from the worker pool if the total number of workers is greater than the configured minimum number of workers.

The scaling of the pool of workers is controlled by the following initialization file settings.

- The [WebOptions] section of the Jade initialization file (**jade.ini**) can contain the following parameters for Jade applications to directly control the scaling of the dynamic worker pool.
  - MaximumInUse, which specifies the maximum number of instances of a Jade application that will be run on a single Jade node. It should be matched to the resources available (that is, the number of CPU cores with allowance for the operating system and the Jade system application) or a target resource cost.
  - MinimumInUse, which specifies the minimum number of instances of a Jade application that will not be subject to removal if they become idle.
  - QueueDepthLimit, which specifies the number of queued requests (that is, the queue depth) that must be continuously exceeded before a new worker will be spawned.
  - QueueDepthLimitTimeout, which specifies a time duration. If the queue depth limit has been continuously exceeded for the specified time, a new worker is spawned.
  - WorkerIdleTimeout, which controls how long an application instance is retained in the connection worker pool when it is idle.

  System designers need to consider their overall strategy when setting values for the dynamic worker pool scaling. If the server is to be configured for maximum responsiveness, set a queue depth limit of **1**, a queue depth limit timeout of **5** seconds, and a worker idle timeout of **10** seconds. If the server response is to be optimized for resource cost, more-detailed analysis will be required. For example, a REST request may take 5 milliseconds to service. The maximum response time that client applications may accept for that request could be half a second. A request could therefore sit in a queue that was 100 entries deep before it would exceed the acceptable response time.

The request queue grows when the incoming rate of requests is greater than the capacity of the server to respond. Even though the load may increase beyond the capacity of the server to respond, it will take some finite time before the queue reaches an unacceptable depth. For example, a 20 percent increase in request load for 5 seconds could result in the maximum queue depth increasing from 3 to 8 requests if the incoming request rate was low (that is, 10 per second) and the response time was high (that is, 100 milliseconds). In this example, if an acceptable response time is 1 second, there would be no need to add server capacity. The overall system performance criterion could be met for a lower resource cost.

Optimal configuration for any system will therefore be a matter of managing the server queue depth, or more precisely, the likely rates at which the queue depth can grow during the specified queue depth limit timeout period.

The growth rate of the queue depends on several factors, including the:

- Average time for requests when the server has spare capacity
- Asymmetry between the incoming request rate and the capacity of the server to respond
- Amount of time the asymmetry exists
- Distribution of incoming request types (assuming not all requests take the same amount of time to service)

Given the number of factors, a system designer would probably need to determine the best queue depth limit and queue depth limit timeout values experimentally. A good starting point for any such experimentation would be to start with a maximum queue depth calculation for each request type and then work in factors like expected load asymmetries (request volatility) and the likely distribution of request types.

Tip When the range in response times for different request types is large or distributions hard to predict, consider servicing these requests in different web server applications.

To disable all dynamic worker pool scaling, configure all of the above parameters in the [WebOptions] section of the **jade.ini** file to zero (**0**).

- The **MaxInUse** parameter in the [*application-name*] section of the **jadehttp.ini** file affects the dynamic worker pool scaling. This setting does not control the scaling, but it can prevent the scaling from working if it is set too low.

  The value must be large enough to allow enough connections between IIS and Jade so the queue depth remains greater than the **QueueDepthLimit** parameter value for the period configured in the **QueueDepthLimitTimeout** parameter defined in the [WebOptions] section of the **jade.ini** file.

  In some respects, the **MaxInUse** parameter sets the upper limit for the response time, but it is better to manage response times using the parameters in the [WebOptions] section of the **jade.ini** file. In general, most REST server usage patterns would be well-served by a **MaxInUse** parameter value that is 10 times larger than the queue depth limit, with a value not less than **50**.

  The **MaxInUse** parameter is the only configuration setting the affects how often the system will respond with a “Server Unavailable” 503 response. In a direct HTTP configuration, all requests are queued, and as such, never return a 503 response.

- The specified number of application copies in the **Application Copies** text box on the **Web Options** sheet of the Define Application dialog provides the Jade node with the starting number of worker instances.

Tuning Your Systems

The Jade object manager uses a default hash value of **2053**:

- When storing and locating persistent and transient locks

  You can vary these hash values (for example, for systems that are likely to have a large number of concurrent persistent or transient locks) by using the **PersistentLockHashSize** or **TransientLockHashSize** parameter in the [JadeClient] and [JadeServer] sections of the Jade initialization file.

- For notifications

  You can vary this hash value (for example, for systems that have a large number of notifications registered) by using the **NotificationHashSize** parameter in the [JadeClient] and [JadeServer] sections of the Jade initialization file.

Specifying Arguments in the Jade Command Line

The Jade command line (or program target), used when running a Jade application from outside of the development environment, contains arguments that enable you to specify your Jade configuration. Each element of the command line has an identifying prefix, followed by an equals sign (**=**) and a specified value.

Command line elements can be delimited by a space, a tab, or a comma. Arguments containing these delimiters must be enclosed in single quotes (**'**) or double quote (**"**) characters.

Notes You can define arguments in the [JadeCommandLine] section of the Jade initialization file that you would normally specify on the command line and you can define two-level initialization file section names so that multiple programs on the same host can share a Jade initialization file. For details, see "Two-Level Section Names" and "Sharing Jade Initialization Files", in the *Jade Initialization File Reference*. See also "Placing Initialization File Parameters on the Command Line", in the following subsection.

As the command line structure of the batch Jade Database utility program (**jdbutilb**) differs from that of other Jade applications, the specification of a command line **name** argument that matches an initialization file section unique identifier does not apply.

Unless your Jade initialization file is located in the **bin** directory, specify the **ini** argument on the shortcut command line rather than in the [JadeCommandLine] section so that the executable knows immediately the location of the Jade initialization file that contains all information required by the program.

You can use the **server** argument in a command line to specify the server Universal Resource Identifier (URI) target database and the client-server transport, instead of the **server** and **path** arguments in a command line and the **ServerNodeSpecifications** parameter in the [JadeClient] section of the Jade initialization file. For details, see "Format of the Server URI String", earlier in this chapter.

For details about:

- Running Jade in thin client mode, see "Invoking a Jade Presentation Client" under "Running the Application Server and Presentation Clients", in Chapter 2 of the *Thin Client Guide*
- Running a Jade client application as a standard (fat) client or a presentation (thin) client from the Jade executable (**jade.exe**) command line, see "Running a Jade Client Application using jade.exe", in Chapter 1 of the *Runtime Application Guide*
- For information about the arguments that enable you to handle multiple copies of Jade, see "Handling Multiple Copies of the Jade Program", later in this chapter

The command line arguments are listed in the following table.

Argument	Specifies …	Example
app	Application name	app=Jade
path	Full path of your Jade database directory containing your ***.dat** Jade files	path=c:\jade\system
schema	Schema name (mandatory except when **app=Jade**)	schema=Faults
ini	Jade initialization file path	ini=c:\jade\system\jade.ini
[server]	Server type	server=SingleUser
[thinClient]	That **jade.exe** checks only arguments required to run in thin client mode	thinClient=false
[name]	That the program checks for a matching initialization file section name	name=SecureAppServer
[newcopy]	A new copy of **jade**.**exe** is to be started	newcopy=false
[host]	Host server node name or IP address	host=cnwchcs38
[port]	Port name or number of the host (server) node	port=6015
[interface]	TCP/IP name or the IP address of the client (local) node	interface=cnwme1c
[localport]	Port name or number of the client (local) node	localport=5017
[appServer]	Remote TCP/IP address or name of the application server	appServer=JadeServer
[appServerPort]	TCP/IP port number of the application server	appServerPort=1500
[startAppParameters] and [endAppParameters]	Arguments to enable **HugeStringArray** string elements to be collected from the command line arguments and passed to the initialization method so that you can run non-GUI standard or thin clients from the **jade** executable or non-GUI client **jadclient** program command line	startAppParameters c:\jade\unittest.xml DailyTests codecoverage=true c:\jade\logs\unittest.log endAppParameters
[endJade]	The end of Jade-specific arguments (applicable only to the non-GUI client **jadclient** program, and ignored for all other Jade programs)	endJade
[arguments]	Optional user-defined arguments for the **jadclient** program	MyNonJadeApp.exe

When specifying your Jade command line:

- You must include the application (**app**) argument.
- You must include the **schema** argument (except when **app=Jade**).
- For Jade executables, you must include fully qualified name of your Jade initialization file in the **ini** argument.
- You must include the path (**path**) argument. The **path** argument specifies the location of the Jade database. The database path cannot be greater than 260 characters.

