Thin Client Guide

VERSION 2020.0.02

jade

Contents

Before You Begin

The *JADE Thin Client Guide* is intended as the main source of information when you are administering JADE thin client environments.

Who Should Read this Guide

The main audience for the *JADE Thin Client Guide* is expected to be system administrators.

What's Included in this Guide

The *JADE Thin Client Guide* has three chapters and two appendixes.

Chapter 1	Covers installing application servers and presentation clients
Chapter 2	Covers running the application server and presentation clients
Chapter 3	Covers administering JADE thin client environments
Appendix A	Covers JADE thin client considerations and restrictions
Appendix B	Covers upgrading software on presentation clients

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…
JADE Database Administration Guide	Administering a JADE database
JADE Developer's Reference	Developing or maintaining JADE applications
JADE Development Environment Administration Guide	Administering the JADE development environment
JADE Monitor User's Guide	Monitoring and examining your JADE environment
JADE *Encyclopaedia of Classes*	System classes (Volumes 1 and 2), Window classes (Volume 3)
JADE Encyclopaedia of Primitive Types	Primitive types and global constants
JADE Initialization File Reference	Maintaining JADE initialization file parameter values
JADE Installation and Configuration Guide	Installing and configuring JADE
JADE Report Writer User's Guide	Using the JADE Report Writer to develop and run reports
JADE Runtime Application Guide	Administering JADE deployed runtime applications
JADE Web Application Guide	Implementing, monitoring, and configuring Web applications

Conventions

The *JADE Thin Client 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 "Invoking an Application Server" 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.
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.

Chapter 1 Installing Application Servers and Presentation Clients

This chapter covers the following topics.

Overview

The separation of the Graphical User Interface (GUI) and database parts of JADE enables you to run JADE in a thin client mode. The JADE application logic is executed on the application server, and the only interaction with the presentation client is to notify the client of any GUI changes and to request information that is available only from that client.

The thin client mode of operation enables JADE to have a smaller operational "footprint" on the user workstation in terms of the required:

- Resource (memory and disk)
- Minimum workstation configuration
- LAN or WAN requirement is significantly reduced

Thin client operation moves most of the processing requirement from the client to the application server and reduces the amount of data that must be communicated when compared with a normal "fat" client. This enables you to run JADE applications over Intranets, dialup lines, and the Internet (World Wide Web).

Thin clients (or *presentation* clients) communicate with one or more application server processes, which in turn communicate with the JADE database on the server node.

Note The term *presentation client* is synonymous with the term *thin client*.

Each application server and each presentation client is considered a JADE "seat". For JADE thin client operation, the application server executable (**jadapp**) can be run on the JADE database server or on other workstations.

The application server executable uses TCP/IP to listen for thin client connections.

The JADE presentation client executable (**jade.exe**) connects to the application server through TCP/IP. The application server executable runs a separate thread for each instance of a connected running application, to provide execution separation.

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.

Remote printing, editing, exception dialogs, and so on, are handled by the application server sending these requests to the presentation client.

When operating in thin client mode, JADE retains initial form definitions previously displayed in a file so that they do not need to be sent again (thin client local caching). All dialogs (message boxes, exceptions, and so on) are invoked on the local workstation.

The **jade.exe** program is considered to be running in thin client mode if:

- Either **appServer** or **appServerPort** appears in the command line
- The **path** and **server** arguments do not appear in the command line

The JADE presentation client executable program **jade.exe** uses its own initialization file, located by default in the installation directory of **jade.exe** that is determined by the value of the **JadeWorkDirectory** parameter in the [JadeEnvironment] section of the JADE initialization file, unless you override this by specifying the **ini** parameter in the JADE presentation client command line. This initialization file contains parameters that affect the JADE presentation client local environment.

By default, the work file directory is created at the same level as the JADE binary directory (**bin**) 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 located 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.

For details about the:

- JADE Home directory, see "Specifying Arguments in the JADE Command Line", in Chapter 2 of the *JADE Installation and Configuration Guide*.
- For details about the **JadeWorkDirectory** parameter in the [JadeEnvironment] section of the JADE initialization file that is used to determine the directory in which work files are created, see the *JADE Initialization File Reference*.
- Directory locations (for example, program data and user data), see "Directory Locations", in Chapter 2 of the *JADE Installation and Configuration Guide*.

As the JADE automatic download installation program **jaddinst.exe** automatically uses the name of the presentation client executable defined in the **PostInstallExe** parameter of the [*environment-type*] environment-specific section of the JADE initialization file if the value is **jade** or **jade.exe**, you can therefore use the same JADE initialization file parameter to re-initiate the presentation client, regardless of the presentation client that initiated the download.

Each presentation client environment supported by JADE has binary files specific to that environment that must be installed. The application server requires different sets of files to be available for downloading to presentation clients. The files must each reside in a different directory identified by the operating system version, and whether they are ANSI or Unicode files. For details about environment-type files, see Appendix B, "Upgrading Software on Presentation Clients".

Caution When you develop an application that could run in JADE thin client mode, subscribe to notifications or use timers with care. When an event occurs, it notifies the application server, which then echoes the event to all attached presentation clients; that is, the application server sends the notification to each presentation client, which then send a response to the application server. This can have a considerable impact on network traffic.

Ensure that you have the appropriate privileges or capabilities to install applications. The configuration of User Account Control (UAC) and your current user account privileges may affect the behavior of the installation.

The Microsoft Windows C++ 2017 Redistributable Package (x64) is required to be installed on all 64-bit Windows systems that run JADE 2018. If 32-bit thin clients are going to be installed, the equivalent C++ 2017 Redistributable Package (x86) will need to be installed. This will be done as part of the normal JADE installation or upgrade process. (This executable is supplied on the JADE distribution media.)

Installing this Microsoft redistributable package requires administration privileges. If possible, deploy this package to all workstations *before* installing JADE 2018, using the appropriate techniques that allow for privileged installations.

Effects of Security on Presentation Client Upgrades

The JADE thin client and installer make use of the JADE directory structure described in the previous section to perform their operations.

If the JADE thin client is installed into an unrestricted area, configuration and running of the presentation client and application server work with no modifications required to the operational procedures.

If JADE is installed in the **\Program Files** directory (or **\Program Files (86**) directory on a 64-bit machine with 32-bit JADE binaries):

- If UAC is disabled for standard users, the thin client upgrade will fail because of lack of permissions.

  For administration users, the necessary privileges are automatically granted so the upgrade will succeed.

- If UAC is not disabled, administrative users are prompted with an **Allow** or a **Cancel** choice, but standard users must know and supply the user name and password of a user with administration privileges to enable the upgrade to succeed.

Read-only data and executables are separated from the created, updated, and deleted dynamic files. The use of data directories for each user is not required for the operation of a presentation client, but can be used for applications.

When running JADE in thin client mode, if the presentation client is installed:

- Under the **\Program Files** directory (or the **\Program Files (x86)** directory on a 64-bit machine with 32-bit JADE binaries), when an automatic presentation client upgrade occurs, you must have administrator rights or know the user name and password of a user with administrator rights. (This is a normal Microsoft requirement that updating of files under the **Program Files** directory requires administration privileges.)

- Outside of the **\Program Files** directory (or the **\Program Files (x86)** directory), privilege elevation to perform an automatic thin client upgrade is not requested.

Running JADE in the Least-Privileged Mode

As the **jade** executable runs within a least-privileged mode, it does not have the security privileges required to create and delete files when located under the **\Program Files** or **\Program Files (x86)** directory but an existing file can be updated.

Users cannot create local files and JADE cannot create or delete files for its internal operations.

JADE creates forms and picture cache files, which need to be shared among one or more users of a JADE thin client installation. During the JADE thin client upgrade process, the lock file to prevent multiple concurrent downloads and the actual download files need to be placed in a shared read-write area of the file system, which is determined by the value of the **ProgramDataDirectory** parameter in the [JadeEnvironment] section of the JADE initialization file and the location of the installation directory.

Running the jaddinst Executable with Elevated Privileges

The **jaddinst** executable that performs the installation of binaries and support files must run with elevated security privileges, to allow it full read and write access into the installation location.

The **jaddinst** executable is built with appropriate manifest entries to allow it to request elevated security permissions. As **jaddinst** can be replaced as part of the upgrade process, any direct assigning of special security rights to the program must be handled carefully. The maintenance of special security rights assigned to **jaddinst** and the transfer of them to the new version of **jaddinst** is not supported.

The JADE Application Server

The application server acts as a proxy JADE executable program, handling:

- All requests to the GUI code from the JADE logic, and the reverse
- All requests to the presentation client for printing, editing, file handling, and so on
- All JADE notifications and timers

Accessing Presentation Client Files

Methods of the **File** and **FileFolder** classes and the **Sound** class **loadFromFile** method are processed on the presentation client when the **FileNode** or **MultiMediaType** class **usePresentationFileSystem** property is set to **true** and the instances are *not* shared transient instances.

Shared transient instances of the **File**, **FileFolder**, and **Sound** classes are processed on the application server and the setting of the **usePresentationFileSystem** property is ignored. For shared transient instances of these classes, the **usePresentationFileSystem** property defaults to **false**. (A file or a folder opened on one presentation client cannot be accessed by another client.)

The schema and form definition load process can access files on the presentation client workstation.

External Function Calls

The JADE thin client mode handles remote function call execution.

Use the **presentationClientExecution** or **applicationServerExecution** option in the function definition of an external function to specify where the function call is made. By default, external functions are called on the presentation client workstation; that is, **presentationClientExecution** is assumed.

Note The **presentationClientExecution** option has no effect if the JADE client is not currently running in thin client mode.

You must define your external function calls for compatibility with the running JADE environment on which the calls are made. For example, calling Windows Application Programming Interfaces (APIs) that pass a Windows handle have different parameter definitions that depend on whether the client is running in 32-bit or 64-bit mode.

To cater for this, you need to establish two different external function call definitions with the required parameter types. Your calling JADE logic must then determine which architecture is being used by the client and which external function call it should therefore make.

Installing Thin Client Components

An application running in JADE thin client mode requires:

- Only a limited number of libraries.

  For details about selecting the JADE set-up **Presentation Client** installation option, see "Selecting the Type of Installation" and "Selecting the Components to Install", in Chapter 1 of the *JADE Installation and Configuration Guide*.

  The presentation client files are also installed as part of the set-up default **Development** installation (which installs the development and application runtime environment and the JADE thin client application server and presentation client files) or as a selected component of the **Custom** installation option.

- A presentation client **jade.exe** command line set-up that differs from that of a standard JADE installation.

- Installation of the application server executable program (**jadapp**).

  The application server is installed as part of the JADE set-up default **Development** installation option; that is, development and application runtime environments and the JADE thin client application server and presentation client files are installed.

  The application server files are also installed as part of the set-up **Runtime** installation option and as a selected component of the **Custom** installation option. For details, see "Selecting the Type of Installation", in Chapter 1 of the *JADE Installation and Configuration Guide*.

- The JADE server node that contains the JADE database requires a slight increase in memory (for details about the recommended memory requirements, see "Minimum Hardware Requirements for Machines Hosting a JADE Database", in Chapter 1 of the *JADE Installation and Configuration Guide*.)

For details about the JADE initialization file [JadeAppServer] and [JadeThinClient] sections that enable you to configure JADE thin clients, see "JADE Thin Client Sections", in the *JADE Initialization File Reference*. For details about upgrading a 32-bit presentation client connecting to a 64-bit application server, see "Automatic Download Issues and Considerations", in Appendix B.

Chapter 2 Running the Application Server and Presentation Clients

This chapter covers the following topics.

Overview

Your JADE schemas and applications can be run in thin client mode simply by specifying the appropriate command line parameters and then invoking the application server. When this is running, you simply specify the appropriate presentation client command line parameters on the workstation that you want to run in thin client mode and then invoke the thin client executable program. For details, see the following sections.

For details about specifying application server or presentation client command line parameters in the JADE initialization file, see "Two-Level Section Names", in the *JADE Initialization File Reference*. See also "Sharing JADE Initialization Files", in the *JADE Initialization File Reference*.

Invoking an Application Server

The command line required to run the application server is as follows.

```
jadapp appServerPort=TCP/IP-communications-port-number
       path=database-path
       ini=JADE-initialization-file-path
       [server=multiUser|singleUser|readOnlyUser]
       [host=host-server-node-name or host-IP-address]
       [port=host-port-name or host-port-number]
       [name=JADE-initialization file-section-name-identifier]
       [interface=client-TCP/IP-name or client-IP-address]
       [localport=client-port-name or client-port-number]
```

The **app** and **schema** arguments are invalid in an application server command line and you must specify the **appServerPort** argument.

Note When running JADE thin client applications in multiuser mode, ensure that the TCP/IP port number specified in the **jadapp** command line **appServerPort** argument is a different port number to 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.

Specify the **server** argument only if you want to run the application server in multiuser mode. (If you do not specify the **server** argument, the application server runs in single user mode.)

For details about these argument values, see "Handling Multiple Copies of the JADE Program", in Chapter 1 of the JADE *Installation and Configuration Guide*.

The following is an example of an application server command line.

```
R:\Jade_Development\jade\bin\jadapp appServerPort=1500 server=multiUser
path=r:\jade_development\jade\system ini=r:\jade\system\test\myjade.ini
host=devsrvr38 port=6015 localport=6099
```

What Happens Next

When you have invoked and signed on to the application server, the Application Server window is then displayed. The title bar of this window contains the current number of connected presentation clients and the application server port number and path. (See also "Running an Application Server as a Service", later in this chapter.)

If the application server was installed as a service, a check mark is displayed at the left of the **Run As Service** command in the Options menu.

The Application Server window itself displays the information listed in the following table for each presentation client that is connected to the application server.

Column	Example
JADE presentation client user name	wilbur1
Presentation Client workstation (PC) name	wilbur1a.trogsunited.com
Schema name	ProductionTest
Application name	AccountsModule
TCP/IP address of the presentation client	143.96.133.78
Time last message was sent to the presentation client	14:28:51
Total user and operating system kernel processor time used by the application logic	0:00:02.484
Number of messages sent from and to the application server	446
Number of bytes that were sent to the presentation client	230706
Number of bytes that were received from the presentation client	44002

You can use the processing time information displayed in the **Proc. time** column to determine which presentation client is looping, for example.

Now that your application server is running, you can initiate your JADE presentation clients.

Tip Use the **DisplayFont** parameter in the [JadeAppServer] section of the JADE initialization file to change the default font that is used in the Application Server window to display all presentation clients currently attached to the application server.

Placing the Application Server Icon in the System Tray

By default, the Application Server window is automatically minimized and an icon is placed in the system tray when the application server (that is, **jadapp.exe**) starts up.

The **Use system tray** command from the Options menu enables you to toggle the minimizing or restoring of the Application Server window to or from the system tray icon. When the window can be automatically minimized and a system tray icon displayed in the Taskbar (the default value), a check mark is displayed to the left of the command in the Options menu and an icon is placed in the system tray in the Taskbar at the lower right of the screen, when the application server program starts up.

The icon is removed from the system tray when the application server closes down or the use of the system tray icon is disabled.

» To toggle the use of the system tray icon

- Click the **Use system tray** command from the Options menu.

When the use of the system tray icon is activated, a check mark is displayed to the left of the command in the Options menu and an icon is placed in the system tray at the lower right of the screen when the application server starts up.

When you disable the use of the system tray icon for the application server, no check mark is displayed at the left of the command in the Options menu, no icon is located in the system tray, and the window is not automatically minimized when the application starts up. In addition, the **UseSystemTrayIcon** parameter in the [JadeAppServer] section of the JADE initialization file is updated to maintain the current value for future work sessions when you exit from the application server.

» To restore the application server window

- Left-click on the application server icon in the system tray at the right of the Taskbar.

The Application Server window is then restored.

» To minimize the window again when the use of the system tray icon is enabled

- Left-click on the application server icon in the system tray again.

The Application Server window is then minimized; that is, left-clicking on the application server icon in the system tray toggles the minimizing and restoring of the Application Server window.

» To access the application server system tray icon menu

- Right-click on the application server icon in the system tray at the right of the Taskbar.

The system tray icon menu for the application server is then displayed. The application server icon menu provides the following commands.

- Restore

  The **Restore** command, determined by the current status of the application server window, toggles the minimizing or restoring of the application server window.

- Use system tray

The **Use system tray** command enables or disables the automatic minimizing of the application server window when the application server starts up and placement of the application server icon in the system tray.

