Runtime Application Guide

VERSION 2018.0.01

jade

Contents

Contents v

Before You Begin

The *JADE Runtime Application Guide* is intended as a main source of information when you are administering JADE runtime user applications.

Who Should Read this Guide

The main audience for the *JADE Runtime Application Guide* is expected to be system administrators.

What's Included in this Guide

The *JADE Runtime Application Guide* has six chapters and one appendix.

Chapter 1	Covers running a JADE user application
Chapter 2	Using skins in runtime applications
Chapter 3	Using rich text controls on runtime forms
Chapter 4	Converting a user database
Chapter 5	Deploying database files
Chapter 6	Upgrading to a new release of a runtime application
Appendix A	Covers customizing the deployment upgrade process

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 Development Environment Administration Guide	Administering the JADE development environment
JADE Development Environment User's Guide	Using the JADE development environment to development JADE applications
JADE Installation and Configuration Guide	Installing and configuring JADE
JADE Initialization File Reference	Maintaining JADE initialization file parameter values
JADE Report Writer User's Guide	Using the JADE Report Writer to develop and run reports
JADE Thin Client Guide	Administering JADE thin client environments
JADE Web Application Guide	Implementing, monitoring, and configuring Web applications

Conventions

The *JADE Runtime Application 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 "Profiling an Application" 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 CAPITALS	Keyboard 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 Running a JADE User Application

This chapter covers the following topics.

Running a JADE Application

You can run JADE applications in the following ways.

- Single user mode via the JADE executable (**jade.exe**), as shown in the following command line example.

  ```
  c:\jade\bin\jade.exe path=c:\jade\system server=singleUser app=Accounts
  ini=c:\jade\system\jade.ini schema=AccountsSchema
  ```

  For details, see "Single User Configuration" under "JADE Configurations", in Chapter 1 of the *JADE Installation and Configuration Guide*.

- Multiuser mode via the JADE executable (**jade.exe**). The following is an example of the command line required to run an **Accounts** application in the default multiuser mode.

  ```
  c:\jade.exe path=s:\jade\system schema=AccountsApp app=Accounts
  ini=c:\jade\system\jade.ini
  ```

  For details, see "Multiuser Configuration" under "JADE Configurations", in Chapter 1 of the *JADE Installation and Configuration Guide*.

- Thin client mode, via the JADE executable (**jade.exe**), as shown in the following command line example.

```
jade.exe app=Accounts ini=c:\jade\system\jade.ini schema=AccountsSchema
appServer=MyAppServer appServerPort=1234
```

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

- From the JADE development environment. For details, see "Running an Application from the JADE Development Environment".
- A non-GUI client application using the **jadclient** executable. For details, see "Running a Non-GUI Client using jadclient".
- A non-GUI server application using the **ServerApplication** parameter in the [JadeServer], [JadeAppServer], or [NonGuiClient] section of the JADE initialization file. (For details, see the *JADE Initialization File Reference*.)
- From JADE code in an application, by using the **Application** class **startApplication**, **startApplicationWithParameter**, or **startAppMethod** method. (For details, see Chapter 1 of the *JADE Encyclopaedia of Classes*.)

A user application developed using an **Enterprise** or **Free** restricted licence that is invoked from outside of the JADE development environment displays a splash screen stating that it is a test environment. (A test environment is not for production use.) The only action that the user can perform is to click anywhere on the splash screen or to press any key to remove the splash screen.

The type of licence is displayed at the bottom of the JADE development environment sign-on screen (that is, an unrestricted primary (**Production**) licence or an **Enterprise** or **Free** restricted licence).

Operating system command line arguments are delimited by white space, which is either a space or a tab. A command-line argument is the information that follows the program's name on the command line of the operating system. Command-line arguments are used to pass information to the program.

A command is split into an array of strings named arguments. Argument 0 is (normally) the command name, argument 1, the first element following the command, and so on.

Gerard votes that the elements on the command line are always referred to as arguments. An argument that is interpreted as a command can have arguments that are inputs or parameters to that command. But they are all arguments.

Running an Application from the JADE Development Environment

When an application shuts down, JADE unloads any dynamically loaded libraries that have been loaded if the library is not being used by any remaining applications that are running on the node. When all user applications are shut down, any user-loaded libraries are unloaded. In a development environment, this allows the executable file for a library to be replaced without having to restart the JADE node, as long as all user applications are shut down. This applies to standard (fat) clients, presentation clients, and application servers, but not to the database server node.

If libraries are loaded during execution of a server method, they are held open by the server thread that last executed the method. Server threads are not released when an application shuts down.