When running in multiuser mode, the path is always relative to the viewpoint of the server node; for example, **c:\jade\system** refers to the **c:** drive of the server node and not that of the local client workstation.

You should specify the fully qualified path name.

When you specify a relative path name in the **path** argument of the command line, the path name is first converted to an absolute path, by using the following rule.

- A relative path name with a single leading slash character is pre-pended by the first two characters of the Jade HOME directory on the server node (that is, ***drive-letter*:**), which is assumed to be the parent of the **bin** directory. For example, if your installation directory is **Jade\bin** (that is, your Jade HOME directory is **Jade**) and you specify **path=system**, the full directory path is **Jade\system** on the server).

In the following examples, the Jade HOME directory is assumed to be **c:\jade**.

Path Specified in the Command Line	Actual Path
path=/jade71/system	c:/jade71/system
path=system	c:/jade/system

- The **ini** argument specifies the location of the Jade initialization file. Letters specifying the drive are relative to the viewpoint of the client node. The initialization file defaults to **Jade**.**ini**, located in your database (system) path.
- The optional **server** argument specifies **SingleUser** or **MultiUser**.

Note If you do not specify a **server** value, the **Server** parameter in the [Jade] section of the Jade initialization file is used. If this parameter is not specified in the Jade initialization file, the server defaults to **MultiUser**.

- When you specify the name of a port rather than the number in the optional **port** argument, the port name cannot start with a numeric value.
- Identifying prefixes are case-insensitive.
- Spaces are required only between the end of one argument and the start of the next.
- Names and paths with embedded spaces should be enclosed in double quote (**""**) or single quote (**'**) characters.
- If you want to specify arguments to your non-GUI client **jadclient** or **jade** application, specify the **startAppParameters** argument as the last of the Jade arguments before the first of your own arguments in the **jadclient** or **jade.exe** command line.

  The **jadclient** or **jade** program treats processing arguments enclosed in double (**""**) or single (**'**) quotation marks after the **startAppParameters** argument as single-string entries in the **HugeStringArray**. The handling of strings in this **HugeStringArray** is application-specific. For example, **dir=program files** is treated as a two-string entry and **dir="program files"** is treated as a one-string entry. How these entries are handled is determined by your application.

  Support for the **endJade** argument is maintained to provide backward compatibility for existing applications. It is recommended that your applications use the [**startAppParameters**] and **endAppParameters** arguments.

  Note All other programs (for example, **jadapp.exe** and **jadload.exe**) do not parse the command line beyond the **startAppParameters** argument, so any subsequent arguments will be ignored.

- Specify arguments (options) once only, in any order.

  If an argument on the command line is duplicated, the last occurrence found on the command line is used.

- When running Jade thin client applications in multiuser mode, ensure that the TCP/IP port number specified in the **jadapp** (or **jadappb**) command line **appServerPort** argument is a different port number from that specified for the database server (for example, the **jadrap** Jade Remote Node Access program).

  The database server TCP/IP port number is specified in the **ServerNodeSpecifications** parameter remote port value in the [JadeClient] section and the **NetworkSpecification** parameter *IP-address* value in the [JadeServer] section of the Jade initialization file.

In multiuser mode, you can optionally use:

- The **host** command line argument, to specify the host server node name or IP address.

  This argument specifies a host name that is resolvable by the Domain Name Server (DNS) or host file, and has a maximum length of **255** characters.

- The **port** command line argument, to specify the host port name or number.

  If you use this argument to specify the port name, you must update the applicable file in a Windows operating system, as follows.

  ```
  /<winnt>/system32/drivers/etc/services

  /<windows9x>/services
  ```

  Note A port name cannot begin with a numeric value.

- The **interface** command line argument, to specify the client (local) node name or IP address.

  This argument specifies a host name that is resolvable by the Domain Name Server (DNS) or host file, and has a maximum length of 255 characters.

- The **localport** command line argument, to specify the client (local) port name or number.

  If you use this argument to specify the local port name, you must update the applicable file in a Windows operating system, as follows.

  ```
  /<winnt>/system32/drivers/etc/services

  /<windows9x>/services
  ```

  Note A local port name cannot begin with a numeric value.

Specify the optional **interface** and **localport** arguments if you want to bind a specific network adapter in a server node that has more than one network adapter installed; for example, to allow an administrator to ensure connections from clients connect on the fastest interface or to allow easier security when used in conjunction with a firewall or router access list. (By default, Jade defaults to the primary network adapter in the node.)

If you do not specify a **host** or a **port** argument, the **ServerNodeSpecifications** parameter value in the [JadeClient] section of the Jade initialization file is used. If neither of these values is specified in the Jade initialization file, they default to **LocalHost** and **6005**, respectively.

The **host**, **port**, **interface**, and **localport** command line arguments, if specified, override (but do not update) the **ServerNodeSpecifications** parameter [*host-name* or *IP-address*], [*remote-port*], [*local-interface-name* or *IP-address*], or [*local-port*] values, respectively, in the [JadeClient] section of the Jade initialization file.

If you are running multiple copies of Jade, you can have multiple **jade.exe** shortcuts using the same Jade initialization file (**jade.ini**) by using the optional **name** argument, instead of having separate files for each copy of Jade that you run. For details about multiple Jade programs on the same host sharing a Jade initialization file, see "Sharing Jade Initialization Files", in the *Jade Initialization File Reference*.

Placing Initialization File Parameters on the Command Line

You can override any Jade initialization file values by placing them on the command line.

Note For Jade executables, specify the fully qualified name of your Jade initialization file in the command line.

Jade first looks for command line arguments of the form *section-name*.*parameter-name*=*value* and if not present, checks the Jade initialization file for the section name and parameter name combination.

Two-level section names are supported in the command line.

Placing Jade initialization file entries on the command line enables:

- Jade developers and system administrators to quickly test alternative configurations, rather than having to create multiple Jade initialization files.
- Jade developers to share one Jade initialization file.

For example, to allow specific application servers to run specified applications only, on the command line, specify the **jadapp** application server executable shortcut command line as follows.

```
R:\Test\jade\bin\jadapp appServerPort=1500 thinClient=true
path=r:\test\jade\system ini=r:\jade\system\test\myjade.ini
JadeAppServer.EnableAppRestrictions=true
JadeAppServer.AllowSchemaAndApp1="MySchema,SecretFixupApp" server=multiUser
host=devsrvr38 port=6015
```

Notes As the size of the command line is imposed by the operating system, this may restrict the number of Jade initialization file values that you can place on the command line. We recommend that Jade administrators keep the placement of initialization file parameters to a minimum, because long command lines can become hard to maintain. However, if you require only a few non-updating initialization file values, putting them on the command line saves the creation of a new Jade initialization file section.

You cannot use the **Application** class **setProfileString** or **setProfileStringAppServer** method to update Jade initialization file parameter values specified on the command line. (Attempts to do so return a value of **false** and the parameter values remain unchanged.)

If you specify Jade initialization file section and parameter names in the command line, initialization file parameter values that are normally updated by methods are *not* updated, as the command line values have precedence. The parameter values in the Jade initialization file remain unchanged in this situation. For example, if you specify **JadeServer.MaxServerThreads=10** in the command line of the Jade Remote Node Access program when the value of the [JadeServer] section **MaxServerThreads** parameter is **5** and you then change the **Maximum Server Threads** text box value in the Thread Configuration dialog to **7**, the command line value of the application remains **10** and the [JadeServer] section **MaxServerThreads** parameter remains **5**.