Tip Select this command from the icon menu if you want to disable the use of the system tray icon and automatic minimizing of the application server window. The icon is then removed from the system tray. (When use of the system tray icon is disabled, you can enable it at any time by selecting the **Use system tray** command in the Options menu of the Application Server window.) The **UseSystemTrayIcon** parameter in the [JadeAppServer] section of the JADE initialization file enables you to configure whether the interrupt icon is positioned in the system tray.

- Exit

  The **Exit** command exits from the application server.

Initiating Non-GUI Applications when the Application Server is Initiated

Use the **ServerApplication** parameter in the [JadeAppServer] section of the JADE initialization file to specify a non-GUI application that is executed when the application server node initializes.

Note Non-GUI applications initiated in JADE thin client mode are run wholly on the application server, with no presentation client interaction.

For details, see "Application Server Section [JadeAppServer]" under "JADE Thin Client Sections", in the *JADE Initialization File Reference*.

Initiating Non-GUI Applications at a Specific Time

Specify the optional *time* variable in the **ServerApplication** parameter in the [JadeAppServer] section of the JADE initialization file to specify a non-GUI server application that is executed at a specified time.

Note Non-GUI applications initiated in JADE thin client mode are run wholly on the application server, with no presentation client interaction.

For details, see "Application Server Section [JadeAppServer]" under "JADE Thin Client Sections", in the *JADE Initialization File Reference*.

Invoking a JADE Presentation Client

The command line required to run a JADE presentation client is as follows.

```
jade.exe app=application-name
         ini=JADE-initialization-file-path
         schema=schema-name
         [AppServer=remote-TCP/IP-address-or-name-of-application-server]
         [AppServerPort=TCP/IP-port-number-of-application-server]
         [name=JADE-initialization-file-section-name-identifier]
         [thinClient=true|false]
         [newcopy=true|false]
         [[startAppParameters command-line-arguments]
         [endAppParameters]]
```

As the **path** and **server** arguments are controlled by the application server, they are not permitted in the JADE presentation client command line. You must specify the **app**, **ini**, and **schema** arguments.

If you do not specify the optional **AppServer** or **AppServerPort** argument, the values in the [JadeThinClient] section of the JADE initialization file are used. The **AppServerPort** argument must be the same as that on the application server.

The following is an example of the command line required for a presentation client.

```
D:\JADE\bin\jade.exe app=SortApp schema=SortTest AppServer=JadeServer
AppServerPort=1500 ini=r:\jade\system\test\myjade.ini
```

What Happens Next

When you have invoked the application server, you can then invoke a presentation client from a workstation. When you invoke the JADE presentation client from your workstation, the JADE start-up form displays the application server TCP/IP communications port number and the remote TCP/IP address or name of the application server.

When you have signed on:

- Your JADE work session then functions like any other JADE work session
- Details about your presentation client connection are displayed in the Application Server window

Tip Set the **TerminateProcessOnDisconnect** parameter in the [JadeClient] section of your application server JADE initialization file to **true** if you want to specify that the process is terminated immediately, when the client detects that the network connection from the JADE server node on which the database is located has been lost. (For details, see Chapter 1 of the *JADE Initialization File Reference*, "JADE Initialization File".)

Getting Presentation Client Sign-On Details

When a presentation client connects to an application server and the connection must be made via a proxy server that requires authentication, the **SSLProxyAuthDetails** parameter in the [JadeThinClient] section of the JADE initialization file may be required. This parameter specifies the name and optionally the entry-point of a library file that supplies a user code and password, and enables a support library to obtain the user name and password required by the proxy server, rather than using the default JADE sign-on dialog.

The default value of **<default>** indicates that the proxy server user code and password authentication are not required or that the user identifier and password specified in the connection JADE sign-on dialog are used.

If you use this parameter to specify the name of the input library file, the optional function name value defaults to **getProxyPassword**. This validates the application sign on to a JADE presentation client over an SSL proxy connection if your JADE thin client is behind a firewall and your network administrator requires connections to the Internet to be done through a proxy.

The JADE Application Programming Interface (API) function call interface signature is as follows.

```
extern "C" int JOMAPI authFunction(GETPASSWORD_UNION * authInfo);
```

This function call, whose use is shown in the following example (in which you would substitute your own user code and password to replace the fictitious ones used in this example), returns zero (**0**) for success or a non-zero error condition if the call failed for some reason.

```
extern "C" int JOMAPI
authFunction(GETPASSWORD_UNION * authInfo)
{
    authInfo->GETPASSWORD_STRUCTv1.bValid = true;
    intlStrcpy(authInfo->GETPASSWORD_STRUCTv1.username,
                TEXT("ExampleProxyUserName"));
```

```
        intlStrcpy(authInfo->GETPASSWORD_STRUCTv1.password,
                   TEXT("ExampleProxyPassword"));

      return 0;
    }
```

To indicate that a user name and password are used, set **bValid** to **true**. A value of **false** indicates that you do not want the function call to provide a user name and password.

To view the format of the data structure and callback that handles obtaining a user name and password from at third-party DLL, see the **GETPASSWORD_STRUCT** type definition in the **jomtypes.h** file in your JADE **includes** directory.

Handling the Loss of Thin Client Connections

When a thin client connection is lost, the presentation client displays a message box that has one of the following button combinations, so that you can programmatically retry the connection or programmatically close it.

- Three buttons
  - **Exit**, to terminate the application.
  - **Retry**, to attempt 10 further connection attempts over 10 seconds. If the 10 reconnection attempts all fail, the same message box is displayed again.

    If a connection was successfully made but the session cannot be continued (for example, if the application server was restarted), the **Retry** button is not displayed.
  - **Restart**, which initiates another copy of the application and terminates the current session.

    This button is displayed only if the application was not started programmatically by JADE logic. Applications programmatically initiated (for example, by calling the **Application** class **startAppMethod**, **startAppMethodWithString**, or **startApplicationWithParameter** method) cannot be restarted, because the mechanisms required cannot be guaranteed or reconstructed.

    If the start also fails to establish another connection, another message box is displayed. (See the **Two buttons (when a retry is not an option)** item, later in this list.)
- Two buttons (when a restart is not an option)
  - **Exit**
  - **Retry**
- Two buttons (when a retry is not an option)
  - **Exit**
  - **Restart**
- One button
  - **OK**, when the only option is to terminate. (The displayed message includes the text *You must restart the Application*.)

The full process is therefore as follows.

1. When the connection is lost, JADE attempts to reconnect for the timeout value specified in the **ReconnectTimeout** parameter in the [JadeThinClient] section of the JADE initialization file on a local presentation client (which defaults to 40 seconds).

2. If no connection attempt succeeds in the reconnect timeout period, the appropriate message box (documented in the list earlier in this section) is displayed.
3. For a retry to succeed, the setup requires that the application server waits for an appropriate length of time before giving up on the reconnection and terminating the application.
4. If a connection succeeds but the session cannot be continued because the application server has already terminated the session, the message box that is displayed will not have a **Retry** button.

Note The text for the displayed messages is taken from the **jadmsgs.eng** file (as are most JADE error messages). You can translate or modify this file, if required. For details, see "Translating Messages", in Chapter 12 of the *JADE Development Environment User's Guide*.

Running an Application Server as a Service

You can run the application server as a service under a Windows operating system that supports services. By default, application servers are *not* run as services.

When services are installed or removed, entries in the **HKEY_LOCAL_MACHINE** (HKLM) area of the registry must be modified but standard users do not have the necessary privileges to do this.

The menu options that enable the application server to run as a service are disabled if you do not have the necessary privileges to install or remove an application as a service. Installing, controlling, and removing a service can be performed only if you have sufficient operating system privileges.

JADE does not currently supply a facility to control the order in which services are started or stopped when the host computer is booted. If the application server is started before the database, a message is recorded in the JADE log file.

If you want to install multiple application servers as services on the same host, you must have a separate JADE initialization file for each application server service, because the service installation process must update a unique **NodeName**, **NodeNameDescription**, and **RunAsService** parameter in the [JadeAppServer] section of the initialization file.

» To install an application server as a service

1. From the Application Server window, select the **Run as Service** command from the Options menu. A check mark is then displayed to the left of the command in the Options menu, indicating that the application server is running as a service. The **RunAsService** parameter in the [JadeAppServer] section of the JADE initialization file is set to **true**.

   Note The newly installed application server service is not yet started and any presentation clients connected to that server node are not affected if the application server is currently running.

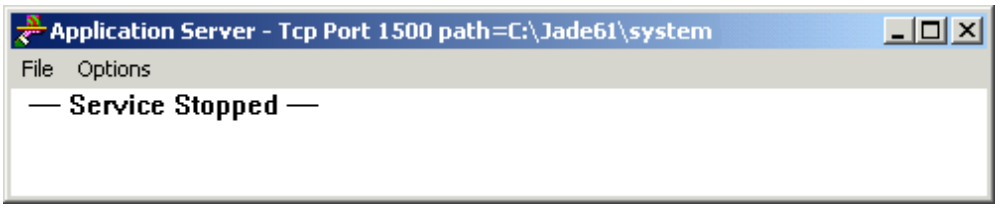
2. When no presentation clients are attached to that application server, exit from the application server (by selecting the **Exit** command from the Application Server window File menu).

When you have installed the application server as a service and then closed the application server, you can then run the application server as a service.

» To start a service after the server has been installed and the application closed

1. Invoke the application server **jadapp** executable program. The application server is then started in control service mode, which does not connect to the JADE database.

The status message in the Application Server window reports whether the service is currently stopped or running, as shown in the following image.



The **Start Service** command in the File menu is now enabled.

2. Select the **Start Service** command from the File menu.

The application server service that is started is determined by the value of the **NodeName** parameter in the [JadeAppServer] section of the JADE initialization file specified in the **jadapp** program command line.

To run more than one concurrent service, you must have a separate JADE initialization file for each application server you want to run as a service, with the appropriate service name specified in the **NodeName** parameter in the [JadeAppServer] section of each JADE initialization file.

When an application server is running as a service:

- The Application Server window displays the **Service Running** status message.
- No connection information is displayed in the Application Server window.
- The application server is connected to the database.
- The **Stop Service** File menu command is enabled.
- The **Run as Service** command in the Options menu is disabled. (If you were to remove an installed service, all presentation clients would be disconnected immediately and the application server would no longer be connected to the database.)

To stop a running application server service

- When the Application Server window displays the **Service Running** status message, select the **Stop Service** command from the File menu.

The **Service Stopped** status message is then displayed in the Application Server window and the File menu **Start Service** and the Options menu **Run as Service** commands are enabled.

Note When you select the **Stop Service** command from the File menu when the application server node is still running as a service, a message box is displayed, advising you that you cannot select this option when the service is running and that you must first stop the service. In addition, the service state (for example, running, stopped, and so on) is continuously updated to reflect service state changes made externally; for example, by using the Task Manager.

To remove an installed service that is not currently running

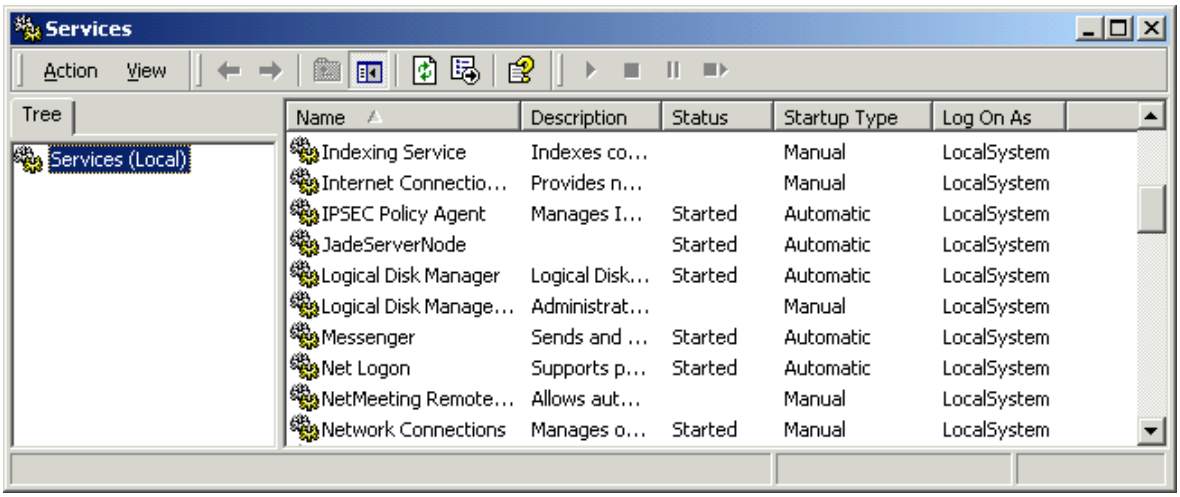
- From the Application Server window, select the **Run as Service** command from the Options menu when it is running as a service and has a check mark displayed to the left of the command. (When a service is installed, this command is enabled only when the service is stopped.)

The check mark is then no longer displayed to the left of the command, the **Control Service Mode** status message is displayed in the Application Server window, and the application server is no longer connected to the JADE database.

To control an installed service using Windows features

- Use the **Services** Control Panel applet to start and stop the service.

  The following image shows an example of the Services window.



  The application server name is the value of the **NodeNameDescription** parameter in the [JadeAppServer] section of the JADE initialization file specified in the **jadapp** program command line.

- Issue the **net start** or **net stop** command from a command prompt.

Running an Application Server in Batch Mode

The **jadappb** program, installed with your JADE software, enables you to automate the application server by running it in batch mode.

Run the batch Application Server program (for example, from a command script), specifying the following.

```
jadappb appServerPort=TCP/IP-communications-port-number
        path=database-path
        ini=JADE-initialization-file-path
        [server=multiUser|singleUser|readOnlyUser]
        [name=unique-qualified-JADE-initialization-file-section-name]
        [host=host-server-node-name or host-IP-address]
        [port=host-port-name or host-port-number]
        [interface=client-TCP/IP-name or client-IP-address]
        [localport=client-port-name or client-port-number]
```

The following is an example of the application server batch command.

```
jadappb appServerPort=1500 path=r:\jade_development\jade\system
ini=r:\jade\system\test\myjade.ini server=singleUser hostdevsrvr38 port=6015
localport=6099
```

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*.

When you have invoked your application server, you can then initiate your JADE presentation clients. If the **jadappb** executable program fails, a non-zero exit code is returned and an error message is displayed; for example, if the database directory was invalid. The **jadappb** program uses the parameters in the JADE initialization file [JadeAppServer] section.

The batch Application Server program parameters are described in the following subsections. (For more details about initiating an application when the application server is initiated or at a specific time, see "Invoking an Application Server", earlier in this chapter.)

appServer

The **appServer** parameter specifies the TCP/IP communications address (for example, **143.57.055.259**) or the name of application server (for example, **wilbur1a**).

appServerPort

The **appServerPort** parameter specifies the TCP/IP communications port number of the application server. This parameter must be specified and it must have a valid TCP/IP port value.

path

The **path** parameter specifies the full path of your JADE database directory in which your JADE database files are located. This parameter must be specified and it must be a valid JADE database path. The database path must exist; for example:

```
d:\jade\system
```

ini

The **ini** parameter 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 **jade.ini**.

For details, see "Location of the JADE Initialization File", in the *JADE Initialization File Reference*.

server

Specify the optional **server** parameter only if you want to run the application server in single user mode. If you do not specify the **server** parameter, the application server runs in multiuser mode.

name

The optional **name** parameter indicates that the **jadappb** program checks for a matching JADE initialization file section name.

For details, see "Two-Level Section Names" and "Sharing JADE Initialization Files", in the *JADE Initialization File Reference*.

host

The optional **host** parameter specifies the valid host server node name or host IP address.

port

The optional **port** parameter specifies the unique valid port number or port name of the host (server) node.

interface

The optional **interface** parameter specifies the TCP/IP name or the IP address of the client (local) node.

localport

The optional **localport** parameter specifies the port number or port name on the client (local) node.

Chapter 3 Administering JADE Thin Client Environments

This chapter covers the following topics.

Thin Client Connection Balancing

Thin client connection balancing allows a group of application servers to evenly share presentation client connections between those servers, regardless of the application server initially configured for each presentation client.

Notes Connection balancing balances incoming connections for GUI-type applications only.