» To initiate a JADE application from the JADE development environment

- Click the **Run Application** button from the browser toolbar to select an application to run, or right-click on the toolbar button to run the currently set application.

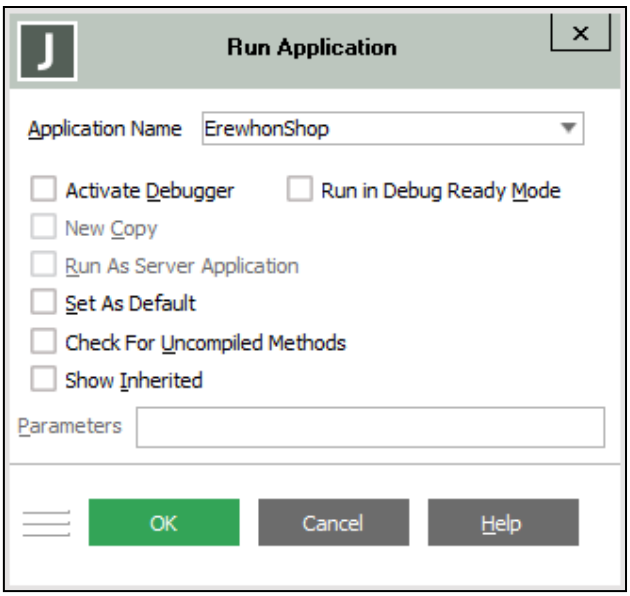
Alternatively, perform one of the following actions.

- Select the Application menu **Run** command from the Application Browser for your selected application or from the File menu
- Press Ctrl+R

Notes As you can run an application only in the current schema context, the **Run Application** toolbar button and **Run** command are disabled in browser windows of the latest schema context.

To run a Graphical User Interface (GUI) application, a start-up form must be specified for that application. (For details, see "Defining Applications", in Chapter 3 of the *JADE Development Environment User's Guide*.)

The Run Application dialog, shown in the following image, is then displayed.



Note To bypass the Run Application dialog, right-click on the **Run Application** toolbar button.

To run your JADE application

1. In the **Application Name** list box, select the application that you want to run. The current application is displayed by default.

   If you want to run another application in the current schema, select the appropriate application in the **Application Name** list box. If no application is set, the first application is displayed.

   For details about running a superschema application from the current schema, see step 7 of this instruction.

2. Check the **Activate Debugger** check box if you want to activate the debugger for the application. By default, the debugger is not activated when an application is run. (For details, see "Starting an Application in Debug Mode", in Chapter 7 of the *JADE Development Environment User's Guide*.)

3. Check the **Run in Debug Ready Mode** check box if you want to run the application in debug mode but without the debugger being initiated when this is checked. Attaching to that application then allows debugging of any method already on the execution stack. (For details, see "Attaching the Debugger to a Running Application".)

Note If the application is idle and no modal dialogs are displayed, attaching the debugger is equivalent to running in debug-ready mode.

4. Check the **New Copy** check box if you want to run a new copy of the selected application when that application is currently running.

   The **New Copy** check box is disabled if the application is not already running.

5. The **Run As Server Application** check box is enabled only when the selected application is of type **Non-GUI** or **Web-Enabled Non-GUI** and you are running JADE in multiuser mode. When you check this check box, the **Activate Debugger and New Copy** check boxes are disabled.

   Check the **Run As Server Application** check box if you are running JADE in multiuser mode and you want to run the application on the server node instead of your client node. For example, you can initiate a non-GUI application to run on the server node so that it continues running after you have shut down your client node.

   For details about specifying the time that JADE waits for an application to initiate on another thread before raising an exception, see the **MaxWaitAppStart** parameter in the [JadeServer] section of the JADE initialization file, in the *JADE Initialization File Reference*.

6. Check the **Set As Default** check box if you want to set an application in your schema as the current default application. (This check box enables you to set the default application without having to access the Application Browser to change the current application.)

   This check box is enabled only when the selected application is not the current application.

7. Check the **Check For Uncompiled Methods** check box if you want to be warned before the application is invoked about any methods in the application that are uncompiled or are being edited.

   This check box is unchecked by default; that is, methods are not checked before the application runs. If an uncompiled method is executed, an exception is raised.

8. Check the **Show Inherited** check box if you want to run a superschema application from a subschema.

   The **Application Name** combo box is then populated with the applications from the current schema *and* its superschemas.

9. If the initialize method for the application has a single parameter of type **HugeStringArray**, the **Parameters** text box is enabled so that you can specify parameters for command line arguments when running an application from within JADE.