When obtaining values for command line arguments (for example, **path**, **schema**, **appServer**, and so on), the order of precedence is as follows.

1. Command line
2. If **name**=*unique-identifier* is on the command line, the [*unique-identifier*.JadeCommandLine] section parameter values are obtained from the Jade initialization file
3. The parameters from the [JadeCommandLine] section of the Jade initialization file
4. An internal default value, if applicable

See also "Location of the Jade Initialization File", "Two-Level Section Names" under "Format of the Jade Initialization File", in the *Jade Initialization File Reference*.

Handling Multiple Copies of the Jade Program

Use the **newcopy** command line argument to force a new copy of the **jade.exe** Jade program to exist; for example, for testing in a multiuser environment. When this argument is set to **true** (the default) and the Jade icon is clicked to launch Jade, a new copy of Jade is launched if the database is already open.

Specify a value of **false** to indicate that when the Jade icon is clicked and Jade is running, another application is started but another copy of the Jade program is *not* started; that is, execution is transferred to the copy of Jade that is currently running.

When initiating a thin client-based **jade.exe** executable with the **newcopy** command line argument set to **false**, execution is transferred to an existing **jade.exe** copy if that copy is running in thin client mode and it is connected to the same application server (the TCP address and TCP/IP port number must match), so that only one copy of **jade.exe** runs multiple applications connected to the same application server.

If no other matching copy of **jade.exe** is running, execution continues as normal.

Note If you create a shortcut that has the **newcopy** argument set to **false** and you specify a different Jade initialization file from the one with which the node was started, the active Jade initialization file is the one that was specified when the node started up and *not* the one specified in the **newcopy=false** shortcut. (You can call the **Application** class **getIniFileName** method in the new application to get the name of the initialization file that was used when the node started up.)

Reregistering a Jade System in Batch Mode

The **jadregb** program enables you to automate the registration of a Jade system with your new licence information by running the registration program in batch mode (for example, from a command script), specifying the following information.

```
jadregb path=database-directory
       [online [ini=Jade-initialization-file-absolute-path-and-name]]
       [help | report | name="licence-name" key=licence-key
       [minStandard=integer-value] [minJade=integer-value]]
```

The number of server licences is split over the two types of run time operation; that is, standard fat client) and Jade thin client.

Note Jade licenses are not transferred automatically between databases in an SDE. It is your responsibility to apply new licenses to any existing databases in an SDE. In addition, to ensure proper operation, you must apply the primary license to every secondary.

You can optionally specify the minimum number of licences reserved for both types of run time operation, if required. When you have successfully specified the new licence information for the database:

- Standard information is output to **stdout** and error information is output to **stderr**.

  For details about displaying and redirecting the output from Jade batch utilities, see the **DisplayApplicationMessages**, **LogServer**, and **UseLogServer** parameters under "Jade Log Section [JadeLog]", in the *Jade Initialization File Reference*.

- The **_control.dat** file is updated.

If the **jadregb** executable program fails, a non-zero exit code is returned and an error message is displayed; for example, the licence key that you specified was invalid or you did not enclose your licence name in double quote characters (**""**). The batch registration program arguments are described in the following subsections.

Using the Jade Version Information Utility

The **jverinfo** Jade Version Information utility enables you to check the hot fixes that are applied to your Jade system and to optionally retrieve the version of user data map files. For more details, see the following subsections.

- Checking which Hot Fixes are Applied
- Obtaining the Version Number of User Database Files

The **jverinfo** program outputs the following version information to a file so that you can view it on your workstation monitor, print it out, or send it to Jade Support if requested to do so.

- System version information (for example, the operating system version, the CPU model, and the registered owner and organization of the software and hardware).
- File version information, starting with the installed location, full release version, and modified timestamp of each executable file followed by the installed location, full version number, and modified timestamp of each Dynamic Link Library (DLL) file.
- Types of CPU and operating system (for example, columns that contain **I686** or **AMD64** for the CPU type and **WinGUI**, **WinCE**, or **SYSV** for the operating system type).
- User data map file version number, starting with the installed location, version number, build type, architecture, and modified timestamp of each user data file.
- As well as listing the version details of executables and libraries found in the **bin** directory, **jverinfo** lists the version information of thin client download binaries if they are in the default download locations (that is, it will not find the download location using the values defined in the Jade initialization file).

  If the directory does not exist or no executable or library files are found, nothing is output.

  The following is an example subset of directories.

  ```
  i686-msoft-win32-ansi\download\bin
  i686-msoft-win32-unicode\download\bin
  armv4i-msoft-wm60-unicode\download\bin
  i686-msoft-x86emu-unicode\download\bin
  ```

The Jade Version Information utility program is not dependent on any other Jade libraries. You can therefore use one version of the **jverinfo** program to report on any release, whether it is a Unicode or an ANSI build.

To run the **jverinfo** program under a Windows operating system:

1. Copy the **jverinfo.exe** program to the Jade binary directory from which you want it run.

   Alternatively, you can create a shortcut on your desktop or in the appropriate Start folder.

2. Execute the program to run it.

By default, the program writes its output to a file called **versioninfo.txt**, which is located in the working directory.

To automate the program, you can optionally specify the binary directory, user data map file directory (which is usually the system directory), system file directory, and the output file name, in the following format.

```
jverinfo [binpath=binary-directory-name]
         [binmask=binary-file-type]
         [datpath=user-data-directory]
         [datmask=user-data-file-type]
         [out=output-file-name|stdout]
         [syspath=system-file-directory]
         [sysmask=system-file-type]
         [noThinClients]
         [noExtraDirs]|
         [extraDirs=[path][directory-name]]
```

The following is an example of the command line set up to run the program.

```
c:\jade\bin\jverinfo binpath=c:\jade\production_bin datpath=c:\jade\system
out=c:\jade\prodver.txt noExtraDirs
```

Note The **jverinfo** command line options are case-insensitive.

You can use the optional:

- **binpath** argument to specify the binary directory that is to be scanned.
- **binmask** argument to refine the files that are output using the **binpath** parameter. The default syntax of this argument is ***.exe**.
- **out** argument to specify **stdout** or the name of the output file that is created (defaulting to **versioninfo.txt**).
- **datpath** argument to specify the user database map file directory (which is usually the **system** directory) that is to be scanned. It outputs an unformatted list of all user data map files, including the version number, the **ANSI**, **Unicode 16**, or **Unicode 32** build type, whether the architecture is **LittleEndian** or **BigEndian**, and the time and date that each file was last modified.
- **datmask** argument to refine the user data files that are output using the **datpath** argument. The default syntax of this argument is ***.dat**.
- **syspath** argument to specify the system file directory that is to be scanned and outputs the system file version, the **ANSI**, or **Unicode 16**, or **Unicode 32** build type, and whether the architecture is **LittleEndian** or **BigEndian**, and the time and date that each file was last modified.
- **sysmask** argument to refine the system files that are output using the **syspath** argument. The default syntax of this argument is **_*.bin**.
- **noThinClients** argument to disable the version details of thin client download binaries, if applicable.
- **extraDirs=*[path][directory-name]*** argument to list custom (non-standard) directories to scan for binaries to display version information. The file specified by the **extraDirs** argument must contain ANSI strings; wide-character strings are not supported by any version of **jverinfo**.

  If you do not specify the optional **=*directory-name*** value, this argument defaults to **jverinfo.extradirs**. The ***directory-name*** value can be an absolute path or a relative path from the Jade Home directory (that is, one level up from the **bin** directory).

  If the directory does not exist or it contains no ANSI files, it is ignored and nothing is output.

- **noExtraDirs** argument, which you can use to disable the **extraDirs** functionality that lists version information about files in non-standard locations (that is, ignore the default **jverinfo.extradirs** file, if it exists).

  If the first character on an input line is the hash (**#**) character, the line is treated as a comment and is ignored.

  The input line can be an absolute path or a relative path from the Jade Home directory (that is, one level up from the **bin** directory).