Do not confuse thin client connection balancing with *load* balancing. Thin client connection balancing is strictly limited to ensuring that the number of connections to each application server in the group is approximately the same; no attempt is made to share any other resource usage load between the application servers.

A database environment can have multiple application server groups.

All application servers in a group must be equally visible to each presentation client that connects to members of that group.

Only the first connection from a presentation client can be redirected to another application server. A presentation client always directs subsequent connections for additional applications or reconnections to the application server that actually completed the first connection.

Tip A system administrator can take advantage of this feature without having to reconfigure existing presentation clients.

Overview

To participate in connection balancing, application servers register themselves as part of an application server group with the database server, supplying the appropriate value of the **AppServerGroupName** parameter in the [JadeAppServer] section of the JADE initialization file and other configuration details; for example, the value for the **MaxLocalProcesses** parameter in the [JadeClient] section of the JADE initialization file, the listen address, and the port number.

Presentation clients make an initial connection to the application server defined on the command line or in the JADE initialization file in exactly the same way as when connection balancing is not used. When the connection has opened, the presentation client sends a redirect-allowed initial message.

When the application server has accepted the connection and received the initial message, if it does not have connection balancing enabled or the message disallows redirection, the application server assumes control of the presentation client. If the value of the **MaxLocalProcesses** parameter or the process license count is exceeded, the application server returns a "connection rejected" message to the presentation client.

If the application server can perform connection balancing, it asks the database server for the list of application servers that share the same value of the **AppServerGroupName** parameter. The returned list includes the current connection and process counts for each of the application servers in the group. The initial application server examines the list and determines which application server in the group can accept more connections and has the fewest current connections. If the selected application server matches the initial application server, connection handling proceeds as normal and the initial application server assumes control of the presentation client. If they are different, the initial application server returns a "redirect" reply to the presentation client, including the address and port of the selected application server. If no application server is selected, the initial application server returns a "connection rejected" message to the presentation client.

If a presentation client uses an SSL/TLS encrypted connection to the initial application server, only those application servers in the group that registered as SSL/TLS supporters are considered as potential redirection targets.

When a presentation client receives a "redirect" reply, it closes its connection to the initial application server and re-opens the connection using the address and port included in the "redirect" reply. The presentation client saves the new address and port for reconnections and secondary application connections. The presentation client sends an initial message with redirection disallowed to the new application server. It is possible that the presentation client can receive a "connection rejected" response from the selected application server.

The application server checks if a download is required after connection balancing has been performed. This ensures that all application servers in a group share the burden of downloading presentation client upgrades.

The presentation client logs redirect replies that it receives to its log file. If a presentation client uses an SSL/TLS-encrypted connection to the initial application server, it is redirected only to application servers that support SLL/TLS-encrypted connections.

Configuration

This section contains the JADE initialization file parameter and application server executable (**jadapp** or **jadappb**) command line argument requirements.

JADE Initialization File Parameters

This section contains the JADE initialization file parameter requirements for thin client connection balancing.

AppServerGroupName Parameter

The **AppServerGroupName** parameter in the [JadeAppServer] section of the JADE initialization file defines the name of the group to which the application server belongs. The value of this parameter is a case-sensitive user-defined string of up to 30 characters.

If you do not specify this value, thin client connection balancing cannot be enabled. All application servers in the same group must have the identical group name.

ExternalAppServerAndPort Parameter

The **ExternalAppServerAndPort** parameter in the [JadeAppServer] section of the JADE initialization file is optional.

You must specify this parameter if the application server is:

- Not listening on a specific address. This is usually indicated when the value of the **AppServer** parameter is not specified or is **"0.0.0.0"**.
- Listening on an address and port which is hidden from the workstation or device running the presentation client by Network Address Translation (NAT). All presentation clients redirected to this application server must be subjected to the same NAT rule; that is, only one override address and port is provided.

You cannot enable thin client connection balancing if the value of this parameter is invalid; for example, it has invalid characters or the port number is not specified.

If the value includes a Fully Qualified Domain Name (FQDN), it must uniquely identify the machine on which the application or server is running. It must not resolve to multiple addresses that point to other machines (otherwise thin client connection balancing is defeated).

AppServer Parameter

Thin client connection balancing cannot be enabled if the **ExternalAppServerAndPort** parameter or its value is not specified, and the **AppServer** parameter in the [JadeAppServer] section of the JADE initialization file or its value is not specified or the value is the default value of **0.0.0.0**.

If the value of the **AppServer** parameter is an FQDN, it must uniquely identify the machine on which the application server is running. It must not resolve to multiple addresses that point to other machines (otherwise thin client connection balancing is defeated).

SSLSecurePort Parameter

If the application server is configured to use SSL/TLS encryption for presentation client connections, the listen port is taken from the value of the **SSLSecurePort** parameter in the [JadeAppServer] section of the JADE initialization file.

The effective default value of this parameter is **443**.

EnableAppRestrictions, *EnableRootSchemaAppRestrictions*, and *AllowSchemaAndApp* Parameters

To avoid unexpected presentation client rejections, all application servers sharing the same group must use the same values for the **EnableAppRestrictions**, **EnableRootSchemaAppRestrictions**, and **AllowSchemaAndApp** parameters in the [JadeAppServer] section of the JADE initialization file.

The system administrator is responsible for ensuring that configuration of the application servers in a group is consistent.

Command Line Arguments for the jadapp and jadappb Programs

This section contains the **jadapp** and **jadappb** program command line argument requirements for thin client connection balancing.

appServer Argument

The value of the **appServer** command line argument of the **jadapp** and **jadappb** program overrides the value of the **AppServer** parameter in the [JadeAppServer] section of the JADE initialization file.

appServerPort Argument

The value of the **appServerPort** command line argument overrides the value of the **AppServerPort** parameter in the [JadeAppServer] section of the JADE initialization file.

Note If the **ExternalAppServerAndPort** parameter is not specified in the [JadeAppServer] section of the JADE initialization file, the actual values used for the listen connection address and port number are used for redirection, regardless of how they are specified.

Enabling Thin Client Connection Balancing

Thin client connection balancing is enabled when at least one JADE process running in the application server successfully invokes the **Application** class **enableThinClientConnBalancing** method. A process on each application server involved in thin client connection balancing must call the **enableThinClientConnBalancing** method.

This method raises exceptions to report configuration errors; for example, a value that is not specified for the **AppServerGroupName** or **AppServer** parameter in the [JadeAppServer] section of the JADE initialization file. Additional information is output to the **jommsg.log** file.

The **enableThinClientConnBalancing** method registers the application server in which it is invoked with the database server, and marks the calling process as a redirection assistant. You can call this method more than once in each application server and by multiple processes.

Tip We recommend that you call this method in the initialization logic of an application that is initiated by using the **ServerApplication** parameter in the [JadeAppServer] section of the JADE initialization file.

The **enableThinClientConnBalancing** method logs that connection balancing has been enabled.

If all JADE processes marked as redirection assistants in a node terminate, the application server stops redirecting presentation clients to other application servers, because it uses those JADE processes to retrieve the application server list from the database server and to select an entry from the returned list.

Tracing Presentation Client Requests and Messages

When the **Trace** parameter in the [JadeThinClient] section of the presentation client initialization file is set to **all** or **messages**, requests made to and from that presentation client are traced to the presentation client directory specified by the **TraceDirectory** parameter in the [JadeThinClient] section of the JADE initialization file.

In addition, when you set the **TraceAppServer** parameter in the [JadeThinClient] section of the JADE initialization file to **true**, a thin client trace file is generated on the application server for each application initiated by the user. (The **TraceDirectory** parameter in the [JadeAppServer] section of the JADE initialization file specifies the application server directory for requests made to and from presentation clients.)

For presentation client tracing, the **TraceAppServer** parameter is independent of the value of the **Trace** parameter. The presentation client splash screen displays *Tracing is On* if the **TraceAppServer** parameter is set to **true** or the **Trace** parameter is set to **all** or **messages**.

A separate trace file, opened for each application that is initiated, has the following format.

```
ThinClientTrace-<user-name>-<date>-<time>.log
```

Tip Although a generated trace file on the application server contains the same information as that of a presentation client trace file, the entries in the trace file on the application server provide actual time gap information between each GUI logic statement that is called. You can therefore use them as an indication as to where performance issues might be. Comparing the file with a presentation client trace file also provides information about network performance issues.

The text **Tracing is ON** is displayed directly under the start-up status line on the splash screen when tracing is on.

When all requests made to and from the presentation client are traced (that is, the **Trace** parameter is set to **all**), the following trace information for each request to or from the presentation client is output to the trace file (with the default JADE directory of **logs** for both presentation clients and the application server).

1. Time of the message or request.
2. Message or request number.

   Lines of output that do not have a message number indicate that the request is buffered with the previous message. Buffered requests are traced only when the **Trace** parameter is set to **all**; that is, these requests are not output when the parameter is set to **messages** (which outputs only the commands that caused a message to be sent).
3. Time (in milliseconds) since the last message or request.
4. The name of the schema and application being initiated.
5. Direction in which the message or request was sent, represented by paired **>>** and **<<** symbols. The **>>** symbols indicate a message or request from the presentation client to the application server and the **<<** symbols indicate a message or request from the application server to the presentation client.
6. The processing depth (or stack) of the message or request (for example, **#=1** or **#=2**).
7. Direction of the message or request, represented by paired **Send** and **End** statements or **Received** and **End** statements.

   Send indicates a message or request from the presentation client to the application server and **End** indicates the completion of that request. **Received** indicates a message sent from the application server to the presentation client and **End** indicates the completion of that request. (These equate to the **>>** and **<<** symbols and the **<<** and **>>** symbols, respectively.)
8. Length (measured in bytes) of the message or request; for example, **(len=3521)**.

9. Description (or reason) of the message or request (for example, the name of the control and the form on which it is defined followed by the . dot notation and the GUI attribute that was accessed).

When the **Trace** parameter is set to **messages**, buffered requests are not output; that is, only commands that caused a message to be sent between the presentation client and the application server are traced.

Note You can view the trace output file by using Notepad or a Windows editor. (The output file is cumulative; that is, records are appended to any existing records in the file.)

The following is an example of traced output to the log file when the **Trace** parameter was set to **all**.

```
******* Thin Client trace file (version 1) opened: 23 November 2005, 13:28:45
  legend: msg=  is message transmission number
          time= is milli-seconds since last msg
          #=    is processing depth
13:28:45 msg=1 time=310  >> #=1 Send (len=15) Check Software Version
13:28:45 msg=2 time=10   << #=1 End send (len=8)
13:28:45 msg=3 time=0    >> #=1 Send (len=3521) Sign on db
13:28:46 msg=4 time=992  <<     #=1 Received (len=10) printer.create
13:28:46 msg=5 time=0    >>     #=1 End receive (len=12)
13:28:46 msg=6 time=120  <<     #=1 Received (len=843) app.initializePart 1
13:28:46       time=0    << #=1 End send (len=8)
13:28:46 msg=7 time=0     >> #=1 Send (len=11) Run the app
13:28:47 msg=8 time=621   <<     #=1 Received (len=28149) form.create
13:28:47 msg=9 time=60    >>     #=1 End receive (len=480)
13:28:47 msg=10 time=100  <<     #=1 Received (len=10) Start-upFm.load called
13:28:47        time=0    <<     #=1 Received (len=53) external function call
13:28:47 msg=11 time=0    >>     #=1 End receive (len=44)
13:28:47 msg=12 time=100  <<     #=1 Received (len=35) write
13:28:47        time=0   <<     #=1 Received (len=11)
                          Start-upFm.mousePointer
13:28:47        time=0    <<     #=1 Received (len=899) resizetest.bubbleHelp
13:28:47        time=0    <<     #=1 Received (len=10) Start-upFm.clear
13:28:47        time=0    <<     #=1 Received (len=52)
                          progressBar1.drawSolidRectangle
13:28:47        time=0    <<     #=1 Received (len=45)
                          progressBar1.drawSolidRectangle
13:28:47        time=0    <<     #=1 Received (len=53)
                          progressBar1.drawTextIn
13:28:47        time=0    <<     #=1 Received (len=11) Start-upFm.show
13:28:47        time=0    >>     #=2 Send (len=12) Notify property change
13:28:47        time=0    >>     #=2 Send (len=15) Notify property change
13:28:47 msg=13 time=0    >>     #=1 End receive (len=8)
13:28:47 msg=14 time=0    << #=1 End send (len=8)
13:28:47        time=10   >> #=1 Send (len=21) Event:paint
13:28:47 msg=15 time=70   >> #=1 Send (len=11) Send queued requests
13:28:47 msg=16 time=0    <<     #=1 Received (len=52)
                          progressBar1.drawSolidRectangle
13:28:47        time=0    <<     #=1 Received (len=45)
                          progressBar1.drawSolidRectangle
13:28:47        time=0    <<     #=1 Received (len=53)
                          progressBar1.drawTextIn
13:28:47        time=0    << #=1 End send (len=8)
13:28:48        time=111  >> #=1 Send (len=12) Notify property change
13:28:48 msg=17 time=20   >> #=1 Send (len=11) Send queued requests
13:28:48 msg=18 time=0    << #=1 End send (len=8)
```

```
13:29:44         time=56050 >> #=1 Send (len=12) Notify property change
13:29:44         time=0     >> #=1 Send (len=15) Notify property change
13:29:44 msg=19 time=0     >> #=1 Send (len=49) Event:mouseDown
13:29:44 msg=20 time=30    << #=1 End send (len=8)
13:29:44 msg=21 time=0     >> #=1 Send (len=24) Event:gotFocus
13:29:44 msg=22 time=10    <<     #=1 Received (len=32) write
13:29:44         time=0     << #=1 End send (len=8)
13:29:44 msg=23 time=40    >> #=1 Send (len=21) Event:click
13:29:44 msg=24 time=0     << #=1 End send (len=8)
13:29:45 msg=25 time=1072 >> #=1 Send (len=28) Event:queryUnload
13:29:45 msg=26 time=0     << #=1 End send (len=12)
13:29:45 msg=27 time=0     >> #=1 Send (len=18) Event:unload
13:29:45 msg=28 time=10    << #=1 End send (len=8)
13:29:45         time=10    >> #=1 Send (len=11) Destroy window
13:29:45 msg=29 time=0     >> #=1 Send (len=11) Shutdown request
13:29:45 msg=30 time=10    << #=1 End send (len=8)
13:29:45 msg=31 time=0     >> #=1 Send (len=11) Call finalize
13:29:45 msg=32 time=0     <<     #=1 Received (len=18) printer.close
13:29:45 msg=33 time=0     >>     #=1 End receive (len=12)
13:29:45 msg=34 time=10    << #=1 End send (len=8)
13:29:45 msg=35 time=0     >> #=1 Send (len=11) Sign off db
13:29:45 msg=36 time=80    <<     #=1 Received (len=10) closeOutputWindow
13:29:45         time=10    << #=1 End send (len=8)
```

Enabling JADE Thin Client Security Encryption

The JADE thin client mode provides optional encryption support for the presentation client and application server communication.

The [JadeAppServer] and [JadeThinClient] sections of the JADE initialization file contain the **RPCEncryptionEnabled** and **RPCEncryptionHookDLL** parameters.

- If the **RPCEncryptionEnabled** parameter in the [JadeAppServer] section is set to **true**, the defined encryption is enforced on all presentation clients attached to that application server. (For details about the valid combinations of the **RPCEncryptionEnabled** parameter and **RPCEncryptionHookDLL** parameter on both the application server and presentation clients, see the JADE application server **RPCEncryptionEnabled** parameter under "Application Server Section [JadeAppServer]", in the *JADE Initialization File Reference*.)
- The **RPCEncryptionHookDLL** parameter identifies the JADE or user-supplied encryption library. The available types of encryption are:
  - **RPCEncryptionHookDLL=Internal**, which uses Windows-supplied 40-bit encryption.
  - **RPCEncryptionHookDLL=SSL_TLS**, which enables the SSL security feature. When SSL security is enabled, the parameters whose names start with **SSL** (described in following subsections) are used. For details, see "Secure Sockets Layer (SSL) Security", in Chapter 2 of the *JADE Object Manager Guide*.
  - **RPCEncryptionHookDLL=<*name*>**, which calls a user-supplied library with the name specified in the **<*name*>** value. (For details, see "Network Message Encryption" under "JADE Security", in Chapter 2 of the *JADE Object Manager Guide*.)