   Specify each entry in the array as a space-delimited token in the parameters string; for example:

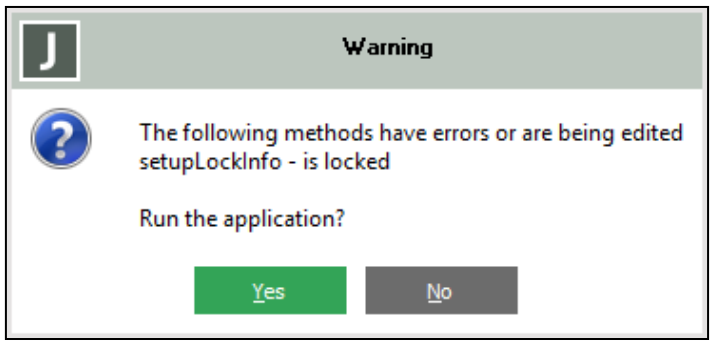
   ```
   "schema=RootSchema app=JadeSchemaLoader ini=MyJade.ini
   schemaFile=d:\jade\scm\test.scm dontSaveSources=true
   loadStyle=onlyStructuralVersioning"
   ```

Note It is up to the application being initiated to delete the shared transient **HugeStringArray** instance.

10. Click the **OK** button to run the selected application. (Alternatively, click the **Cancel** button to abandon your selections.) Your selected application is then initiated, and the following actions are performed.

    - If the **New Copy** check box is unchecked and the application is currently active, focus is set to the active application.
    - If the **Set As Default** check box is checked, the Application Browser is updated to reflect the new current (default) application.

- If the **Check For Uncompiled Methods** check box is checked and the application contains any methods that have not yet been compiled or that are locked for editing, the message box shown in the following image is then displayed.



If you want to compile or save the specified method before running the application, click the **No** button and compile or save the method before you retry running the application. Alternatively, click the **Yes** button if you want to run the application with an uncompiled method or one that is locked for editing.

What Happens Next

When you have started your JADE application, the start-up form for that application is then displayed (for example, a log-in screen or a menu), if you are running a GUI application.

Running a Non-GUI Client Application using jadclient

The **jadclient** executable program enables you to automate the running of non-GUI client applications. You can run a non-GUI client JADE application (defined as application type **Non-GUI** in the Define Application dialog) as a service under Windows Enterprise (that is, Windows 10, Windows 8, Windows 7, Windows Server 2016, Windows Server 2012, or Windows Server 2008) or you can run it from the command line, to add processing nodes to your JADE network.

Notes To debug a non-GUI application, you must first change the application type to **GUI, No Forms** (by using the **Application Type** combo box in the **Application** sheet of the Define Application dialog).

The **jadclient** executable program uses the specified **schema** and **app** values, in conjunction with the optional **signonUser** and **signonPassword** values, to sign on to the JADE database. It performs no user logic in this sign-on process and it signs off as soon as it has issued an **Application** class **startApplication** or **startApplicationWithParameter** method call, which is the process that actually runs the user logic.

The **jadclient** executable program will not run server applications specified in a JADE initialization file **ServerApplication*<application-number>*** parameter when it is running an application defined in a system schema (for example, user-defined server applications are not started when **schema=RootSchema** and **app=JadeReorgApp**).

If you are a standard user, you cannot install or remove an application as a service, as you do not have the necessary privileges. (For details, see "Service Management", in the *JADE Platform Differences Guide*.)

The **jadclient** executable starts the application specified in the **app** argument. The specified application may start other applications or they may be started by the **ServerApplication*<application-number>*** parameter in the [NonGuiClient] section of the JADE initialization file. **jadclient** waits for *all* running applications to terminate before exiting back to the command line prompt, which allows application **A** to start **B**, then have **A** terminate but **B** to continue until it has performed the tasks for which it was written.

Run the non-GUI client program (for example, from a command script), specifying the following.

```
jadclient schema=schema-name
          app=application-name
          path=database-path
          ini=JADE-initialization-file-path
          [server=multiUser|singleUser]
          [causeEventOnSystem=[cause-event-number]]
          [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]
          [service=install|remove|cmdline]
          [nodeName=name-of-service]
          [nodeNameDescription="descriptive-name-of-service"]
          [noReorgRecovery=true|false]
          [delta=delta-name]
          [executeClass=class-name]
          [executeMethod=method-name]
          [executeParam=string]
          [executeSchema=schema-name]
          [executeScript=file-name]
          [executeTransient=true|false]
          [executeTypeMethod=true|false]
          [signonPassword=password]
          [signonUser=user-name]
          [startAppParameters [user-defined-arguments] [endAppParameters]]
          [endJade [user-defined-arguments]]
```

Notes The **jadclient** program treats processing arguments enclosed in double (**" "**) or single (**' '**) quotation marks after the **startAppParameters** or the **endJade** argument as single-string entries in the huge string array. The handling of strings in this huge string array is application-specific. For example, **path= "program files"** is treated as a two-string entry and **"path= program files"** is treated as a one-string entry. How these entries are handled is determined by your application.