Checking which Hot Fixes are Applied

Should a problem arise that you cannot resolve by taking the appropriate action described in the relevant error message (documented in the **JadeMsgs.pdf** file) and your Jade licences include support, you may be asked for your current Jade version information in addition to the log files and user mode or process dumps specified in the previous section.

If you are uncertain of which hot fixes are applied to your Jade database, you can perform the following actions to determine exactly which hot fixes have been applied.

- Run the **jverinfo** Jade Version Information utility to obtain your current system and file versions. This stand-alone utility enables you to output your current Jade version information, the consolidated release you are running, and the hot-fixes with which you are running. Although you may think you have applied fixes, you may sometimes find that your records do not reflect the Jade system information itself.
- In the **RootSchema**, call the **Schema** class **getAppliedPatches** method to return a string containing all patches applied by a schema load to system schemas in your database. For details about the format of the string returned by this method, see the **getAppliedPatches** method in Volume 2 of the *Encyclopaedia of Classes*.

Obtaining the Version Number of User Database Files

If you want to obtain the version number of user database map files, you can run **jverinfo** Jade Version Information utility with the optional **datpath** argument, to obtain the current user data map file version number. This argument can be applied only to *user* database map files; that is, user schema files (for example, **_userdev.dat**, **_userscm.dat**, and so on) represented by the **DbFile** class **Kind_User_Schema** constant and user data files (for example, **_rootdef.dat**, **locktest.dat**, and so on) represented by the **DbFile** class **Kind_User_Data** constant.

Use the optional **setUserFileVersions** argument in the **jdbutilb** batch Jade Database utility to set the user data map file version number to the specified 32-bit unsigned integer value.

When you specify the **datpath** argument, the output file contains an unformatted list of all user data map files, including the version number, the **ANSI**, **Unicode 16**, or **Unicode 32** build type, whether the architecture is **LittleEndian** or **BigEndian**, and the time and date that each file was last modified.

In the following example of user data version information, **93** is the specified user data file version number. If a user data version number has not been specified, the version number is zero (**0**).

```
User:                  (path= c:\jade\system)
==============
locktest.dat              93  Ansi,LittleEndian Tue Jun 10 08:16:21 2008
simpleconsumer.dat        93  Ansi,LittleEndian Tue Jun 10 08:16:21 2008
simpleprovider.dat        93  Ansi,LittleEndian Tue Jun 10 08:16:21 2008
wsconsumer.dat            93  Ansi,LittleEndian Tue Jun 10 08:16:22 2008
_environ.dat              93  Ansi,LittleEndian Tue Jun 10 08:16:20 2008
_monitor.dat              93  Ansi,LittleEndian Tue Jun 10 04:00:57 2008
_reports.dat              93  Ansi,LittleEndian Tue Jun 10 08:16:21 2008
_rootdef.dat              93  Ansi,LittleEndian Tue Jun 10 04:00:57 2008
_stats.dat                93  Ansi,LittleEndian Tue Jun 10 04:00:07 2008
_userdev.dat              93  Ansi,LittleEndian Tue Jun 10 08:16:21 2008
```

```
_usergui.dat              93  Ansi,LittleEndian Tue Jun 10 08:16:20 2008
_userint.dat              93  Ansi,LittleEndian Tue Jun 10 08:16:20 2008
_userscm.dat              93  Ansi,LittleEndian Tue Jun 10 08:16:20 2008
_userxrf.dat              93  Ansi,LittleEndian Tue Jun 10 08:16:20 2008
```

In the **RootSchema**, call the **DbFile** class **getUserPatchVersion** method to return a 32-bit unsigned value as an **Integer64** primitive type, which is the unformatted version number of user data map files.

If a user data version number is not set, this method returns zero (**0**).

Jade Sentinel

Jade Sentinel (**jadesentinel.exe**) is a debugger that attaches to an executable and takes a process dump when necessary. Every Jade executable will attempt to launch Sentinel on startup, unless the option is disabled in the Jade initialization file. There are two benefits to Jade Sentinel:

1. Sometimes a program cannot take a process dump of itself.
2. Sentinel (and any other Jade executable) knows how to exclude irrelevant database cache from the dump, which extends the time and space required to create the process dump, if included.

Sentinel can also be started manually to take a process dump of a specified program. This is useful when Jade appears to be hung, you want a process dump for diagnosis, and you want to exclude the database cache to reduce downtime. To use this feature, the command line is:

```
jadesentinel.exe ini=path-and-name-of-initialization-file-of-system
pid=process-id-of-process-to-dump mode=dumpProcess
```

Sentinel returns the exit values listed in the following table.

Value	Description
0	Success, no error
1	Incorrect command line
2	Manual dump failed

The Jade initialization file should be supplied so that Sentinel can obey the [FaultHandling] and [JadeSentinel] sections of that system. The process id should be specified in decimal format. 32- and 64-bit executables must be dumped by 32- and 64-bit sentinels, respectively, and you must use the **jadesentinel.exe** that is located in the **bin** folder of the system you are dumping.

Chapter 3 Containerization

This chapter covers the following topics.

Applies to Version: 2020.0.01 and higher

Overview

Containerization is defined as a form of operating system virtualization through which applications are run in isolated user spaces called containers, all using the same shared operating system.

A container is essentially a fully packaged and portable computing environment.

- Everything an application needs to run – its binaries, libraries, configuration files, and dependencies – is encapsulated and isolated in its container.
- The container itself is abstracted away from the host operating system, with only limited access to underlying resources – much like a lightweight virtual machine (VM).
- As a result, the containerized application can be run on various types of infrastructure – on bare metal, within VMs, and in the cloud – without needing to refactor it for each environment.

Compared to a VM, there is less overhead during start-up and no need to set up a separate guest operating system for each application since they all share the same operating system kernel. Because of this efficiency, containerization is commonly used for packaging up the many individual microservices that make up modern applications.

Console Remote Access Program (jadrapb)

Run the console version of the Jade Remote Access Program from a command line, specifying the following.

```
jadrapb path=database-path ini=Jade-initialization-file-path
```

The following is an example command line.

```
jadrapb path=c:\jade\system ini=c:\jade\system.ini
```

The following is an example Dockerfile entry point specification.

```
WORKDIR /LogMonitor
SHELL ["c:/LogMonitor/LogMonitor.exe", "powershell.exe"]

# define the entrypoint process
ENTRYPOINT c:/jade/bin/jadrapb.exe ini=c:/jade/system.ini, path=c:/jade/system
persistentdb.journalrootdirectory=c:/jade/journals, jadelog.logfile=db_server,
jadelog.logdirectory=c:/jade/logs
```

Container-Ready Services

The service-hosting console applications listed in the following table have been extended to operate correctly in Docker containers.

Application	A console application...
jadrapb	To provide a docker container-ready entry-point process.
jadappb	That runs an application server node as a background process.
jadclient	That runs a standard client node as a background process. This is the preferred way to run a SOAP- or REST-based web service in a Docker container.

Container-ready processes shut down gracefully when the container is stopped by a user or a container orchestrator.

Docker Images

Docker images that can be used to configure and deploy a fully containerized Jade environment are served from the Jade Container Registry (JCR) [**registry.jadeworld.io**].

Each container image is configured to run a single Jade process. The code file variants cover Unicode, ANSI, and x64 variants.

Image Naming Convention

Jade container images are named according to the following convention.

registry-name/jade/*component-name*:*TAG*

The *component-name* is one of the following.

- *database-server*
- *application-server*
- *application-database-server*
- *non-gui-client*
- *gui-client*

The ***TAG*** is used to identify the version and build configuration, using the following format.

```
build-version-architecture-codeset
```

The ***TAG*** values are as follows.

- *build-version* is a Jade version string; for example, **22.0.01**
- *architecture* is **x64**
- *codeset* is **U** for Unicode or **A** for ANSI

The following is an example of the tag value.

```
22.0.01-x64-U
```

The following is an example of the full image name (or tag).

```
registry.jadeworld.io/jade/database-server:22.0.01-x64-U
```

Jade Images

Docker images for the Jade services are provided.

The following table lists the description and entrypoint process for each container image.