Note The RPC encryption library must be thread-safe; that is, it must be able to handle multiple threads calling this library simultaneously.

For details, see "Application Server Section [JadeAppServer]" and "JADE Presentation Client Section [JadeThinClient]" under "JADE Thin Client Sections", in the *JADE Initialization File Reference*. See also "Controlling JADE Thin Client Application Execution", in the following section.

For details about returning the type of encryption being used by a JADE thin client TCP/IP connection in the current application, see the **Application** class **getThinClientEncryptionType** method.

Transport Layer and Secure Socket Layer Security

If you want Payment Card Industry (PCI) compliance or to protect against potential Denial of Service (DoS) attack, set the **SSLPermitClientRenegotiation** parameter in the [JadeAppServer] section of the JADE initialization file to **false** so that any client-initiated renegotiation will cause the network connection to be dropped.

The default value of **true** indicates that if an application server is public-facing on the Internet, for example, this will help protect against Denial of Service (DoS) attacks. Payment Card Industry (PCI) compliance checks can require that measures are present to prevent vulnerability attacks.

A message is added to log the build version of OpenSSL and the version of the OpenSSL DLLs used (that is, **ssleay32.dll** and **libeay32.dll**).

Note To support secure client renegotiations, you require a minimum version of 1.0.2g of the OpenSSL libraries.

Controlling JADE Thin Client Application Execution

You can control the applications that can be executed from any presentation client attached to an application server. The [JadeAppServer] section of the JADE initialization file contains the **EnableAppRestrictions** and **AllowSchemaAndApp<*n*>** parameters.

When the **EnableAppRestrictions** parameter is set to **false** (the default value), all applications can be executed by presentation clients. If this parameter is set to **true**, only the schema and optional applications specified in the **AllowSchemaAndApp<*n*>** parameters can be executed.

In addition, the [JadeAppServer] section provides the **KeepAliveTime** parameter, which enables you to specify the number of minutes that each running application checks whether the connection to the presentation client is still valid by sending a *'keep alive'* message to the client, after every period that is specified by this parameter expires without any activity. When the application server detects that a connection has been lost, it reads the **ReconnectWaitTime** parameter in the [JadeAppServer] section of the JADE initialization file, which specifies a value in seconds and which defaults to zero (**0**), indicating that the value of the **KeepAliveTime** parameter is used. The application server then waits for a reconnection for the specified period before terminating the process.

When the **EnableAppRestrictions** parameter is set to **true** and the application being started is not defined in the **AllowSchemaAndApp<*n*>** parameter, the following message is displayed when attempting to start the application from the JADE development environment.

```
Application application-name cannot be started from the connected application
server
```

Note When an application server recognizes that a connection has been lost, it does not know whether the presentation client has also realized. As a result, it waits for the number of minutes specified by the **KeepAliveTime** parameter, to give the presentation client the chance to reconnect. In the worst case situation, the delay before the application server disconnects the presentation client can be twice the length of the keep-alive time.

By default, the application server performs this check after every ten minutes of inactivity. For details, see "Application Server Section [JadeAppServer]" under "JADE Thin Client Sections", in the *JADE Initialization File Reference*.

Timing Out a JADE Thin Client Connection

Use the **ConnectionTimeout** parameter of the JADE initialization file [JadeAppServer] section to specify the maximum number of minutes that the application server waits for activity (messages sent by or received) from any application attached to the server before the application session times out.

Notification and timer traffic is ignored when determining if the connection has timed out. The default value of zero (**0**) indicates that there is no timeout. If no activity occurs within the specified time, the application terminates and the following message is displayed in a message box on the presentation client.

```
The Current Application Session timed out
```

In addition, if the **EnableAppRestrictions** parameter in the [JadeAppServer] section is set to **true**, you can use the **AllowSchemaAndApp*<n>*** parameter to specify a timeout period for a specific schema and optional application, by using the following parameter syntax.

```
AllowSchemaAndApp<number> = schema-name [, application-name], timeout-value
```

The following example shows the timeout value in the **MyApp** application in the **MySchema** schema set to 30 minutes.

```
[JadeAppServer]
ConnectionTimeout     = 0
EnableAppRestrictions = true
AllowSchemaAndApp1    = JadeSchema
AllowSchemaAndApp2    = MySchema, MyApp, 30
```

This example allows all **JadeSchema** applications to be run without any timeout and the **MyApp** application in the **MySchema** schema to time out if no activity occurs within 30 minutes.

You can set the **app.setInactiveTimeoutPeriod** method, which results in the **app.inactiveTimeout** method being called when the inactive period has expired. This then allows your user logic to perform whatever is required, including a terminate, which allows a graceful exit.

Notes When an application times out (that is, the period specified in the **ConnectionTimeout** parameter of the JADE initialization file [JadeAppServer] section elapses without activity), the equivalent of a monitor force off is performed, closing all forms and signing off without calling any methods.

The application server issues a test to determine if each thin client connection is still valid after 10 minutes of inactivity. If the connection is not available, the application server terminates that connection. (See also the note under "Controlling JADE Thin Client Application Execution", in the previous section.)

Attempting to Reconnect following a TCP/IP Connection Failure

The [JadeAppServer] and [JadeThinClient] sections of the JADE initialization file contain the **AttemptReconnect** parameter, which specifies whether presentation clients attempt to reconnect to the application server when a TCP/IP connection failure occurs. By default, this parameter is set to **true** on both the application server and presentation clients.

Set the **AttemptReconnect** parameter to **false** in the [JadeThinClient] section on a presentation client if you want the application server to generate an exception when a TCP error is detected and then terminate the presentation client.

When the **AttemptReconnect** parameter is set to **true** in both JADE initialization file sections on both the application server and presentation client, use the **ReconnectTimeout** parameter in the [JadeThinClient] section of the JADE initialization file to specify the number of seconds that the presentation client continues to attempt to reconnect to the application server. (By default, reconnection is attempted for **40** seconds.)

The reconnection form displays the application server and application to which reconnection is being attempted.

For details about the actions that you can perform when a disconnect event occurs (for example, so that you can programmatically retry the connection or programmatically close it), see "Handling the Loss of Thin Client Connections", in Chapter 2.

Compressing Transmitted Data

Set the **UseCompression** parameter of the JADE initialization file [JadeThinClient] section to **true** if you want to specify that the data sent to and from the application server is compressed. (By default, data transmitted to or from the application server is not compressed.) The compression ratio that is achieved is appended to the *"Thin Client disconnected and signed off"* message written to the **jommsg.log** file of the application server when an application closes.

The compression process is transparent to both you and the user, and enables you to reduce the size of data that is sent over a slow network connection at the cost of the extra processing CPU required to compress and uncompress the data.

File Read Optimization

The overheads associated with reading a file from the presentation client is reduced by lowering the number of messages sent between the application server and the presentation client if the file is "static"; that is, the file is opened for input only (**mode = Mode_Input**) with the **File** class **shareMode** property set to **Share_Read** or **Share_Exclusive**.

Under those conditions, the file is read by the application server in chunks from the presentation client and the file buffer management is then performed by the application server so that performance is not affected by whether the **File** class **readLine**, **readString**, or **readBinary** method is used to access the file.

As files are opened as **Share_ReadWrite** by default, to achieve this performance boost your logic must set the required **shareMode** property value for the file. This applies to the processing of schema (.**scm**) and form and data definition (.**ddb** or .**ddx**) file loads, resulting in a dramatic reduction of the number of messages required to process the files when running in thin client mode. A 200-to-1 reduction has been seen on a large form and data definition file, which is due to the way the .**ddb** or .**ddx** file is read, with **continual** peek requests requiring reads, seek repositioning, and re-reading.

Note This optimization is not used when reading UTF8 files, non-native files (that is, Unicode files on an ANSI system or ANSI files on a Unicode system) or if the value of the **File** class **maxRecordSize** property is greater than 4K bytes.

The maximum size of data that can be read or written by each read or write type statement when the value of the **TcpIpConnection** class **usePresentationClient** property is **true** is 2G bytes. If the data exceeds this limit or the length is less than zero (**0**), exception 5047 *(Invalid record size)* is raised.

File Handling

To ensure that any exception is returned against the correct file object, the following handling occurs for file write requests to the presentation client.

1. The initial write request performed after the file is opened is forced to the presentation client and the result is returned. (This covers the situation where the initial write fails due to environmental access conditions and it forms the majority of write failures.)
2. Out-of-disk situations present the user with a message box, advising that the write failed and requesting that space be made available. The user can then click the **Retry** button to repeat the original operation without loss of data and without generating an exception.
3. If a write error occurs for any other situation or if the out-of-disk request is cancelled, the failure is reported against the next non-buffered request performed on that file, to ensure that the correct object is reported against in the generated exception.

Presentation Clients Running in 64-Bit Mode

Presentation clients can run in 32-bit or 64-bit mode.

When running presentation clients in 64-bit mode, you can use the properties and method summarized in the following subsections. (See also "External Function Calls", in Chapter 1.)

CMDFont::*printerDC64* Property

The **CMDFont** class **printerDC64** property contains a 64-bit printer device context handle to the font common dialog. If this property is set, the **CMDFont** class **printerDC** property is ignored.

Note A value set in the **CMDFont** class **printerDC** property is truncated to a 32-bit integer, and may not function as required in a 64-bit environment.

CMDPrint::*printerDC64* Property

The **CMDPrint** class **printerDC64** property contains the printer device context handle when the **returnDC** property specifies that requirement.

The **CMDPrint** class **printerDC** property also contains the printer device context handle if it is not larger than a 32-bit integer; otherwise the **printerDC** property is set to zero (**0**).

Window::*getHwnd* Method

The **Window** class **getHwnd** method returns the Microsoft Windows handle for a form or control.

Call the **getHwnd** method instead of the **Window** class **hwnd** method when presentation clients are running in a 64-bit environment.

Although these methods return the same value if the value is a 32-bit integer, if you call the **hwnd** method and the Window handle is larger than a 32-bit integer, an exception is raised.

Enhancing the Performance of Windows and Forms

JADE provides the following features that enable you to enhance the performance of windows and forms when running applications in JADE thin client mode.

- Controlling Cells in Tables
- Handling Text Box Data Entry
- TextBox::firstChange Event
- Caching Pictures and Forms
- Creating Pictures
- Flushing Commands Queued in the Application Server
- Registering Form, Edit Mask, and Text Box Keys
- Updating the Progress Bar

For details, see the following sections.

Controlling Cells in Tables

When the **automaticCellControl** property is set to the default value of **false**, the **cellInputReady** event must set the text box value from the cell text if a text box is assigned to a table cell and the **TextBox** class **change** event must then set the cell text as it is changed.

As an event has to be processed for each keystroke, this impacts on the performance of JADE thin clients running over slow links.

You can therefore set the **Control**::**automaticCellControl** property of the cell control to **true** if you want the **Table** class to manage the **cellControl** property automatically, as follows.

- For a text box, the table loads the text box with the value of the current cell. When the text box is changed, the table cell is updated and the **Table** class **change** event is called. All events are still called if they are defined, but they are not generally needed.
- For a combo box, the table sets the combo box **listIndex** property using the text of the current cell. If a new combo box entry is selected, the table cell text is updated and the **Table** class **change** event is called.

Note When the **automaticCellControl** property of the cell control is set to **true**, arrow keys can be used to step to other cells for a text box or combo box in a table. If the caret is at the beginning of the text, the left arrow key steps to the prior cell. If the caret is on the top line, the up arrow key steps to the prior row. If the caret is at the end of the text, the right arrow key steps to the next row. If the caret is on the last line of the text, the down arrow key steps to the next row.

In addition, pressing the Alt key and an arrow key performs the requested action, regardless of where the caret is placed.

The **automaticCellControl** property is currently ignored for any control type other than a combo box or text box. In addition, the **automaticCellControl** property has no meaning unless the control is assigned to a **Table** control as a **cellControl** property. For more details, see Chapter 2 of the *JADE Encyclopaedia of Classes*.

Handling Text Box Data Entry

Data entry into a text box control can suffer from keyboard entry 'lag behind' which can result from having to process too many key event methods on the form and text box and from the increasing size of the text value.

As the text is changed, that text is transmitted so that it is available for any logic reference.

If there are no form or text box key event methods, the text is transmitted only when some other event occurred (for example, the **lostFocus** event).

Tip Use the **JadeEditMask** control to improve thin client performance by removing validation requirements. User-written logic that performs input validation keystroke by keystroke has a dramatic impact on thin client performance over a slow link, making such validation impractical, the **JadeEditMask** control reduces the need for you to write such logic and therefore speed up thin client performance. (For details, see "JadeEditMask Class", in Chapter 2 of the *JADE Encyclopaedia of Classes*.)

In addition to the **TextBox**::**firstChange** event described in the following subsection, the **Form** and **Window** classes provide the methods listed in the following table that enable you to improve the performance of data entry in JADE thin client mode over slow network links.

Class	Method	Description
Window	enableEvent	Enables you to control at run time whether JADE logic associated with an event for a specific form or control is executed. You could use this method in thin client mode, for example, to speed up the data entry process for a text box control by disabling the **keyDown** event.
Form	getRegisteredFormKeys	Populates an array of the form keys that are in effect for the form. This array contains entries only if the **Form**::**registerFormKeys** method has been called.
Window	isEventEnabled	Specifies whether an event is currently enabled for the window.
Form	registerFormKeys	Establishes the entire set of key codes in which key events of the receiver form are interested. After calling the **registerFormKeys** method with a key code list, the form key events are called on the form only if the key that is pressed is in the supplied list.

For details, see Chapter 2 of the *JADE Encyclopaedia of Classes*.

TextBox::*firstChange* Event

The **firstChange** event of the **TextBox** control is generated when the user performs keyboard or cut and paste actions that result in any of the following conditions.

- The text box goes from an empty to a non-empty state
- The text box goes from a non-empty to an empty state
- The first change made after the text has been set by the form build process or by JADE logic

Use the **firstChange** event to move logic from other key events when that logic is relevant only to the first time the text changes; for example, when the logic is recording that the text content has changed. As a result, removal of the other key events reduces the number of events that must be sent and processed for each key that is pressed.

Note The **firstChange** event for the **TextBox**, **JadeEditMask**, **JadeRichText**, and **Form** classes is not available from a Web browser.

Caching Pictures and Forms

The **Picture**::**cachePictures** property has meaning only when the picture object is attached to a physical window.

By default, setting the picture properties of a **Picture** or **JadeMask** control at run time causes those pictures to be stored in the cache file of the application server and the cache file of the presentation client unless the **UseCacheFile** parameter in the [JadeThinClient] section of the JADE initialization file is set to **false**.

However, you can set this property to **false** in situations where you do not require this caching because the picture is only ever downloaded once and will therefore unnecessarily add entries to those cache files. Examples of where you could set this property to **false** are as follows.

- Using the **Control**::**createPicture** method to create a dynamic runtime image that is then displayed. (See "Creating Pictures", in the following subsection, for more details about picture creation.)
- Displaying a customer invoice that was scanned in.

Setting the **cachePictures** property to **false** affects the following.

- **Picture** class **picture**, **pictureDown**, and **pictureDisabled** properties
- **Picture**::**setPicture** method
- **JadeMask** class **pictureRollOver**, **pictureRollUnder**, and **pictureMask** properties

In addition, you can use the **CacheEntryTimeout** parameter in the [JadeAppServer] section or the **CacheEntryTimeout** parameter in the [JadeThinClient] section of the JADE initialization file to control the number of days entries are held in the cache file on the application server or presentation client before being discarded. This parameter provides control over cache file entries that become orphaned when forms and pictures are removed from the JADE system.

By default, a form that has not been created or a picture that has not been displayed within the last 30 days is discarded. You can set this parameter to zero (**0**) if you want forms and pictures to remain indefinitely in the cache file on the presentation client. In addition, the **Window** class **drawPictureAt** and **drawPictureIn** methods cache the pictures that are drawn on the presentation client.

Creating Pictures

Instances of the **Control** class provide the **createPictureIndirect** method, which creates a short binary that contains an instruction concerning the window to be copied.

When the binary is assigned to a **Picture** control property, the picture is created using the current image of the requested window.

Note The **createPictureIndirect** method improves JADE thin client mode performance, as it does not have to pass large binary large objects (blobs) back and forth between the presentation client and the application server when creating a bitmap for the receiver control.