All subsequent arguments up to the **endAppParameters** argument or the end of the command line are collected into a **HugeStringArray** and passed to the **initialize** method. Support for the **endJade** argument is maintained to provide backward compatibility for existing applications. It is recommended that your applications use the **startAppParameters** and **endAppParameters** arguments.

You can call the **Node** class or **Process** class **getCommandLine** method to return a string containing the current command line of the respective node or process of the receiver.

The following are examples of the command prompt that runs a non-GUI client application.

```
jadclient schema=ASchema app=NonGUIApp ini=c:\jade\test\jade.ini
path=c:\jade\system delta=foxtrot

jadclient schema=ASchema app=CSharpGenerator ini=c:\jade\test\jade.ini
path=c:\jade\system delta=foxtrot
```

The following example specifies the non-GUI client executable (**jadclient.exe**) command line for the **GUI No Forms JadeUnitTestGuiNoForms** application.

```
jadclient path=c:\jade\system schema=TestUnitTestSchema ini=c:\jade\system\jade.ini
app=JadeUnitTestGuiNoForms startAppParameters c:\jade\unittest.xml DailyTests
```

```
codecoverage=true output=true profile=true c:\jade\logs\unittest.log
endAppParameters
```

You can run the **ExtractAllExposures** application from the **jadclient** command prompt to generate C# code for all classes in a schema. The generated code for each class contains get and set methods for every property in every class. (Note that methods and constants are not included in this generation.) The following is an example of the command prompt that generates code for the .NET interface.

```
jadclient path=c:\jade\system ini=c:\jade\system\jade.ini app=ExtractAllExposures
schema=JadeSchema startAppParameters FCSchema FCSchema e:\temp\fcschema
endAppParameters
```

The parameters following the **startAppParameters** argument specify the schema name, the namespace, and the folder to which the .NET exposures are extracted. The **ExtractAllExposures** application exit code returns:

- **0**, if successful
- **1**, if the three parameters are not specified
- **2**, if the specified schema does not exist
- **3**, if the path name is invalid

For details about .NET exposures, see "Extracting a C# Exposure", in Chapter 17 of the *JADE Development Environment User's Guide*.

The following is an example of the command prompt that installs a non-GUI client application as a service.

```
jadclient service=install nodeName=bgapp1 nodeNameDescription="NonGuiApp in
ASchema on c:\jade\system" schema=ASchema app=NonGUIApp ini=c:\jade\test\jade.ini
path=c:\jade\system
```

The following is an example of the command prompt that removes a non-GUI client application service.

```
jadclient ini=c:\jade\test\jade.ini service=remove nodeName=bgapp1
```

The following is an example of the command prompt that externally causes an event on the JADE **System** object.

```
jadclient path=c:\jade\system ini=c:\jade\test\jade.ini causeEventOnSystem=12345
```

A progress report is created as the server node initializes. Standard initialization information is output to **stdout** and error information is output to **stderr**. For details about displaying and redirecting the output from JADE batch utilities, see the **DisplayApplicationMessages**, **LogServer**, and **UseLogServer** parameters under "JADE Log Section [JadeLog]", in the *JADE Initialization File Reference*.

Use the **ReadWriteStdio** parameter in the [NonGuiClient] section of the JADE initialization file to specify how JADE **read** and **write** instructions are handled when running the non-GUI client application. By default, Jade Interpreter Output Viewer and User Input windows are not displayed, and the output of **write** statements is directed to the Windows Command window. Set the parameter to **false** if you want the Jade Interpreter Output Viewer and User Input windows displayed as for a standard GUI application.

If the **jadclient** program fails, a non-zero exit code is returned and an error message is displayed; for example, if the schema is invalid or you attempt to run a GUI application.

The **jadclient** program uses the parameters in the JADE initialization file [NonGuiClient] section.

If you have installed your non-GUI client application as a service, you can control it by using standard Windows features. (For details, see "Controlling Non-GUI Client Application Services", later in this chapter.)

Note If you are a standard user, you cannot install or remove an application as a service, as you do not have the necessary privileges. (For details, see "Service Management", in the *JADE Platform Differences Guide*.)

You can also initiate a non-GUI application when the server is initiated or at a specific time. For details, see "ServerApplication<*application-number*>" under "JADE Non-GUI Client Section [NonGuiClient]", in the *JADE Initialization File Reference*.