Component Name	Description	Entrypoint Process
database-server	Database server	jadrapb.exe
application-server	Application server node	jadappb.exe
application-database-server	Application server with local database	jadappb.exe
non-gui-client	SOAP or REST web service	jadclient.exe
gui-client	Web application or GUI web service	jade.exe

The following table lists the image names for each Jade component. Image names are used in docker **pull** commands, to pull Jade container images from the Jade container registry; that is, **registry.jadeworld.io**.

Component Name	Image Names
database-server	registry.jadeworld.io/jade/database-server:22.0.01-x64-U registry.jadeworld.io/jade/database-server:22.0.01-x64-A
application-server	registry.jadeworld.io/jade/application-server:22.0.01-x64-U registry.jadeworld.io/jade/application-server:22.0.01-x64-A
application-database-server	registry.jadeworld.io/jade/application-database-server:22.0.01-x64-U registry.jadeworld.io/jade/application-database-server:22.0.01-x64-A
non-gui-client	registry.jadeworld.io/jade/non-gui-client:22.0.01-x64-U registry.jadeworld.io/jade/non-gui-client:22.0.01-x64-A
gui-client	registry.jadeworld.io/jade/gui-client:22.0.01-x64-U registry.jadeworld.io/jade/gui-client:22.0.01-x64-A

In the image names listed in the above table, the **TAG** value *build-version* is the Jade version string for the release. For details about image names, see "Image Naming Conventions", elsewhere in this chapter.

Container Logging

Jade container images are configured to write JOM message logs to a mappable internal directory that can be 'bind mounted' to an external directory on the host file system. This provides a simple way to persist and view **jommsg.log** output from outside a running container.

In addition, JOM log messages are written to the **STDOUT** pipeline so that log output can be accessed by the Docker engine or by log collection tools such as **Fluentd** or **Logstash**.

Logging to **STDOUT** is achieved by specifying **JadeLog.UseLogServer=true** and **JadeLog.LogServer=Console** as **Entrypoint** command line arguments.

Windows Base Image

Jade container images use the Long-Term Servicing Channel base image https://hub.docker.com/_/microsoft-windows-servercore tagged **ltsc2019**.

The image is pulled from **mcr.microsoft.com/windows/servercore:ltsc2019**.

Image Update Policy

Jade base images are updated to include cumulative hotfixes as they are released and on a regular basis to incorporate security and bug fix updates rolled out by Microsoft in updates to Windows server core base images.

Note As part of the containerized delivery way of working, we recommend that you put a process in place to ensure you update your image references to the latest base images on a regular basis.

Support Policy

Jade supports running Jade containers in on-premises configurations or cloud-based container platforms capable of correctly running Windows images based on **windows/servercore:ltsc2019**. If you experience issues or have questions about Jade container-related Docker functionality, Jade is your first point of contact.

For information about Microsoft's support policies for containers and related services on Azure, see:

https://learn.microsoft.com/en-us/troubleshoot/azure/general/support-policy-containers

For information about Microsoft's support policy for Windows containers and Docker in on-premises scenarios, see:

https://learn.microsoft.com/en-US/troubleshoot/windows-server/containers/support-for-windows-containers-docker-on-premises-scenarios

Jade Container Examples

Several examples that demonstrate how to use Jade containers to stand up a variety of application environments are provided on a public GitHub repository at:

https://github.com/jadesoftwarenz/JADE-container-examples

The examples are all based on the Erewhon sample application. Each example has a **README** that explains how to deploy and run the example in a step-wise fashion as well as a single script that can be run to deploy a fully operational example environment on a laptop, PC, bare-metal server, or in the cloud.

The example repository also contains documentation about how to get started using Docker Desktop for Windows and provides a script to automate the setup process.

Appendix A Exit Values

This appendix covers the following topics.

Overview

Jade provides exit values that apply to Jade programs and utilities. These exit values enable Jade administrators to develop tools that can take appropriate actions based on the exit values of the programs.

Jade programs can return exit values in the range of zero (**0**) through **2^32**. Jade programs generally returned zero (**0**) for success or the Jade error number if a problem occurs.

Standard exit values are limited to the range **0** through **127**. Any value above this range is reserved by the operating system. You can derive the signal number that caused the program to exit by subtracting 128 from the exit value; for example, an exit value of 139 indicates that the program exited because it received a fatal signal 11 (*SIGSEGV - Invalid memory reference*).

Using standard exit values, ranges of Jade error numbers are grouped into a common generic exit value. In addition, a range of exit values is set aside for warnings, enabling a program to return information without it being regarded as a fatal problem.

Jade programs or utilities default to using current exit values, or the generic exit values by setting an initialization file parameter. For details, see "Enabling the Use of Generic Exit Values for Windows", in the following section.

Note Using generic exit values does not impact on performance of your Jade programs or utilities. It gives the site administrator the ability to write standard support tools.

Enabling the Use of Generic Exit Values for Windows

The [FaultHandling] section of the Jade initialization file provides the **StandardExitValues** parameter that enables you to specify that Jade programs or utilities return generic exit values.

This parameter defaults to **false**, as Jade programs and utilities can return exit values in the range of zero (**0**) through **2^32** by default.

To specify that programs and utilities return generic exit values, set the **StandardExitValues** parameter in the [FaultHandling] section of the Jade initialization file to **true**.

In addition, you can use the:

- **CustomExitValue**<*Jade-error-message-number*> parameter in the [FaultHandling] section of the Jade initialization file when the **StandardExitValues** parameter is set to **true**, to remap Jade exit values (error numbers) to a user-defined standard exit code in the range 32 through 63.

The **CustomExitValue** parameter is valid only when the **StandardExitValues** parameter is set to **true**. If the **StandardExitValues** parameter is set to **false**, Jade exits with the applicable Jade error number. If the **StandardExitValues** parameter is set to **true**, Jade exists with the applicable error number from the remap table (listed in the following section) and checks for the existence of the **CustomExitValue** parameter or parameters in the [FaultHandling] section of the Jade initialization file.

- **Node** class **userExitCode** property to specify the value that is returned when a Jade program (for example, **jade.exe**, **jadapp**, **jadrap.exe**, **jaded**, and so on) exits. For more details, see Chapter 1 of the *Encyclopaedia of Classes*.

Tip You can use the **userExitCode** property, for example, to set a non-zero exit code that can then be checked in a batch file by using the **ERRORLEVEL** keyword to check for appropriate values of the **userExitCode** property.

General Exit Values Unique to Each Program

The following table lists the exit values that are unique to each program (that is, an exit value for the **jadload** program may have a different meaning from that for the **jade** program).

Value	Description	Jade Number Range
127	Fatal network errors	-101 through -120
0	Success, no error	0
1	Methods in error were detected by the schema load	8510
2	One or more commands in command file did not complete successfully	8514
3	Upgrade validation failed	8723
4	Reorganization is required	8511
5	Reorganization operation cancelled by user request	3404
6	Reorganization is suspended waiting for the transition to be initiated	3413
7	Versioned control subclasses require reorganization during online deployment	8525
8	Schema load resulted in one or more incomplete schemas	8527
9 through 31	Reserved, on an individual Jade program basis	Unspecified
32 through 63	User-defined exit values	Unspecified
64 through 78	Reserved by the operating system	Unspecified
105	Jade Object Manager errors	4 through 1299
106	Collection exception errors	1300 through 1349
107	Miscellaneous runtime errors	1400 through 1448
114	Trace exception errors	1500 through 1599
127	Remote procedure call (RPC) request errors	1600 through 1699
108	Security errors	1700 through 1799
113	Database engine errors	3000 through 3199