If you use the **Control**::**createPicture** method in JADE thin client mode, the picture image would be created on the presentation client and brought back to the application server. Assigning the picture then causes the image to be sent back to the presentation client and to be cached, which could involve significant delays and overheads for a reasonable-size image. Using the **createPictureIndirect** method, a small binary is sent instead and the image data itself is not transported between the presentation client and the application server.

For more details, see Chapter 2 of the *JADE Encyclopaedia of Classes*.

Flushing Commands Queued in the Application Server

Use the **Application**::**flushThinClient** method to force visual changes to be applied at some logical point before the current processing is complete (for example, a progress bar to update its display at a logical point). Calling this command causes any commands queued in the application server for the JADE thin application to be passed immediately to the presentation client for processing.

Queued commands are normally passed to the presentation client when the current Graphical User Interface (GUI) process that generated the processing is complete or when access to a method, property, or so on, can be satisfied only by a call to the presentation client waiting for the return.

Note The **flushThinClient** method is ignored if the client node is not a JADE thin client presentation client or if there are no queued commands on the application server.

Registering Form, Edit Mask, and Text Box Keys

The **Form**, **JadeEditMask**, and **TextBox** classes provide the methods listed in the following table, which enable you to register keys that are in effect for the form, edit mask, or text box.

Method	Description
Form::getRegisteredFormKeys	Returns an array of the form keys that are in effect for the form
Form::registerFormKeys	Establishes the entire set of key codes in which the key events of a form are interested
JadeEditMask::getRegisteredKeys	Returns an array of the keys that are in effect for the edit mask
JadeEditMask::registerKeys	Establishes the entire set of key codes in which the key events of an edit mask are interested
TextBox::getRegisteredKeys	Returns an array of the keys that are in effect for the text box
TextBox::registerKeys	Establishes the entire set of key codes in which the key events of a text box are interested

For details, see Chapter 2 of the *JADE Encyclopaedia of Classes*.

By default, if a form, edit mask, or text box has key event methods defined, those key events are sent for any key event for any control on that form or for any edit mask or text box. In most situations, the key events are interested only in specific keys (for example, the Tab key, arrow keys, and function keys).

After calling the **Form**::**registerFormKeys**, **JadeEditMask**::**registerKeys**, or **TextBox**::**registerKeys** method with a key code list, the key events are called on the form, edit mask, or text box only if the key that is pressed is in the supplied list. This results in not having to call the form, edit mask, or text box key events for every key action, which reduces the number of events that must be sent and processed.

Updating the Progress Bar

Use the **thinClientUpdateInterval** property of the **ProgressBar** control to specify (in milliseconds) how often the progress bar is redrawn when the percentage changes when running the application in thin client mode. The default value of **1000** milliseconds (1 second) specifies that the control is updated only when the percentage changes and at least one second has elapsed since it was last updated or the 100 percent mark has been reached.

If you set the value of the **thinClientUpdateInterval** property to zero (**0**) when the application is running in thin client mode, the progress bar is updated for every required percentage change specified by the **partsDone** property.

Tip This automatic thin client optimization prevents an unnecessary number of messages being sent over the TCP connection when the progress updates at a fast rate.

Appendix A JADE Thin Client Considerations and Restrictions

This appendix covers the following topics.

JADE Functionality Affected by Thin Client Operation

When you run JADE in thin client mode, you should be aware of the following functionality.

- If a large amount of data must be transmitted to the presentation client (for example, the initial download of a form containing a bitmap), a progress bar is displayed on the presentation client workstation if it is determined that the download time will be significant.
- The **read** and **write** instructions are always executed on the presentation client workstation.
- All printer functions are performed by the presentation client and connect to printers local to that presentation client workstation.
- The **Printer**::**getAllPrinters** method returns printers:
  - Local to the presentation client only, when executed on a presentation client
  - Attached to that server, when called from a server method (which enables you to get a list of valid printers for the application server if you want to schedule report jobs for a reporting presentation client that runs on the application server)
- The **Printer**::**setPrinter** method sets the local printer environment to the requested environment.

  When you use the **=pagenofm** or **=totalpages** option of the **Label** or **TextBox** class **formatOut** property for formats of data in text boxes or labels and the report is being stored in the database (that is, the report uses the **Printer**::**setReport** method), output is retrieved from a temporary file and stored in the database only after the printer is closed. This is most evident when running in JADE thin client mode, as the printed output must be retrieved from the presentation client and passed to the application server at the end of the report rather than page by page as the report is produced. (For details, see the **Label** or **TextBox** class **formatOut** property in Chapter 2 of the *JADE Encyclopaedia of Classes*.)
- All common dialogs execute on the presentation client and return information relative to the presentation client.
- The **Application**::**msgBox** method executes on the presentation client.
- The **Application**::**userName** method returns the name of the user on the presentation client.
- The **Application**::**computerName** method returns the name of the presentation client.
- By default, all locale information is based on the locale of the presentation client that initiated the application. Only the options defined by the application server for that locale apply.

  Set the **EnhancedLocaleSupport** parameter in the [JadeEnvironment] section of the JADE initialization file to **true** if you want regional overrides set on the presentation client to be forwarded to the application server so that both use consistent locale settings for the application. The regional overrides set on the presentation client are forwarded to the application server so that both use consistent locale settings for the application.

- When using the **Connection** class, all connections are made to the workstation that is running the JADE logic (that is, to the application server).
- When a connection is lost, the initial connection error is reported. If logic attempts to subsequently communicate with the presentation client, the application is automatically terminated (that is, **APPLICATION_TERMINATE_REQUEST** is returned for any communication attempt). This results in all outstanding methods being aborted and the application is then signed off. The **finalize** method of the application is not run.

  When reconnection is attempted, the reconnection form displays the application server and application to which reconnection is being attempted.
- Profiler output is always output to the workstation that is running the JADE logic (that is, to the application server).
- The **getOSPlatform** method of the **Process** class returns the operating platform of the presentation client.
- Methods of the **File** and **FileFolder** classes and the **loadFromFile** method of the **Sound** class are processed on the presentation client when the **FileNode** or **MultiMediaType** class **usePresentationFileSystem** property is set to **true** and the instances are *not* shared transient instances.

  Shared transient instances of the **File**, **FileFolder**, and **Sound** classes are processed on the application server and the setting of this property is ignored. For shared transient instances, the **usePresentationFileSystem** property defaults to **false**. (A file or a folder opened on one presentation client cannot be accessed by another client.)
- The **Application**::**checkPictureFile** method always refers to a file on the presentation client.
- The **Application**::**loadPicture** method always refers to a file on the presentation client.
- The **Sound**::**isPlayable** method always executes on the presentation client.
- The **Sound**::**play** method always executes on the presentation client.
- The **Application** class **playSound** and **playSoundAsync** methods always execute on the presentation client.
- The **Application**::**alert** method always executes on the presentation client.
- The **Application**::**actualTime** method returns the current date and time relative to the presentation client. Use the **Application**::**actualTimeAppServer** to return the current date and time relative to the application server.
- Local variables of type **Date**, **Time**, and **TimeStamp** in JADE logic are initialized with the date and time relative to the presentation client.
- JADE exception dialogs are always displayed on the presentation client. The displayed information includes the application server and application that resulted in the error.

  The default exception dialog sets the **Abort** button **cancel** property to **true** when running as a presentation client using **jade.exe** so that pressing the Esc key generates an abort action.
- The start-up form always looks for the .**bmp** bitmap, .**avi** video, or .**gif** (Graphics Interchange Format) files on the presentation client.
- The **Window**::**getSystemColor** method always returns the colors defined for the presentation client.
- The **Window**::**getSystemMetrics** method always returns information as defined for the presentation client.
- Help file processing is performed on the presentation client.

- External function definitions have **applicationServerExecution** and **presentationClientExecution** options, with **presentationClientExecution** being the default option. If JADE is not running in thin client mode, the qualifier has no effect and the external call is made on the workstation executing the JADE logic.

  If running in thin client mode, the **presentationClientExecution** option causes the function to be executed on the presentation client. For details, see "Calling External Functions from JADE Thin Clients", in Chapter 1 of the *JADE External Interface Developer's Reference*.

- The **Application**::**getIniFileName** method retrieves the initialization file from the presentation client. Use the **Application**::**getIniFileNameAppServer** method to obtain the initialization file from the application server.

- The **Application**::**getProfileString** method retrieves the specified initialization file option from the specified file on the presentation client.

  Use the **Application**::**getProfileStringAppServer** method to obtain the information from the application server.

- The **Application**::**setProfileString** method sets the specified initialization file option in the specified file on the presentation client. Use the **Application**::**setProfileStringAppServer** method to set the option in an initialization file on the application server.

- The **Window** class **drawPictureAt** and **drawPictureIn** methods cache the pictures that are drawn on the presentation client.

- The **Node**::**createExternalProcess** method provides the **thinClient** parameter (which you can set to **true** if you want to initiate the process on the presentation client workstation).

- The **Application**::**getJadeInstallDir** method retrieves the installation directory on the presentation client. Use the **Application**::**getJadeInstallDirAppServer** method to obtain the installation directory on the application server.

- JADE reduces memory overheads by always compressing **OleObject** data when passing it to and from the application server and presentation clients. This is transparent if you use the **OleObject** class **copy**, **getData**, and **setData** methods to manipulate the binary image.

- JADE notifications may have a differing execution order when intermixed with window events in thin client mode. This difference arises because the notifications occur on the application server rather than the presentation client. Notifications are usually interlaced with any window events that may occur.

  In JADE thin client mode, the notification occurs when the application server thread processing the presentation client operations becomes idle. However, the presentation client may also be idle and send event notifications such as form activations, focus changes, and so on, at the same time. This asynchronous operation may result in a slightly different execution order for these events from that experienced in JADE when it is not running in thin client mode.

- As inconsistent results could be returned to the application server when running in JADE thin client mode and there are locale (regional) overrides, all overrides on the application server are suppressed. Formatting of locale data is done on the application server, based on the locale of the corresponding presentation client.

  For example, if the locale of your application server is set to English (United Kingdom), which has a default short date format of **dd/mm/yyyy**, and it has been overridden with a short date format of ***yyyy-mm-dd***, this is returned in the default **dd/mm/yyyy** format.

- The value of the **Process** class **signOnTime** property is relative to the server executing the application. In thin client mode, the time is relative to the workstation executing the presentation client.

JADE Thin Client Performance Considerations

When running in thin client mode, you should be aware of the following performance considerations.

- The information required to display a form (determined at JADE paint time) is collected and then stored in the database when possible the first time a form is created. (This is not done for read-only schemas.)

  Subsequent creates of the form then send a request to build that form using the cached build data on the presentation client, unless that form or one of its subclasses has changed.

  You should therefore ensure that all of the forms in deployed systems (including JADE) have had their forms built or generated by accessing all forms for the application. Failure to do so can cause a rebuild and a retransmission of that data to the presentation client on each form **create** method if the created data cannot be stored (that is, in read-only schemas or for JADE-supplied forms).

- Presentation clients should always be run in the font mode of the application server, which is usually a small font. If the font mode of the presentation client differs from that of the application server, font sizing operations executed from logic must be passed to the presentation client for correct execution (that is, the **Window** class **drawTextWidth**, **drawTextHeight**, **getTextExtent**, and the control **create** method where the application default control height must be determined).

- To achieve maximum performance in the JADE thin client mode of operation, the number and size of transmissions to and from the presentation client need to be reduced.

  All events that have defined JADE methods are transmitted to the application server for processing, with the exception of **mouseMove** and **dragOver** events, whose presentation client handling is described in the following item. On a slow communications link, this can result in a noticeable turnaround time.

  The **thinClientUpdateInterval** property of the **ProgressBar** control enables you to specify in milliseconds how often the progress bar is updated as the percentage changes, preventing an unnecessary number of messages being sent over the TCP connection when the progress updates at a fast rate. (For details, see "Updating the Progress Bar", earlier in this appendix.)

  Key events can also be significant. For example, if a form has **keyDown**, **keyPress**, and **keyUp** event methods and the control with focus also has **keyDown**, **keyPress**, and **keyUp** event methods, six transmissions are required to process the action (or seven transmissions for a text box where the **change** event also occurs). The use of form key events in JADE thin client mode can have a great impact on performance because of the number of events sent to the application server for every keystroke.

  If performance is an issue, perform one of the following actions.

  - Remove or disable the form key events.
  - Use the **Form**::**registerFormKeys** method to declare the set of key strokes that invoke the key events.
  - Use the **JadeEditMask** control functionality to improve thin client performance by removing validation requirements.

- The **MouseMoveTime** parameter in the [JadeThinClient] section of the JADE initialization file enables users to specify the time at which **mouseMove** and **dragOver** events are discarded when moving the mouse within the same window if the time since the execution of the last move event is less than the specified time defined for the current application, unless the mouse comes to rest. (The mouse comes to rest if no **mouseMove** events are received for the minimum of the specified time or the default value of 200 milliseconds.)

  If the user moves the mouse slowly enough, the same results are achieved as those when running your application in standard (fat) client mode.

The first **mouseMove** event received after left-clicking a control in thin client mode immediately generates a **mouseMove** event call to the application server (when that control has logic defined for that event). The **mouseMove** time processing then starts with the next **mouseMove** event that is received.

In JADE thin client mode, no **mouseMove** events are sent to the application server if there is no **mouseMove** event defined for that window. This mouse move style is transparent to most application operations and achieves a significant reduction of events that are sent. (The extent of the reduction varies, according to the type of move performed.) For more details, see Chapter 1 of the *JADE Initialization File Reference*, "JADE Initialization File".

Note The Painter running in JADE thin client mode uses a fixed **MouseMoveTime** of 200 milliseconds, regardless of the setting of the **MouseMoveTime** initialization file parameter.

In addition, use the **Application**::**getMouseMoveTime** method to get the current mouse move time that is in use on presentation clients or the **setMouseMoveTime** method to dynamically set the current mouse move time for the current application running on presentation clients. (For details, see Chapter 1 of the *JADE Encyclopaedia of Classes*.)

- All form definitions (presentation cache files) used in the thin client mode are saved in the ***<jade-temporary-directory>*\Jade*<modified-application-server-name>*<*application-server-port*>.jfm** file in the directory specified by the value of the **JadeWorkDirectory** parameter in the [JadeEnvironment] section of the JADE initialization file of the presentation client that is running the process (unless overridden by the **FormCacheFile** parameter in the [JadeThinClient] section of the JADE initialization file on that workstation).

  The *modified-application-server* value consists of any characters that are alpha, numeric, space, underscore, dash, or dot (which are converted to underscores).

  By default, the work file directory is created at the same level as the JADE binary directory (**bin**) 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 cache file format enables the same JADE binaries to connect to different application servers and retain use of a presentation client cache file that is specific to the application server and port. Because it is not known whether the previously used cache file is used or required by the initiation of other applications, it is not deleted.

  When a form is displayed in thin client mode, that form does not need to be transmitted to the presentation client again unless it or one of its superclasses has been changed or the form build data cannot be stored.

  When an application is initiated, it transmits the forms it already has cached (and their build versions) to the application server. When form **create** method is called, the application server needs only transmit a request for that form to be built without having to transmit the form data. This can be significant for a large form or where the form has large picture definitions.

  This stored data is used across sessions. The first access of a form can take a significant time over a slow link when the form definition is transmitted initially.

Tips To overcome this initial delay, you can generate the form in a local thin client environment and then include the generated file in the distributed client installation files.

Set the **UseCacheFile** parameter in the [JadeThinClient] section of the JADE initialization file to **false**, to specify that form and picture caching is not used on a presentation client. By default, forms and pictures are cached, as performance is significantly improved when form and picture information is cached on most connections. However, on a high-speed LAN where caching may not be necessary, setting this parameter to **false** transmits form and picture data in full to the presentation client, instead of creating a cache file.

For details about discarding picture and form entries in the presentation client cache file, see "CacheEntryTimeout" under "JADE Presentation Client Section [JadeThinClient]", in the *JADE Initialization File Reference*.

- JADE skins use a facility similar to form build data. When the **app.setApplicationSkin** or **Form::setApplicationSkin** method is first called, the collected skin data is stored as a blob on the **JadeSkinApplication** instance. Subsequent calls then do not need to retrieve the skin information, and use the stored information.

  In addition, a presentation client caches the skin information. As a result, subsequent calls of the **app.setApplicationSkin** or **Form::setApplicationSkin** method only need to request the creation of the skin from the presentation client cache file without having to transmit the skin data.