Value	Description	Jade Number Range
116	Synchronized Database Service (SDS) database errors	3200 through 3399
112	Database reorganization errors	3400 through 3499
127	Database server remote interface errors	3500 through 3599
120	Jade language interpreter errors	4000 through 4499
109	Jade Query Engine internal errors	4500 through 4999
114	File handling errors	5000 through 5099
114	Multimedia handling errors	5100 through 5299
114	Sort errors	5300 through 5399
121	Licence errors	5500 through 5599
112	Jade Object Manager data interchange errors	5700 through 5799
112	Internal management message exchange errors	5900 through 5999
119	Compiler errors	6000 through 6999
118	Method and schema file syntax errors	7000 through 7999
110	External database errors	8000 through 8255
125	ODBC errors	8256 through 8499
126	Jade Schema Load errors	8500 through 8599
111	Upgrade errors	8700 through 8799
124	Jade Database utility errors	9000 through 9999
117	TCP/IP network errors	10000 through 10499
122	Jade Monitor errors	11000 through 11999
127	Jade Remote Access Program errors	12000 through 12999
123	Jade user interface run time errors	14000 through 14499
123	Jade translatable string errors	14500 through 14999
114	Print errors	15000 through 15099
114	Editor errors	15500 through 15599
115	Jade Platform development environment errors	16000 through 16999
117	Application connection errors	30000 through 30999
117	TCP/IP connection errors	31000 through 31499
117	Network proxy errors	31500 through 31999
117	Secure Sockets Layer (SSL) errors	32000 through 32499
117	X509 Certificate errors	32500 through 32999
112	Shared memory errors	33500 through 33599
116	Kernel, multiple server, errors, SDS	34000 through 34999
116	SDS and inter-node communication errors	35000 through 35999
64	Command line usage error	

Value	Description	Jade Number Range
65	Data format error	
66	Cannot open input	
67	Addressee unknown	
68	Host name unknown	
69	Service unavailable	
70	Internal software error	
71	System error (for example, cannot fork)	
72	Critical operating system file missing	
73	Cannot create (user) output file	
74	Input/output error	
75	Temporary failure; user is prompted to retry	
76	Remote error in protocol	
77	Permission denied	
78	Configuration error	

Specifying Your Administration Options

When you sign on to Jade, you can select the **Administration** option button from the Select Options group box to specify your installation preferences. The installation preferences are those that apply to all Jade browser and Painter windows in the Jade development database of your installed Jade release.

When Jade has been installed, these preferences can then be changed individually by a user (by selecting the **Preferences** command from the Options menu) or for the whole Jade Platform development environment (by using this option).

The Jade installation preferences provide a form that enables you to select the appropriate sheet to specify the global options for your Jade Platform development environment work sessions.

The options that you can maintain and the sections in this chapter (for options that apply only to the administrative Jade Installation Preferences dialog) or in Chapter 2 of the *Development Environment User's Guide* that describe these options are listed in the following table.

Installation Option	For details, see…
Accelerator Keys	"Maintaining Accelerator Keys", in Chapter 2 of the *Development Environment User's Guide*
Browser	"Maintaining Browser Options", in Chapter 2 of the *Development Environment User's Guide*
Editor	"Maintaining the Editor Display", in Chapter 2 of the *Development Environment User's Guide*
Editor Options	"Maintaining Editor Options", in Chapter 2 of the *Development Environment User's Guide*

Installation Option	For details, see…
Exit	"Maintaining Exit Options", in Chapter 2 of the *Development Environment User's Guide*
Lock	"Maintaining Lock Options", in Chapter 2 of the *Development Environment User's Guide*
Message Box Suppression	"Maintaining Message Box Suppression Options", in Chapter 2 of the *Development Environment User's Guide*
Miscellaneous	"Maintaining Miscellaneous Options", in Chapter 2 of the *Development Environment User's Guide*
Painter	"Maintaining Painter Options", later in this chapter
Relationship	"Maintaining Relationship Options", in Chapter 2 of the *Development Environment User's Guide*
Schema	"Maintaining Schema Options", in Chapter 2 of the *Development Environment User's Guide*
Short Cut Keys	"Maintaining Shortcut Keys", in Chapter 2 of the *Development Environment User's Guide*
Source Management	"Maintaining Source Management Options", in Chapter 2 of the *Development Environment User's Guide*
Status list	"Maintaining Status List Options", in Chapter 2 of the *Development Environment User's Guide*
Text Templates	"Maintaining Text Templates", in Chapter 2 of the *Development Environment User's Guide*
Window	"Maintaining Window Options", in Chapter 2 of the *Development Environment User's Guide*

The Jade Installation Preferences dialog also contains the File, Admin, Admin, and Help menus.

Using the File Menu

Use the commands in the File menu from the Jade Installation Preferences dialog to administer the installation preferences for your Jade development database.

The File menu commands are listed in the following table.

Command	For details, see…	Description
Save & Logoff	Saving Your Options and Logging Off	Saves your options, logs off from Jade, and displays the Jade sign-on dialog
Logoff	Logging Off from the Installation Preferences Dialog	Logs off from Jade and then displays the Jade sign-on dialog
Save & Exit	Saving Options and Exiting from the Installation Preferences Dialog	Saves your options and then exits from the Jade Installation Preferences dialog
Exit	Exiting from the Installation Preferences Dialog	Exits from the Jade Installation Preferences dialog

Saving Your Options and Logging Off

Use the **Save & Logoff** command from the File menu to save your installation preferences and then exit from the Jade Installation Preferences dialog, in preparation for starting a new work session.

To save your installation preferences and then log off

- Select the **Save & Logoff** command from the File menu.

When all installation preferences have been saved, the Jade Installation Preferences dialog is closed after a momentary delay and the Jade sign-on dialog is then displayed, to enable you to enter your password and start a new Jade work session.

Logging Off from the Installation Preferences Dialog

Use the **Logoff** command from the File menu to exit from the Jade Installation Preferences dialog, in preparation for starting a new work session.

To log off from the Jade Installation Preferences dialog

- Select the **Logoff** command from the File menu.

After a momentary delay, the Jade Installation Preferences dialog is closed and the Jade sign-on dialog is displayed, to enable you to enter your password and start a new Jade work session.

Note No installation preferences that you have specified will be saved.

Saving Options and Exiting from the Installation Preferences Dialog

Use the **Save & Exit** command from the File menu to save your installation preferences and then exit from the Jade Installation Preferences dialog.

To save your installation preferences and then exit

- Select the **Save & Exit** command from the File menu.

When all installation preferences have been saved, the Jade Installation Preferences dialog is then closed after a momentary delay.

Exiting from the Installation Preferences Dialog

Use the **Exit** command from the File menu to exit from the Jade Installation Preferences dialog.

To exit from the Jade Installation Preferences dialog

- Select the **Exit** command from the File menu.

After a momentary delay, the Jade Installation Preferences dialog is then closed.

Note No installation preferences that you have specified will be saved.

Using the Admin Menu

Use the commands in the Admin menu from the Jade Installation Preferences dialog to administer the installation preferences for your Jade development database.

The Admin menu commands are listed in the following table.

Command	For details, see…
Reset to Defaults	Resetting Jade Default Preferences
Remove Sources	Removing Source Code
Backup Database	Backing Up Your Development Database from a Client Workstation

Resetting Jade Default Preferences

Use the **Reset to Defaults** command from the Admin menu to reset all installation preferences to the default values.

Any installation preferences that you have specified are then reset to the Jade default preferences. These default values are the installation preferences that were last specified and saved using the Jade Installation Preferences dialog.

If you have not yet specified and saved any installation preferences, the default preferences are those that are supplied with Jade.

Removing Source Code

The **Remove Sources** command enables you to remove source code from your user-defined schemas only. You cannot remove source code from Jade-supplied schemas.

Use this command when you want to release Jade applications that do not contain method source code; for example, when you release a schema containing Jade applications developed for a third-party.

Notes Ensure that you have backed up the schema, including source code, before you use this command.

For details about including recompiled methods in patch versioning when a patch is to be applied to a schema that does not have source available, see "Enabling or Disabling Patch Versioning" and "Setting Up a Patch Number", in Chapter 3 of the *Development Environment Administration Guide*. For details about finding the position in a method source if you want to locate the position at which an exception occurred in an application from which source code has been removed, see "Finding a Method Source Position", in Chapter 4 of the *Development Environment User's Guide*. For details about encrypting extracted source code, see "Encrypting Schema Source Files", in Chapter 10 of the *Development Environment User's Guide*.

Backing Up Your Development Database from a Client Workstation

The **Backup Database** command enables you to backup your Jade Platform development environment database without having to sign off all users or bring down the server node.

You can backup your Jade Platform development environment database online when the database is active for both read and write access. When backing up your Jade database, you can:

- Specify the backup directory path
- Compress or verify backed up files and optionally overwrite existing files
- Lock the database for write access

For more details, see "Backing Up Your Jade Platform Development Environment", in Chapter 3 of the *Database Administration Guide*.

Using the Patch Menu

Use the commands in the Patch menu from the Jade Installation Preferences dialog to administer patch version control for your Jade development database.

Patch version control enables you to set a numeric patch version that records all additions, deletions, and updates made to schema entities (for example, methods, properties, constants, and so on) until a new patch version number is set and the patch version history is removed when the changed entities in the previous patch version are extracted.

The Patch menu commands are listed in the following table.

Command	Documented under…	Description
Enable / Disable	Enabling or Disabling Patch Versioning	Enables or disables patch versioning for your database
Set Patch Number	Setting Up a Patch Number	Sets a new patch version number
Extract Patch	Extracting a Patch Version	Extracts all entities that have changed in the current patch version
Recreate History	Recreating a History of Patch Version Changes	Recreates the history of changes in the current patch version
Remove Patch History	Removing a Patch History of Changes	Removes the patch history that matched specified criteria

For details, see "Patch Versioning" under "Administering Your Jade Environment", in Chapter 3 of the *Development Environment Administration Guide*.

Using the Help Menu

Use the commands in the Help menu from the Jade Installation Preferences dialog to access the standard Common User Access (CUA) help options. (For details about these options, see "Jade Online Help", in Chapter 2 of the *Development Environment User's Guide*.)

The Help menu commands are listed in the following table.

Command	Description
Index	Opens the directory of Jade online help documents
About	Displays details about the current version of Jade

Specifying Your Jade Installation Preferences

The sheets of the Jade Installation Preferences dialog enable you to maintain your Jade installation preferences. The preferences that you specify apply to all new users.

The preferences for the current users are not modified. For details about maintaining preferences for current users and details about the sheets that are common to administrative installation and to individual user preferences, see "Setting User Preferences", in Chapter 2 of the *Development Environment User's Guide*.

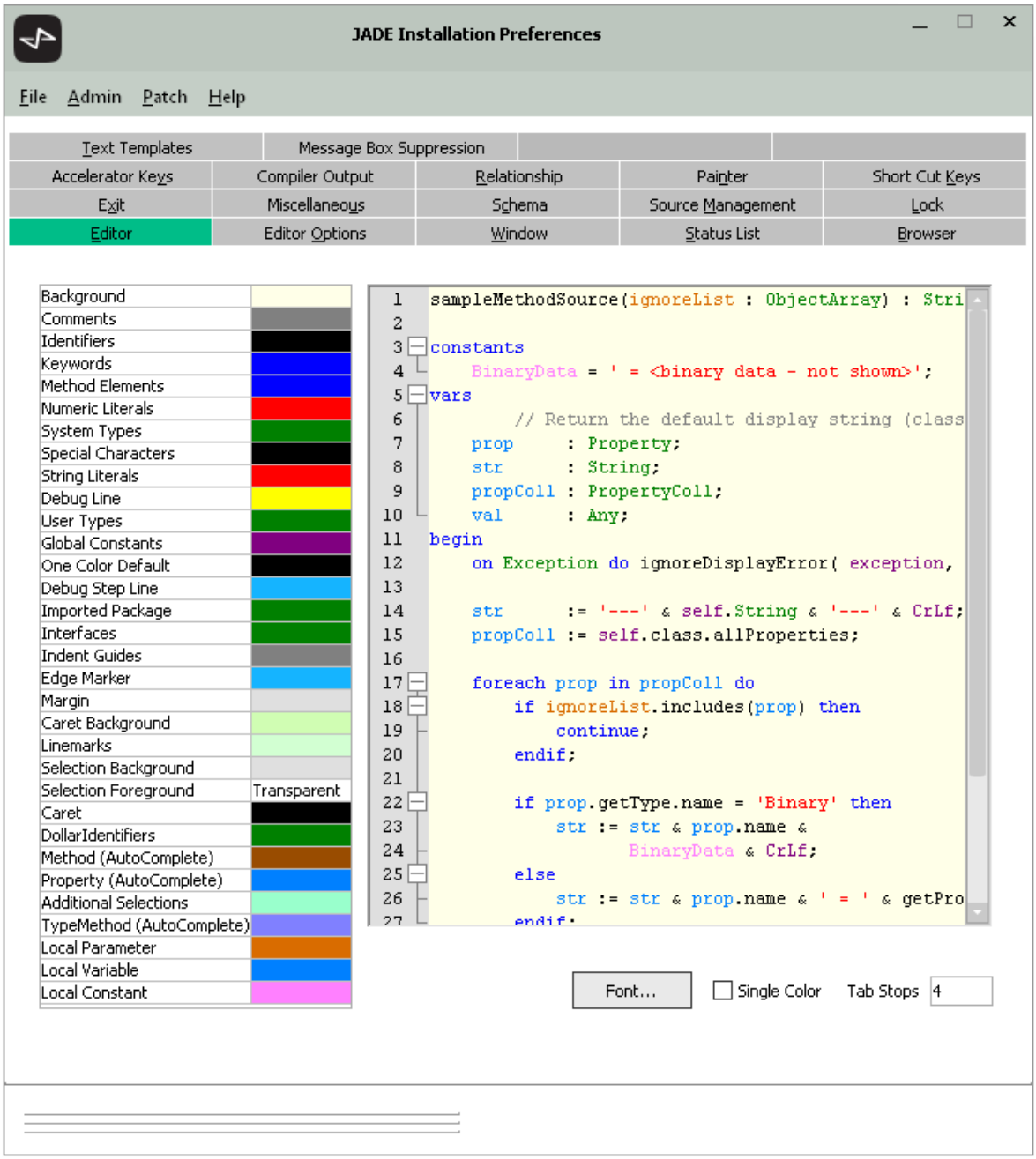
Specify your installation preferences when you first install Jade and before any users sign on, or at any time for new users of the Jade release.

The preferences that you set (and then save) become the new default preferences for your installation of Jade.

To maintain your installation preferences

1. From the Jade sign-on dialog, select the **Administration** option button from the Select Options group box.

   The Jade Installation Preferences dialog, shown in the following image, is then displayed.



The types of options are contained in sheets, with the **Editor** sheet displayed by default. (For details about using the **Editor** sheet, see "Maintaining the Editor Display" and "Maintaining Editor Options", in Chapter 2 of the *Development Environment User's Guide*.)

2. Select the type of options that you want to change, by clicking on the sheet name; for example, click **Status List** if you want to view or change these options.
3. Make the required option changes.
4. To save your changes, use the **Save & Logoff** command or the **Save & Exit** command from the File menu. For details, see "Saving Your Options and Logging Off" or "Saving Options and Exiting from the Installation Preferences Dialog", respectively, earlier in this chapter.

When you have selected and saved your Jade preferences, the specified preferences become the new Jade default preferences when you exit or log off from the Jade Installation Preferences dialog.