  Changing a skin definition using the **JadeSkinMaintenance** form or by loading a form and data definition (.**ddb** or .**ddx**) file will update the timestamp of all **JadeSkinApplication** instances so that a rebuild of the skin information will be required the first time each skin is set for an application or form. If you change JADE skin information by any other means, you must call the **updateSkinTimeStamp** method on the **JadeSkinApplication** instance, to reset the instance timestamp and cause the skin build data to be rebuilt.

- All picture properties set by logic are cached on both the application server and in the form file on the presentation client. When a picture has been transmitted to the client, subsequent use of that picture can be achieved by use of an identifier to that picture, eliminating the need to transmit the picture data.

- When the contents of a text box change, only changed text is transmitted to the application server.

- If the connection speed is low, disable the Debugger bubble help when stepping through code in thin client mode, by clicking the **Bubble Inspector** check box in the Debugger Options dialog.

- All JADE logic GUI set property calls are buffered. The application server validates the property change and queues the change request for future processing. The changes are processed when a subsequent request requires interaction with the presentation client or the current processing is complete.

  This buffering process should be transparent to the user and any logic, as the property values always return the values as if they had been applied immediately. (For details about forcing visual changes to be applied at some logical point before the current processing is complete, see "Flushing Commands Queued in the Application Server", earlier in this appendix.)

- Most GUI property value read requests have the required data available within the application server environment, eliminating the need to access the presentation client for that information.

Tip Even if you are not operating in thin client mode, it is much more efficient to copy a GUI value into a local variable for reuse rather than request the value again. For example, **listBox.listCount** requires the calling of a list box method to retrieve the value.

Storing the value in a local property for reuse avoids a significant overhead for the second and subsequent requests for that value when it will not change. The first of the following examples is much more expensive than the second of these examples.

```
while count <= listBox.listCount do // inefficient use of the variable


vars                                // recommended use of the variable
   drag : Integer;
begin
    drag := keysList.dragListIndex;
    if drag <> -1 then
        keysList.addItemAt(keysList.text, drag);
    endif;
    keysList.listIndex := drag;
end;
```

- The following actions, get property requests, and method calls require communication with the presentation client and must wait for a response from the client.
  - Any form creation
  - Any external function call executed on the client that returns a value or has usage **io** or **output** parameters
  - **node.createExternalProcess**
  - **app.doWindowEvents** or **Window**::**doWindowEvents**
  - **app.getIniFileName**
  - **app.getProfileString**
  - **app.getJadeInstallDir**
  - **app.setProfileString**
  - Any application start (that is, the **startApplication**, **startAppMethodWithString**, **startApplicationWithParameter**, or **startAppMethod**) method
  - **Window**::**drawSize**
  - **Window** class **drawTextHeight**, **drawTextWidth**, and **getTextExtent** methods when the application server and client default font settings differ (that is, small fonts versus large fonts)
  - **Window**::**getPoint**
  - **Window** class **left**, **top**, **width**, and **height** property values must be refreshed after changes where these values may be affected by relative positioning or parent modification
  - **Window**::**refreshNow**
  - **Window**::**setFocus**
  - **Window**::**showHelp**
  - **Form**::**popupMenu**

- **Form**::**show** or **Form**::**showModal**
- **Form**::**unloadForm**
- **Control**::**createPicture**
- **ComboBox**::**isDroppedDown**
- **ComboBox**::**selLength**
- **ComboBox**::**selStart**
- **ComboBox**::**selText**
- **ListBox**::**getScrollRange**
- Any **MultiMedia** class get property request or method call; for example, **position**
- Any **Ocx** or **ActiveXControl** class get property request or method call that returns a value or has a usage **io** or **output** parameter
- Any **OleControl** class get property request or method call
- **Picture**::**pictureHeight**
- **Picture**::**pictureType**
- **Picture**::**pictureWidth**
- **Table**::**expandedHeight**
- **TextBox**::**lineCount**
- **TextBox**::**getScrollRange**
- **TextBox**::**selLength**
- **TextBox**::**selStart**
- **TextBox**::**selText**
- **Binary**::**convertPicture** or **Binary**::**convertToFile**
- **OleObject**::**isServerRegistered**
- **Application**::**isActiveXClassIdRegistered**
- **FileNode** class operations executed on the presentation client
- **Printer** class property or method request that must be obtained from the presentation client

Note When the presentation client requests a print preview, the pages of the printed report do not have to be transferred to and from the application server. (This optimizes the performance of the print preview process when running JADE thin client mode over a slow network.)

However, if your application calls **Printer**::**setReport** to indicate that user logic subsequently stores or manipulates the report output, each page of output is transferred to the application server.

See also "Enhancing the Performance of Windows and Forms", in Chapter 3 of this document.

JADE Thin Client Restrictions

When you run JADE as a presentation client:

- Event methods that are added to a JADE application are not invoked for a form that is running, until that form is restarted.
- Control class event methods that are added are not invoked until the application is restarted.
- Do *not* use the **bufferAddress** method of the **String** or **Binary** primitive type when your application is running in JADE thin client mode, as the use of this method may corrupt your machine as you are passing an address relating to one machine to another machine.
- An external method call made from JADE logic executes on the application server workstation and not that of the presentation client.
- Shared transient instances are shared among all presentation clients connected to the same application server. Logic obtaining the first shared transient instance of a class may therefore no longer work as expected.
- Use of timers whose logic interacts with the presentation client side of the thin client processing may cause a processing loop if the interval between timer calls is less than the time taken to process each request. This could arise over a slower-speed line where the transmission time to the presentation client becomes significant.

  When you develop an application that could run in JADE thin client mode, use timers with care. When a timer event occurs, it notifies the application server, which then echoes the event to all attached presentation clients (that is, the application server sends the notification to each presentation client, which then send a response to the application server). This can have a considerable impact on network traffic.
- Locale handling uses the locale id that is in use on the presentation client workstation, by default. This locale must be installed on the application server workstation. Any local changes on the presentation client to the locale options are ignored (for example, the date format). As inconsistent results could be returned to the application server when there are regional overrides on presentation clients, all overrides on the application server are suppressed by default. For more details, see "JADE Functionality Affected by Thin Client Operation", earlier in this appendix.

  The locale of the presentation client is used to determine the locale to run the thread for that client on the application server machine. It then uses the regional settings of the application server for that locale to determine the formatting of any locale-specific dates, times, and so on. You therefore cannot customize the results returned from a **shortFormat** method or any locale-dependent method on a presentation client. Set the **EnhancedLocaleSupport** parameter in the [JadeEnvironment] section of the JADE initialization file to **true** if you want regional overrides set on the presentation client to be forwarded to the application server so that both use consistent locale settings for the application.
- It is assumed that any fonts set by logic are available on both the application server and the presentation client workstation. When the font setting of the application server matches that of the presentation client workstation (for example, both use small fonts), some font-sizing requests are performed on the application server
- To minimize the footprint of the Presentation Client installation type, the online help files and their required support binaries are neither installed nor shipped for that type of installation.

  If users in JADE thin client mode require JADE online help, install the presentation client software, and then separately use the **Custom** installation option of the full installation CD-ROM and check the **On-line Help Files** component in the Select Components dialog.

Appendix B Upgrading Software on Presentation Clients

This appendix covers the following topics.

Overview

Any JADE version from 7.1.0 and higher will upgrade to a JADE 2016, 2018, or 2020 version. Downgrading from JADE 2020 or 2018 to a version less than JADE 2018.0.01 Hotfix 91 will fail.

Note The rules for the versions that will be handled when upgrading to a later release or downgrading to an earlier release may change in future releases.

You can automatically upgrade JADE and user software on presentation clients. This feature reduces JADE thin client deployment issues when presentation client software changes (installing a new JADE release or upgrading a patched library) and ensures that you cannot run different versions of software on the application server and presentation clients.

Caution Because of the changes to the security model in Windows 10, Windows 8, and Windows 7, if you are upgrading presentation clients under Windows 10, Windows 8, or Windows 7, ensure that you have the appropriate privileges or capabilities to install applications. The configuration of Windows' User Account Control (UAC) and your current user account privileges may affect the behavior of the installation.

The Microsoft Windows C++ 2017 Redistributable Package (x64) is required to be installed on all 64-bit Windows systems that run JADE 2018. If 32-bit thin clients are going to be installed, the equivalent C++ 2017 Redistributable Package (x86) will need to be installed. This will be done as part of the normal JADE installation or upgrade process. (This executable is supplied on the JADE distribution media.) Installing this Microsoft redistributable package requires administration privileges. If possible, deploy this package to all workstations *before* upgrading to JADE 2018, using the appropriate techniques that allow for privileged installations.

See also "Effects of Security on Presentation Client Upgrades", in Chapter 1.

The automatic upgrade of JADE software from the application server keeps the binary files on the presentation client synchronized without any user action being required.

If JADE is installed in the **\Program Files** directory (or the **\Program Files (x86)** directory on a 64-bit machine with 32-bit JADE binaries):

- If the machine has had UAC disabled, the thin client upgrade will fail because of lack of permissions for standard users. For administration users, the necessary privileges are automatically granted so the upgrade will succeed.

- If UAC is not disabled, administrative users are prompted with an **Allow** or a **Cancel** choice but standard users must know and supply the logon and password of a user with administrative privileges to enable the upgrade to succeed.

Set the appropriate parameters in the JADE initialization file only if you are utilizing additional features such as downloading non-JADE files.

The [JadeAppServer] section of the JADE initialization file on each application server provides the **UpgradeRuntimeTo64bit** parameter, which enables you to upgrade 32-bit presentation (thin) client binaries to 64-bit binaries when the client is using a 64-bit operating system. For details, see "Automatic Download Issues and Considerations", later in this appendix.

When a JADE 2016 Windows 32-bit ANSI or Unicode presentation client connects to a JADE 2018 application server, the presentation client is upgraded to version 2016 Microsoft Visual Studio 2017 binaries, by default. (If the client is not running a JADE version that is using Visual Studio 2005 runtime libraries, the JADE thin client download will not revert to a version that uses Visual Studio 2005 runtime libraries.)

Note JADE 2018 does not support 32-bit presentation clients using the Visual Studio 2005 C++ runtime binaries that were required by JADE 6.3 releases.

The presentation client binaries always check against the same version as the binaries themselves; for example, 32-bit or 64-bit. You can change them from 64-bit to 32-bit, or the reverse, only by re-installing the presentation client binaries.

If the presentation client binaries are 64-bit, they are always downloaded from the **bin** directory files on the application server and any additional files are downloaded from the **download** directory under the **..\x64-msoft-win64-ansi** if an ANSI or **..\x64-msoft-win64-unicode** if a Unicode directory, and these are expected to be the correct 64-bit versions. No check of a 32-bit versus a 64-bit binary version is performed.

If the presentation client binaries are 32-bit and built with Microsoft Visual Studio 2017, Microsoft Visual Studio 2013, Microsoft Visual Studio 2010, or Microsoft Visual Studio 2005, the application server looks for all files to be downloaded under **..\i686-msoft-win32-ansi** or **..\i686-msoft-win32-unicode**, if ANSI or Unicode, respectively.

The automatic upgrade process enables you to:

- Disable the facility.
- Automatically update presentation client binaries.
- Download additional files into the binary directory.
- Download other files into a directory that you specify.
- Pre-download files ready for a future application server upgrade.
- Control the number of concurrent downloads from an application server.
- Monitor information about downloads in progress and pre-downloads previously completed.
- Run your own post-installation program on the presentation client.
- Register any **ActiveXControl** or **Ocx** controls that are downloaded.
- Installs any .**msi** installers that are downloaded.
- May attempt to install C++ runtime, if the appropriate redistributable executable is downloaded (that is, **vcredist_x64.exe** or **vcredist_x86.exe**).

The JADE thin client automatic download process:

- Rejects files greater than 1G byte.
- Determines up-front the memory size required to collect all of the files, so that only one allocation is performed rather than resizing the buffer as each file is added.
- Raises an exception if the memory allocation fails.
- Reuses buffer allocation for all of the steps required in the download process.

As a JADE application server can handle presentation clients simultaneously, the application server requires different sets of files to be available for downloading to presentation clients. Those files must each reside in a different directory identified by the hardware type, vendor, operating system version, and whether they are ANSI or Unicode files.

The following rules apply to presentation clients. If the files for:

- The thin client (**jade.exe**) environment is available on the application server, all of the files are upgraded as required.
- Required binaries are not all available to be downloaded, the presentation client cannot continue the session because the existing presentation client binaries may not be valid and consistent with those of the application server, even though the JADE versions match; for example, because of ANSI versus Unicode binaries, presentation client protocol changes, or client operating system changes.
- The non-running environment are not available, they are ignored, no warnings are logged, and the download completes only for the environment that is running.
- The required Microsoft redistributable packages are already installed, the JADE installation program skips the **vcredict** install step. A message is logged accordingly. If the redistributable package is not already installed, the installation fails if you are installing presentation clients and you do not have the appropriate administrator privileges. You must therefore manually install the redistributable package under administrator privileges.

  To install Visual C++, execute one of the following from **<jade>\<a_>bin** or **<u_>bin** on the release medium, as applicable.

  - vcredist_x64.exe
  - vcredist_x86.exe

For example:

```
jade7103.002\Files\Ansi64\a_bin\vcredist_x64.exe
```

The automatic JADE thin client download process determines if the existing files on the presentation client and new files on the application server are identical. The automatic download rules are as follows. If the file:

- Is not present, the file is downloaded.
- Length is different, the file is downloaded.
- Is an executable (that is, a file of type .**dll** or .**exe**), the timestamps of when the two files were linked are compared, rather than the timestamps of when the files were modified. (This timestamp is held in the file contents.)
- Is not an executable and the modified timestamps of the two files do not match, an MD5 comparison of the file contents is used to determine if the two files are different.

A file download therefore occurs only when the two files on the presentation client and application server are actually different; not when the timestamps do not match.

The following are examples of directory file names.

- i686-msoft-win64-ansi
- armv4i-msoft-wce42-unicode
- armv4i-msoft-wce50-unicode

The JADE initialization file on the application server must contain an [*environment-type*] section that controls the download process for each environment type; for example:

```
[i686-msoft-win64-ansi]
DownLoadDirectory        = <default>
FullJadeInstallDirectory =
PostInstallExe           =
```

The directory specified in the **DownLoadDirectory** parameter must have the following two subdirectories.

- **download**, which contains the directories and their read-only files to be downloaded to the JADE HOME directory on the presentation client (that is, the parent directory of the installation directory)
- **downloadProgramData**, which contains program data and read-write files to be downloaded to the directory specified by the value of the **ProgramDataDirectory** parameter in the [JadeEnvironment] section of the JADE initialization file and the location of the installation directory on the presentation client
- **predownload**, which contains the directories and their read-only files to be pre-downloaded to the JADE HOME directory on the presentation client (that is, the parent directory of the installation directory)
- **predownloadProgramData**, which contains the program data and read-write files to be pre-downloaded to the directory specified by the value of the **ProgramDataDirectory** parameter in the [JadeEnvironment] section of the JADE initialization file and the location of the installation directory on the presentation client

If the JADE HOME directory and JADE program data directory of the presentation client are:

- Identical, the result of the download installation is the same as if all the files and directories had been defined under the application server download and predownload directories.
- Different, the read-only files installed under the JADE HOME directories are separated from the read-write program data files installed under the JADE program data directories.

Note To ensure total read-only and read-write separation, the JADE initialization file cannot be located in the JADE HOME directory.

The **DownLoadProgramDataDirectory** and **PreDownLoadProgramDataDirectory** parameters in the [JadeThinClient] section of the JADE initialization file on the presentation client enable you to specify the directory and subdirectories on the presentation client into which all read-write program data files destined for the system directory are copied temporarily prior to them being installed during the automatic software upgrade process, and the directory and subdirectories into which all read-write program data files are copied ready for installation in the future, respectively.

By default, these directories and that of the **DownLoadDirectory** and **PreDownLoadDirectory** parameters are subdirectories of the directory specified by the value of the **ProgramDataDirectory** parameter in the [JadeEnvironment] section of the JADE initialization file and the location of the installation directory on the presentation client.