Maintaining Painter Options

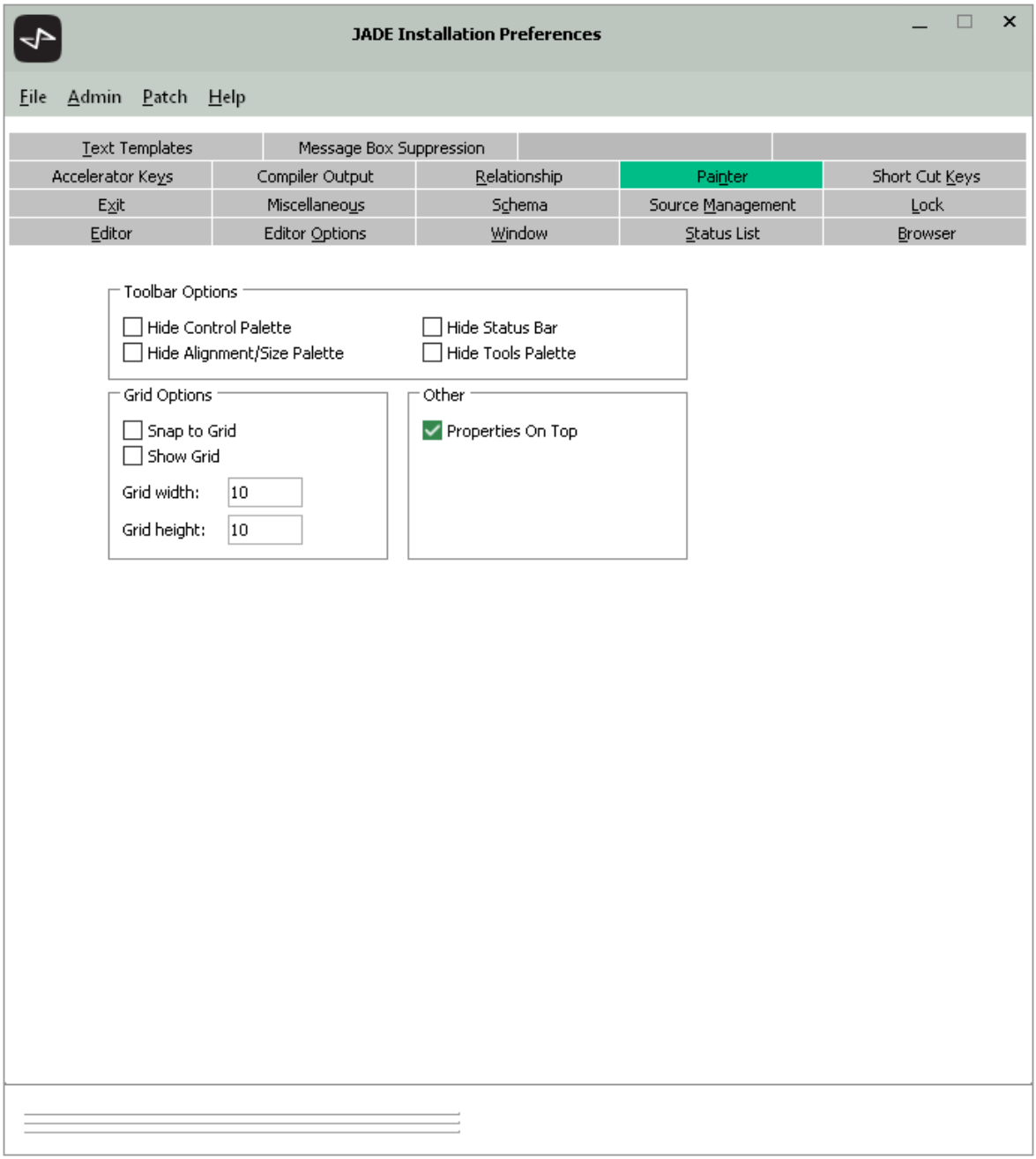
In Jade Painter, the default display of toolbars and grids and the position of the Properties dialog can be specified. For example, you can specify that the **Control** palette is not displayed on the Painter toolbar by default.

As painter values apply to the whole of the Jade Platform development environment, the **Painter** sheet is available only on the administrative Jade Installation Preferences dialog.

To change the default Painter options

1. Click the **Painter** sheet of the Jade Installation Preferences dialog.

   The **Painter** sheet, shown in the following image, is then displayed.



2. If you do not want the **Control** palette displayed in the Painter toolbar, check the **Hide Control Palette** check box.

3. If you do not want the **Alignment and Size** palette displayed in the Painter toolbar, check the **Hide Alignment/Size Palette** check box.

4. If you do not want the status bar displayed in Painter windows, check the **Hide Status Bar** check box.
5. If you do not want the **Tools** palette displayed in the Painter toolbar, check the **Hide Tools Palette** check box.
6. If you want controls on Painter forms to snap to a grid, check the **Snap to Grid** check box.
7. If you want a grid displayed on Painter forms, check the **Show Grid** check box.
8. If you do not want a default grid width of 10 pixels, enter the required width in the **Grid width** text box.
9. If you do not want a default grid height of 10 pixels, enter the required height in the **Grid height** text box.
10. If you want the Properties dialog to be always positioned on top of other control in Painter forms by default, check the **Properties On Top** check box.

Reregistering a Jade System in Batch Mode

The **jadregb** program enables you to automate the registration of a Jade system with your new licence information by running the registration program in batch mode (for example, from a command script), specifying the following information.

```
jadregb path=database-directory
       [online [ini=Jade-initialization-file-absolute-path-and-name]]
       [help | report | name="licence-name" key=licence-key
       [minStandard=integer-value] [minJade=integer-value]]
```

The number of server licences is split over the two types of run time operation; that is, standard fat client) and Jade thin client.

Note Jade licenses are not transferred automatically between databases in an SDE. It is your responsibility to apply new licenses to any existing databases in an SDE. In addition, to ensure proper operation, you must apply the primary license to every secondary.

You can optionally specify the minimum number of licences reserved for both types of run time operation, if required. When you have successfully specified the new licence information for the database:

- Standard information is output to **stdout** and error information is output to **stderr**.

  For details about displaying and redirecting the output from Jade batch utilities, see the **DisplayApplicationMessages**, **LogServer**, and **UseLogServer** parameters under "Jade Log Section [JadeLog]", in the *Jade Initialization File Reference*.

- The **_control.dat** file is updated.

If the **jadregb** executable program fails, a non-zero exit code is returned and an error message is displayed; for example, the licence key that you specified was invalid or you did not enclose your licence name in double quote characters (**""**). The batch registration program arguments are described in the following subsections.

path

The **path** argument specifies the full path of the Jade database directory; for example:

```
c:\jade\system
```

ini

The optional **ini** argument enables you to specify the fully qualified name of your Jade initialization file if it is not located in the database directory or it has a file name other than the default value of **jade.ini**.

name

The **name** argument specifies your registered licence name (displayed on your Certificate of Authorisation, which may be an email message providing you with your licence name and key). For example:

```
"Snazzy Solutions Incorporated"
```

You must specify your licence name, enclosed in double quote characters. This name must be typed correctly (it is case-sensitive), as it is validated against your licence key.

key

The **key** argument specifies the assigned licence key; for example:

```
99999999FFFFFFFF99999999FFFFFFFF
```

Enter the licence key exactly as it is specified on your Certificate of Authorisation, but without spaces.

online

The optional **online** argument specifies that the **jadregb** program attempts to sign on to the Jade Remote Access Program (**jadrap**) to update the licence key.

If you specify the **online** argument, you may also need to specify the **ini** argument, which consists of the full (absolute) path and name of the Jade initialization file that enables the **jadregb** program to obtain the multiuser initialization file settings that it requires to sign on to the Jade Remote Access Program (**jadrap**).

If you do not specify the **online** argument, the **_control.dat** file is updated directly and no client application or database server can be running.

minStandard

The optional **minStandard** argument specifies the minimum number of runtime standard fat client licences that can be reserved at run time to ensure that a specific number of licences is available for standard fat clients at any time; for example:

```
minStandard=4
```

You cannot specify a minimum number of standard fat client licences greater than the number of your registered server (run time) licences.

The default value of zero (**0**) indicates that there is no minimum number of standard fat client runtime licences.

minJade

The optional **minJade** argument specifies the minimum number of runtime Jade thin client licences that can be reserved at run time to ensure that a specific number of licences is available for Jade thin clients at any time; for example:

```
minJade=6
```

You cannot specify a minimum number of Jade thin client licences greater than the number of your registered server (run time) licences.

The default value of zero (**0**) indicates that there is no minimum number of Jade thin client runtime licences.

report

The optional **report** argument displays information about all existing licences installed in your system; for example, primary, secondary, and relational licences.

help

The optional **help** argument displays the required arguments and their values.