The following is an example of the directory structure on an application server.

```
jade (or the directory in which the JADE application server is installed)
  bin (or the name of the application server binary directory)
..\Win32\i686-msoft-win32-ansi
      \download\...              All files and directories to be installed in
                                 the JADE HOME directory
          \bin
          \moredata
      \predownload\....          All files and directories to be pre-downloaded
                                 for the JADE HOME directory
      \downloadProgramData\...   All files and directories to be installed in
                                 the JADE program data directory
          \misc
      \predownloadProgramData\.  All files and directories to be pre-downloaded
                                 for the JADE program data directory
```

The following is an example of the directory structure on a presentation client.

```
..\JADE-home-directory
      \bin\..
      \moredata\...              Example of installed directory under the JADE HOME
                                 directory
..\JADE-program-data-directory
      \download                  Temporary files and directories prior to
                                 installation in JADE HOME directory
      \predownload               Current pre-downloaded files for the JADE HOME
                                 directory
      \downloadProgramData       Temporary files and directories prior to
                                 installation in program data directory
      \predownloadProgramData\   Latest pre-downloaded files for the program data
                                 directory
      \misc                      Example of installed directory under the JADE
                                 program data directory
```

To support the separation of read-only and read-write files at run time, the **jaddinst.exe** program does not write any files into the JADE installation directory. The **JadeWorkDirectory** parameter in the [JadeEnvironment] section of the JADE initialization file defines the location of a directory that JADE uses for work files. This directory, which is required for internal use such as interlock files used during downloading, defaults to ***<jade-program-data-directory>*\temp**.

If the JADE thin client binaries to be downloaded from the application server are placed in the base download directory (for example, **...download/jade.exe**) instead of a subdirectory (for example, **.../download/bin/jade.exe**), the thin client installation or upgrade issues the following warning in the **jommsg.log** file on the application server and the thin client **download.log** file if **jade.exe** is found in the base download directory.

```
Warning: Jade.exe is in the download base directory on the app server instead of a
sub-directory

e.g. ...\download\Jade.exe instead of ...download\bin\Jade.exe

This discrepancy will be handled, BUT no sub-directory downloading will be
performed.
```

The thin client cache file is also written into this directory if the location has not been specified by the **FormCacheFile** parameter in the [JadeThinClient] section of the JADE initialization file. The **download.log** file created by the download install process is output to the log directory specified in the **LogDirectory** parameter in the [JadeLog] section of the JADE initialization file.

Notes All directories and subdirectories under the **download**, **downloadProgramData**, **predownload**, and **predownloadProgramData** directories are downloaded to the presentation client. As a result, these directories should contain only those files relevant to a thin client environment. Only the **download**, **downloadProgramData**, **predownload**, and **predownloadProgramData** subdirectory names in each environment structure are fixed and should therefore not be changed. However, the presentation client directory names must be the same as the directories under the application server **download** and **downloadProgramData** directory for each presentation client environment, with the exception of the directory containing the **jade.exe** executable, which is installed into the equivalent directory on the presentation client. Any directories that are not present are created.

If the presentation client environment matches that of the application server (that is, the hardware type, vendor, operating system version, and the ANSI or Unicode file type), the JADE binary files are taken from the application server installation environment and any such files under the directory specified in the **FullJadeInstallDirectory** parameter are ignored.

When the presentation client environment differs from that of the application server and a value is specified in the **FullJadeInstallDirectory** parameter in the environment-specific section of the JADE initialization file, the specified directory in that parameter is used as the source for downloading the JADE thin client binary files to the presentation client. Use this parameter to ensure that the thin client files are downloaded from a standard JADE installation file set without having to maintain a separate copy of the JADE thin client files for the download process.

All files and directories in the application server **download** and **downloadProgramData** directories are downloaded to the presentation client using the same directory structure. The first of the following examples is installed into the presentation client JADE install directory **C:/41260/**. The second of these examples is installed into the **C:/41260/a_thin** subdirectory of the JADE presentation client install directory **C:/41260/**.

```
../i686-msoft-win32-ansi/download/File1.txt
../i686-msoft-win32-ansi/download/a_thin/File2.txt
```

When files and directories are downloaded to the presentation client from the application server, the directory structure under the application server download directory (for example, **...download/jade.exe**) for the architecture of the presentation client is duplicated in the JADE installation directory on the presentation client. The files in those directories on the application server are downloaded and copied to the equivalent directory on the presentation client. (The directories are created if they are not present.) The exception to this, however, is the directory that contains the **jade.exe** executable on the presentation client and the application server download directory that contains the **jade.exe** executable, as those directories are assumed to be the equivalent **bin** directories in both environments.

The result is that all of the files in the download **bin** directory of the application server are copied to the presentation client **bin** directory, even if the names are different. In addition all subdirectories of the application server **bin** directory are copied as subdirectories of the presentation client **bin** directory. For example, the:

- Application server has:

  ```
  ......\download\bin\ (which contains jade.exe)

   .....\download\bin\sub-bin\
  ```

- Presentation client has:

  ```
  <JADE-install-directory>\mybin\ (which contains jade.exe)
  ```

The result is that the files in **bin** are copied to **mybin** and a **sub-bin** directory is created as a subdirectory of **mybin**.

Note The presentation client **DownLoadDirectory** and **DownLoadProgramDataDirectory** parameters in the [JadeThinClient] section of the JADE initialization file specify the temporary location of the respective downloaded read-only and read-write program data files before they are installed.

The **JadeWorkDirectory** parameter in the [JadeEnvironment] section of the JADE initialization file specifies the location of the JADE presentation client **jade.exe** or **jaddinst.exe** program.

The **DownLoadVersion**, **DownLoadDescription**, and **PreDownLoadDescription** parameters in the [JadeAppServer] section of the JADE initialization file are global to the application server. Changing the version causes all presentation client users who attach to the application server and have a different local version to compare their file set with that of the environment type on the application server. In all other situations, automatic download version checking and update occurs only when:

- One of the JADE libraries used by the presentation client has a different version to that being run by the application server
- The JADE executable in use by the presentation client has a different version from the file in the directory specified by the **DownLoadDirectory** parameter in the [*environment-type*] section of the JADE initialization file that controls the download process for each environment type section (for example, the [i686-msoft-win32-ansi] section).

When the presentation client connects to the application server, a comparison of the release versions and patch numbers of the JADE thin client libraries and the **jaddinst** programs is performed. If there is a mismatch of any module, a download is initiated by default. If there is a mismatch and the automatic download feature is turned off, a version error is reported.

The current values of the download version and pre-download versions are stored in the **DownLoadVersion** and **PreDownLoadVersion** parameters in the [JadeThinClient] section of the JADE initialization file on the presentation client and are automatically maintained. This enables the use of the same binary (**bin**) directory for different database systems if unique JADE initialization files are used in each shortcut or command line on the presentation client. As this assumes that each database system is at the same JADE version level, use of each shortcut or command line causes a download to occur when the JADE version level of the database differs.

Note To perform the installation of downloaded files in JADE thin client mode, the **jaddinst** file must be available. If this program is not available, a message is logged in the **jommsg.log** file on the presentation client and the automatic download feature is switched off.

The new **jaddinst** installation program is therefore not required. If a new **jaddinst** executable file has been downloaded, a copy of that new file with a name of ***jade-binary-directory*\jaddinstall** is taken after the latest **jade.exe** file has been installed on each presentation client. The installation is performed by executing the new **jaddinst** file so that the old **jaddinst** file can be backed up and replaced. The temporary **jaddinstall** executable file is removed when the next thin client session is initiated.

In addition, the [JadeThinClient] section of the JADE initialization file on the presentation client provides the **AskToDownload** and **AskToPreDownload** parameters, which are set to **true** by default. These parameters control whether the thin client download and pre-download processes are automatically performed.

The download progress form is always displayed as the top window for the first four seconds of its display, after which it reverts to being a normal window. For more details, see "JADE Presentation Client Section [JadeThinClient]", in the *JADE Initialization File Reference*.

Optional Files Downloaded to Presentation Clients

Downloading of the following files to presentation clients is optional.

- **jadedit.dll**, only if the user will never run a JADE development environment or allow use of the **Debug** button on the exception dialog.

- **jadcnet.dll**, if SSL is not required.
- **jomsec.dll**, if SSL is not required.
- **jadewpf.dll**, if the **JadeXamlControl** class is not used.
- **jadedotnetthin.dll**, if no .NET controls are used.
- **jadodbc_c.dll**, if the ODBC interface is not used by the presentation client.
- **odbcinst.exe**, if the ODBC interface is not used by the presentation client.

The following additional files are optional for 32-bit thin client downloads.

- **vcredis1.cab**, if an upgrade of the runtime distributable files is not required.
- **vcredist.msi**, if an upgrade of the runtime distributable files is not required.
- **javagui.dll**, if using **jade.exe**.

The following additional files are optional for 64-bit thin client downloads.

- **vc_x64runtime.cab**, if an upgrade of the runtime distributable files is not required.
- **vc_x64runtime.msi**, if an upgrade of the runtime distributable files is not required.
- **javagui.dll**, if using **jade.exe**.

Download File Types

For a thin client download, if files with a type of .**msi** (Microsoft Installer) are included in the files downloaded from the application server by the thin client automatic download process, those programs are run synchronously by the thin client download install process after all of the download files have been installed.

The execution of those msi programs should then install whatever is part of each package. If the msi program installations succeed, a copy of the msi files are left in the JADE presentation client installed directories so that subsequent thin client downloads ignore those files. This process is necessary, because what is installed by an msi program cannot be determined from the file itself (for example, using this procedure to install the Microsoft Visual Studio 2017 libraries required by JADE 2020 on a presentation client).

JADE Initialization File Parameters

The [JadeAppServer] and [JadeThinClient] sections of the JADE initialization file provide the parameters listed in the following table, to enable you to control the automatic downloading of JADE software on presentation clients.

Parameter	Description
AskToDownload	Specifies whether users are presented with a message box prompting them to confirm that they want to download the files or that the files are automatically downloaded without requesting user confirmation.
AskToPreDownload	Specifies whether users are presented with a message box prompting them to confirm that they want to download the files or that the files are automatically pre-downloaded without requesting user confirmation.

Parameter	Description
AutomaticDownLoad	Specifies whether the automatic downloading of JADE software is enabled or disabled.
DownLoadDescription	Optionally specifies an additional description of the download displayed to presentation client users.
DownLoadDirectory	In the [JadeThinClient] section only, and specifies the directory and subdirectories into which all read-only files destined for the binary directory are copied temporarily prior to them being installed.
DownLoadMaximum	Controls the number of concurrent downloads of files to presentation clients that can be in effect for an application server.
DownLoadProgramDataDirectory	In the [JadeThinClient] section only, and specifies the directory and subdirectories into which all read-write program data files destined for the system directory are copied temporarily prior to them being installed.
DownLoadVersion	Specifies the current version of additional files located in the directories specified by the **DownLoadDirectory** parameter in the [*environment-type*] section for each environment type.
PreDownLoadCount	Contains the number of users who have pre-downloaded the available JADE upgrade software.
PreDownLoadDescription	Optionally specifies an additional description of the pre-download displayed to presentation client users.
PreDownLoadDirectory	In the [JadeThinClient] section only, and specifies the directory and subdirectories in which files for a future software upgrade are held or into which files are copied ready for future installation.
PreDownLoadVersion	Specifies the current version of pre-download files located in the directories specified by the **DownLoadDirectory** parameter in the [*environment-type*] section for each environment type.
PreDownLoadProgramDataDirectory	In the [JadeThinClient] section only, and specifies the directory and subdirectories in which read-only program data files for a future software upgrade are held or into which files are copied ready for future installation.
PreventFileDownload	In the [JadeAppServer] section only, and specifies whether you want to prevent the automatic thin client download from the application server.
PreventFileDownloadDescription	In the [JadeAppServer] section only, and optionally specifies the message displayed on presentation clients instead of the JADE message when the file download is prevented and the files are not available locally.
RemovePreDownLoadFiles	Specifies whether files installed from the pre-downloaded directories are deleted during the installation process.
UpgradeRuntimeTo64bit	In the [JadeAppServer] section only on each application server indicates whether to upgrade 32-bit binaries running on a 64-bit client operating system to 64-bit binaries.

Each [*environment-type*] section in the JADE initialization file on the application server provides the parameters listed in the following table, to enable you to control the download process for each environment type (that is, hardware type, vendor, operating system version, and ANSI or Unicode files). For details, see "Instructions for Downloading JADE Thin Client Software", in the previous section, or "JADE Thin Client Sections", in the *JADE Initialization File Reference*.

Parameter	Specifies …
DownLoadDirectory	The directory and subdirectories that contain any additional files for each environment type that are to be downloaded to the **jade.exe** binary directory on the presentation client.
FullJadeInstallDirectory	The directory that is the source for downloading the JADE thin client binary files to the presentation client when the presentation client environment differs from that of the application server. This parameter ensures that the thin client files are downloaded from a standard JADE installation file set without having to maintain a separate copy of the JADE thin client files for the download process.
PostInstallExe	A user-written program that is run after the installation of JADE software.

JADE applications rely on the Windows environment to provide specific common fonts.

For details about substituting preferred fonts on presentation clients for those defined in an application, see "Font Substitutions Section [JadeFontSubstitutions]", in the *JADE Initialization File Reference*.

Preventing the Automatic Downloading of Files to Presentation Clients

The [JadeAppServer] section of the JADE initialization file on the application server node provides parameters that enable you to control what happens when any automatic thin client download is required by a presentation client attaching to that application server. The **PreventFileDownload** parameter, which defaults to **false**, enables you to specify that you want to prevent the automatic thin client download from the application server by setting the value to **true**.

The **PreventFileDownloadDescription** parameter does not have a default; that is, it is empty. If you set the **PreventFileDownload** parameter to **true**, you can use the **PreventFileDownloadDescription** parameter to define your own message that is displayed in a message box on the presentation client when pre-download files are not available locally.

If the **PreventFileDownload** parameter has a value of **false** (the default), the standard automatic download and pre-download process occurs.

When you set the **PreventFileDownload** parameter to **true**, the presentation client expects to find all of the files required to be installed on the presentation client, either in the **predownload** directory, the **download** directory, or already installed. If any files are not locally available, a message box is displayed, indicating the install process cannot occur and displaying the description that is specified in the **PreventFileDownloadDescription** parameter.

If the file download is prevented, the files are not available locally, and you do not specify a description, the following is displayed in a message box.

```
The JADE Thin Client needs to update one or more files, but the latest version of
these cannot be found and updates from the application server are currently
disabled. Please contact your System Administrator.
```

This functionality enables you to handle the situation in which presentation clients have their **predownload** directory defined on a local server and the files to be installed are copied to that local server manually or by a non–JADE process. In this situation, no download occurs over the thin client connection and all presentation client upgrades occur locally. The prevention of file downloads therefore enables you to ensure that the operational processes associated with a presentation client upgrade occurs locally and does not involve the standard thin client TCP/IP connection.

Setting the **PreventFileDownload** parameter to **true** also means that a pre-download upgrade over the thin client TCP/IP connection is also ignored.

Notes This functionality applies to all presentation clients attached to that application server, without exception.

The values of these parameters are ignored if the value of the **AutomaticDownLoad** parameter is set to **false** in the [JadeAppServer] or [JadeThinClient] section of the JADE initialization file, as no download is attempted.

Checking for an Automatic Download

When JADE thin client mode checks whether an automatic download is required, the following actions occur.

1. The patch numbers of the required libraries and the release version of the **jade.exe** and **jaddinst** programs are passed to the application server during the initial connection. If there is a mismatch:

   - If the **AutomaticDownLoad** parameter (on both the application server and the presentation client) is set to **true**, a download is invoked to copy the files from the binary directory on the application server.
   - If the automatic download is not in effect, a version error is returned to the **jade.exe** program and the download process terminates.

   If the application server and presentation client are running different environments, the release version and patch numbers of common libraries are compared. If they are different or the module is a presentation client module only, the files in the environment directory relevant to the presentation client are obtained (that is, loaded and retained).

   If the patch number of any common module differs from that of the application server, a message is logged (in the **jommsg.log** file on the application server) and the automatic download continues.

   If the required executable (that is, **jade.exe**) is not available in the JADE work directory and library modules are not available in the binary directory on the application server, an automatic download does not apply and the appropriate entry is logged (in the **jommsg.log** file on the application server).

2. If the **DownLoadVersion** parameter value in the JADE initialization file on the application server differs from the value of the **DownLoadVersion** parameter passed from the presentation client during the initial connection of the presentation client to the application server or a version mismatch occurs, a download is initiated. (This download version is not case-sensitive.)

   The **DownLoadVersion**, **DownLoadDescription**, and **PreDownLoadDescription** parameters in the [JadeAppServer] section of the JADE initialization file are global to the application server. Changing the version causes all thin clients that attach and have a different local version to compare their file set with that of the environment type on the application server.

   In all other situations, automatic download version checking and update occurs only if one of the JADE libraries used by the presentation client has a different version to that being run by the application server or the JADE executable in use by the presentation client has a different version from the file in the download directory.

3. If a version mismatch download was not required during the initial connection, the value of the **PreDownLoadVersion** parameter in the JADE initialization file on the application server is compared with that of the value passed from the presentation client. (This download version is not case-sensitive.) If these versions match, if the **PreDownLoadDirectory** or **PreDownLoadProgramDataDirectory** parameter is empty, or if the maximum number of downloads that is allowed is already in progress, the pre-download does not occur and the application initiation continues as normal. If a pre-download situation is found, a download is initiated.

Note To enable the pre-download process to work, the pre-download directories must have the same subdirectory structure as the corresponding download directories. For details about the directories that are required for each environment type, see "Overview", earlier in this appendix.

Downloading a New Version of Thin Client Software

During the downloading of a new version of software from the application server to presentation clients, the following actions occur.

1. As it is necessary to cater for an environment where several users share the same JADE thin client binary files, only one user can initiate a download and install process at a time into the same binary directory. Therefore, whenever the **jade.exe** program is initiated in JADE thin client mode, it attempts to create an interlock file (**thinlock.fil**) in the work directory of the **jade.exe** program before performing the initial version check with the application server.

   If the creation of the lock file succeeds, the version check is performed. If a download is not required, the interlock file is removed and the execution continues as normal.

   If a download or an install of preloaded files is required, this interlock file has the user name and computer name of the user and the workstation that created the lock written into the file. This file is removed only when the installation process terminates or the pre-download is complete.

   If the creation of the interlock file fails, another user is in the download process for that binary directory. The user is informed of the name of the presentation client workstation and user who is performing the download process, and asked to try again shortly.

   For details about controlling whether the thin client download and pre-download processes are automatically performed, see the **AskToDownload** and **AskToPreDownload** parameters in the [JadeThinClient] section of the JADE initialization file, in the *JADE Initialization File Reference*.

   The file lock is retained by the JADE thin client install process for standard Windows clients. This prevents multiple clients attempting to perform the same download process. If another JADE thin client is initiated while the first client is processing the download requirements and the application server indicates that a download is required, the file lock is will be detected. If the value of the **AskToDownload** parameter is set to **true**, a message box will be displayed (using a temporary copy of **jaddinst.exe** to do the message box display), informing the client that another client is performing the download and asking him or her to wait. If the value of the **AskToDownload** parameter is **false**, the information is output only to the client **jommsg.log** file. In both cases, the JADE executable terminates with exit code 14165 *(The thin client software is currently being downloaded, please wait and try again shortly. The user performing the download is:)*.

   Note You should not use the same JADE binaries to access different application servers at the same time. If the application servers have different download files, a conflict of interest will occur when two JADE thin clients are initiated at the same time and the installation process will most likely fail with files being in use.

2. When the requirement of a download is detected, an analysis is performed of the files that must be downloaded. The application server passes a complete list of all files available for download and their file lengths and timestamps.

The **jade.exe** program determines the files that it already has locally by checking its own work, binary, download, and pre-download directories.

If all files are already local for a download, the download version in the **DownLoadVersion** parameter in the [JadeThinClient] section of the JADE initialization file on the presentation client is updated automatically, the interlock file is removed, and execution of the **jade.exe** program continues. As each database system is assumed to be at the same JADE version level, use of each shortcut or command line causes a download to occur when the JADE version level of the database on the presentation client differs from that on the application server.

If all files are already local for a version mismatch download, a message box prompts the user to confirm that he or she wants to proceed with the installation of the local files. If the user rejects the installation, the interlock file is removed and the **jade.exe** program terminates. If the user accepts the installation, the installation process described in step 5 is performed.

If there are files to download, the pre-download operation in step 3 and the version mismatch download operation in step 4 are performed.

3. In a pre-download situation where there are files still to be downloaded, an attempt is made to create another interlock file (**prelock.fil**). If the creation of the pre-download interlock file fails, another user is already performing the pre-download operation and the original interlock file is removed and the execution of the **jade.exe** program is continued.

   If the creation of the interlock file succeeds, the original interlock file (**thinlock.fil**) is removed, allowing other users to initiate sessions with the application server while the pre-download files are obtained. A message box gives the user the option of performing or ignoring the download process. If the user ignores the download process, the new interlock file (**prelock.fil**) is removed and the application execution continues. This process is repeated every time a JADE thin client connection is initiated to the application server until the pre-download operation is performed.

   If the user accepts the pre-downloading of files, the required files for each environment type are downloaded from the directories and subdirectories specified by the **DownLoadDirectory** and **FullJadeInstallDirectory** parameters in the [*environment-type*] section of the JADE initialization file for the appropriate environment type and the **PreDownLoadDirectory** and **PreDownLoadProgramDataDirectory** parameters in the [JadeThinClient] section of the JADE initialization file. The data is compressed during transmission and a progress bar displays the status of the pre-download operation if the transfer of data takes longer than one second.

   The pre-downloaded files are created with the same timestamp as the modified time stamp of the file recorded on the application server.

   When the pre-download of files is complete, the value of the **PreDownLoadVersion** parameter in the [JadeThinClient] section of the JADE initialization file on the presentation client is updated with the new pre-download version, the interlock file is removed, and the execution of the original JADE application is continued.

   Note If any file fails to copy, a message box advises the user of this and prompts the user to abort or retry the pre-download operation. If the user aborts the operation, any partially written file is deleted, the interlock file is removed, and the execution of the JADE application continues as normal. If the user retries the operation, another attempt is made on the file operation, allowing the user to deal with issues such as lack of disk space.

4. If files must be downloaded for a version mismatch or upgrade, a message box informs the user that a download is required before the application execution can continue. If the user rejects the download, the interlock file is removed and the execution of the **jade.exe** application terminates. When the user accepts the download, the application server downloads the files to the presentation client. The data is compressed during transmission and a progress bar displays the status of the download operation if the transfer of data takes longer than one second.

   The application server downloads files that are not already present on the presentation client from the directories and subdirectories defined by the values of the **DownLoadDirectory** and **FullJadeInstallDirectory** parameters in the [*environment-type*] section. Note, however, that if a file version mismatch is detected, the mismatched files are read from the binary directory on the application server. The currency of any other file required by the JADE thin client (for example, the **jadmsgs.eng** file) is also checked, using the files in the binary directory on the application server.

   The files downloaded from the directory and subdirectories specified in the **DownLoadDirectory** and **FullJadeInstallDirectory** parameters for the specific environment type (for example, in the [i686-msoft-win32-ansi] section) on the application server are copied into the directory and subdirectories specified by the **DownLoadDirectory** and **PreDownLoadProgramDataDirectory** parameters in the [JadeThinClient] section of the JADE initialization file on the presentation client. If these parameters are not specified, directories named **download** and **downloadProgramData** at the same level as the binary directory containing the **jade.exe** program on the presentation client are used.

   When the downloading of files is complete, the installation process described in step 5 occurs.

   Note If any file fails to copy, a message box advises the user of this and prompts him or her to abort or retry the download operation. If the user aborts the operation, any partially written file is deleted, the interlock file is removed, and the execution of the JADE application continues as normal. If the user retries the operation, another attempt is made on the file operation, allowing the user to deal with issues such as lack of disk space.

5. When the files are available for installation, the JADE **jaddinst** download installation program is initiated, passing all of the information required to complete the installation.

   If the **jaddinst** installation program was downloaded, it is initiated from a temporary copy of the file in the **temp** directory so that the file can be copied into the JADE binary directory. If the **jaddinst** download installation program was not downloaded, it is executed from the JADE binary directory. See also the note at the end of the "Overview" section, earlier in this appendix.

   The original **jade.exe** process terminates so that the original binary files can be overwritten.

   The **jaddinst** download installation program performs the following actions.

   a. Takes a backup copy of the files that will be replaced into a directory named **backup**, at the same level as the work directory that contains the **jade.exe** program.

   b. Ensures that the files in the original binary directory are not in use. If they are, the user is requested to shut down the processes that use those files.

   c. Copies the downloaded files into the required directories.

   d. Clears the files from the download directories and subdirectories.

   e. Registers with Windows any downloaded files that have the **.ocx** file type.

   f. Updates the **DownLoadVersion** or **PreDownLoadVersion** parameter in the [JadeThinClient] section of the JADE initialization file on the presentation client with the new download or pre-download version from the **DownLoadVersion** or **PreDownLoadVersion** parameter in the [JadeAppServer] section of the JADE initialization file on the presentation client.

g. Removes the interlock file.

h. If the value of the **PostInstallExe** parameter for the specific environment specified in the [*environment-type*] section (for example, in the [i686-msoft-win32-ansi] section) of the JADE initialization file is valid, this program is executed.

If this parameter does not contain a valid value, a message box advises the user that the installation is complete and that he or she can now initiate the original JADE application request.

i. After the latest **jade.exe** file has been installed on each presentation client, the files installed from the pre-downloaded directories are deleted if the value of the [JadeThinClient] section **RemovePreDownLoadFiles** parameter in the JADE initialization file on presentation clients is set to **true**.

Notes All installation operations are logged into a file named **download.log** in the work directory of the **jade.exe** program on the presentation client.

If any file fails to copy, a message box advises the user of this and prompts the user to abort or retry the installation operation. If the user aborts the operation, the installation restores the binaries using the backup copy that it created previously, deletes the interlock file, and terminates. If the user retries the operation, another attempt is made on the file operation, allowing the user to deal with issues such as lack of disk space.

The installation program allows the renaming of the **jade.exe** program on the presentation client. This is achieved by changing the **jade.exe** program downloaded from the application server to the name that appeared on the command line originally.

Monitoring the Automatic Download Process

The Application Server monitor window draws the background color of any entry in which a download is occurring in yellow. In addition, the schema column displays **Downloading**.

The **Node** class provides the **downloadCount** method, which has the following signature.

```
downloadCount(): Integer;
```

This method returns the number of processes that are currently performing a download.

For details about the **PreDownLoadCount** parameter in the [JadeAppServer] section of the JADE initialization file on the application server, see "PreDownLoadCount" under "JADE Thin Client Sections", in the *JADE Initialization File Reference*.

Instructions for Downloading JADE Thin Client Software

» To pre-download thin client software

1. Ensure that directories have been specified for the **DownLoadDirectory** and **FullJadeInstallDirectory** (if required) parameters in the environment-specific section of the JADE initialization file on the application server.

2. Copy the files that are to be pre-downloaded into these two directories (and any subdirectories), as required.

   For details about files that are optional in specific situations, see "Optional Files Downloaded to Presentation Clients", earlier in this appendix.

3. Clear the value of the **PreDownLoadCount** parameter in the [JadeAppServer] section of the JADE initialization file on the application server so that a correct count of the number of pre-downloads can be determined.

4. Set the value of the **PreDownLoadDescription** in the [JadeAppServer] section of the JADE initialization file on the application server, if required.

5. Change the value of the **PreDownLoadVersion** parameter in the [JadeAppServer] section of the JADE initialization file on the application server so that each presentation client can detect that a pre-download is available.

» To install a new or patch version of thin client software

1. Copy all non–JADE files destined for the binary directory on the presentation client into the directory (and any subdirectories) specified by the value of the **DownLoadDirectory** and **FullJadeInstallDirectory** (if required) parameters in the environment-specific section of the JADE initialization file on the application server.

   Notes Any files that are pre-downloaded must still be copied to the directory (and any subdirectories) specified by the **DownLoadDirectory** parameter.

   For details about files that are optional in specific situations, see "Optional Files Downloaded to Presentation Clients", earlier in this appendix.

2. Install the new JADE files into the binary directory on the application server, including the extra files required by the presentation client (that is, the **jadmsgs.eng** and **FreeImage** files).

3. Update the values of the **DownLoadVersion** parameter and optionally the **DownLoadDescription** parameter in the [JadeAppServer] section of the JADE initialization file on the application server.

Automatic Download Issues and Considerations

When a 32-bit presentation client connects to a 64-bit application server, the application server upgrades the version of the presentation client but it does not change the 32-bit to 64-bit type of the presentation client, because:

- The presentation client currently does not check to see if the operating system on which it is running is 64-bit-capable (and it would have to inform the application server about this).
- Any support libraries needed by the presentation client (for example, ActiveX control and automation libraries) would also have to be downloaded or already installed in the presentation client.

Note Visual C++ runtimes are always upgraded (that is, the **x64** version is installed) as part of the upgrade process.

If a presentation client running with 32-bit binaries requires 64-bit binaries, you can manually install them. Once installed, the presentation client will automatically upgrade, but with 64-bit binaries. Alternatively, 32-bit presentation clients can be upgraded to 64-bit binaries, as follows.

- The **UpgradeRuntimeTo64bit** parameter in the [JadeAppServer] section only on each application server, when set to **true**, causes a 32-bit client running on a 64-bit operating system (WoW64) to be upgraded to 64-bit binaries.

When the **UpgradeRuntimeTo64bit** parameter is set to **true**, presentation clients that are running a 32-bit version of **jade.exe** on a 64-bit operating system cause a download and installation of 64-bit binaries to that client (with the exceptions described later in this section). The default value of **false** indicates that 32-bit version presentation clients continue to download 32-bit binaries when required. Presentation clients that are running a 32-bit version of **jade.exe** on a 32-bit operating system continue to download 32-bit binaries when required. If the client is not running a JADE version that is using Visual Studio 2005 runtime libraries, the JADE thin client download will not revert to a version that uses Visual Studio 2005 runtime libraries.

Note The **UpgradeRuntimeTo64bit** parameter applies only after the *next* thin client download has occurred, because any currently deployed **jade.exe** currently does not communicate the operating system type to the application server. To get 32-bit clients to upgrade to 64-bit binaries therefore requires two download and install processes: one to upgrade the **jade.exe** so that it will identify the operating system type to the application server, and the second to upgrade from 32-bit to 64-bit binaries.

The 32-bit version of presentation client binaries must be installed on the 64-bit application server, in one of the following directory structures.

- *<jade-program-data-directory>*/i686-msoft-win32-ansi/download/
- *<jade-program-data-directory>*/i686-msoft-win32-unicode/download/

The 64-bit version of presentation client binaries must be installed on the 32-bit application server, in one of the following directory structures.

- *<jade-program-data-directory>*/i686-msoft-win64-ansi/download/
- *<jade-program-data-directory>*/i686-msoft-win64-unicode/download/

The automatic download feature requires that the **jade.exe** and **jaddinst** executable program binaries and all libraries required by the **jade.exe** program running in thin client mode must be present in the binary directory on the application server. As the version information for these components is loaded when the application server is initiated, these modules therefore cannot be renamed on the application server.

The binary directory on the application server must have read access. The binary directory on each presentation client must allow read and write access to create the interlock files, create and update the values of the **DownLoadVersion** and **PreDownLoadVersion** parameters in the [JadeThinClient] section of the JADE initialization file on the presentation client, copy the older versions of the files from this directory, and allow the **jaddinst** program to copy files into the directory and overwrite the existing files.

Do not use the same binary directory to communicate with multiple application servers, as each application server may attempt to alternately upgrade the files in turn.

If the JADE initialization file on the presentation client is read-only, the download version cannot be updated. As a result, each time the presentation client initiates the application and the application server and presentation client download versions are different, they go through a file comparison phase that is not necessary, consuming more resources, and causing a longer initiation time.

When the JADE thin client downloads any files with an **.ocx** name type, an attempt is made to register these files to the Windows operating system. This registration process can be performed only under administration mode. The JADE thin client installer detects that an OCX file has to be installed and attempts to re-run itself in administration mode so that the OCX file can be registered with the Windows operating system.

If the installer can be run in administration mode, the files downloaded are installed as normal and any OCX files are registered. If the client environment does not permit the operation of administration mode, the attempted re-initiation fails and the download installation continues in non-administration mode. This results in the OCX registration process failing and being reported to the user. This is a non-fatal error, and use of the OCX will require manual intervention from an administrative user.