The non-GUI client program arguments are described in the following subsections.

For details about using non-GUI client (**jadclient**) applications to:

- Automate the extraction of schemas, see "Extracting Schemas as a Non-GUI Client Application", in Chapter 10 of the *JADE Development Environment User's Guide*.
- Automate the generation and extraction of a C# exposure, see "Automating C# Exposure Generation and Extraction", in Chapter 17 of the *JADE Development Environment User's Guide*.
- Deploy schema changes directly to an application server running in single user mode. For details and an example, see "JadeSchemaLoader Application", in the *JADE Schema Load User's Guide*.
- Convert a database from one operating system or hardware platform to another or to convert the data between ANSI and Unicode formats, see "Converting a User Database", in Chapter 4.
- Reorganize a schema from the command line, see "Reorganizing the Database from the Command Line", in Chapter 3 of the *JADE Development Environment User's Guide*.
- Create a JCF file that includes all excluded table and columns for an RPS mapping that you can then use as input to the batch JADE Schema load utility (**jadloadb**) to reapply all excluded RPS mapping tables and columns to a deployed application, see "Site-Specific RPS Mapping Customization", in Chapter 2 of the *JADE Synchronized Database Service (SDS) Administration Guide*.
- Extract and load a patch history, see "Extracting and Loading a Patch History", in Chapter 3 of the *JADE Development Environment Administration Guide*.
- Automate the batch running of unit tests to improve the quality and reliability of the JADE applications, see "Running Unit Tests In Batch Mode" under "Using the JADE Testing Framework", in Chapter 17 of the *JADE Developer's Reference*.

See also "Running a JADE Non-GUI Client Application with Parameters", "Passing Parameters to Non-GUI Applications using jadclient", "Inserting a Schema into the Schema Hierarchy", "Reblocking Collection Class Maps", "Ad Hoc Index Batch Interface", "Stripping Method Source Code", and "Unloading All Report Writer Data to a Single File", later in this section.

If you reimplement the **Global** class **getAndValidateUser** and **isUserValid** methods for user validation, consider that these methods will be called in non-GUI applications. Creating and attempting to show forms in non-GUI applications raises an exception unless the application is in exception state.

schema

The **schema** argument specifies the schema in which the non-GUI application is defined.

This argument must have a valid name, and it must be specified if you are installing a non-GUI client application service or you are running the **jadclient** program from the command prompt.

Note This argument is optional if you have specified a **service** argument value of **remove**.

app

The **app** argument specifies the name of the non-GUI application that you want to run.

This argument must be an application of type **ApplicationType_Non_GUI_Web** or **ApplicationType_Non_GUI** that is defined in the **schema** argument, and it must be specified if you are installing a non-GUI client application service or you are running the **jadclient** program from the command prompt.

Note This argument is optional if you have specified a **service** argument value of **remove**.

path

The **path** argument specifies the full path of your JADE database directory in which your JADE database files are located.

This argument must be a valid existing JADE database path, and it must be specified if you are installing a non-GUI client application service or you are running the **jadclient** program from the command prompt.

If no directory is specified for the path, it is assumed to be under the JADE HOME directory on the server node. For example, if your installation directory is **Jade\bin** (that is, your JADE HOME directory is **Jade**) and you specify **path=system**, the full directory path is **Jade\system** on the server.)

Note This argument is optional if you have specified a **service** argument value of **remove**.

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

- A relative path name with a single leading slash (*/*) character is pre-pended by the first two characters of the JADE HOME directory (that is, ***drive-letter:***).
- Path names with no leading slash character are pre-pended by the JADE HOME directory. The JADE HOME directory is assumed to be the parent of the **bin** directory.

In the following examples, the JADE HOME directory is assumed to be **c:\jade**.

Path Specified in the Command Line	Actual Path
path=/jade2018/system	c:/jade2018/system
path=system	c:/jade/system

ini

The **ini** argument enables you to specify the fully qualified name of your JADE initialization file if it is not located in the database directory or it has a file name other than the default value of **jade.ini**.

server

Specify the optional **server** argument only if you want to run the non-GUI client application in **singleUser** mode. If you do not specify the **server** argument, the non-GUI client application runs in **multiUser** mode.

Use the server URI string in the **server** argument of a command line, to specify the target database and the client-server transport. For details, see "Format of the Server URI String", in Chapter 2 of the *JADE Installation and Configuration Guide*.

causeEventOnSystem

The optional **causeEventOnSystem** argument enables you to cause an external event on the JADE **System** object. For example, you can use this feature to shut down JADE systems from a batch process.

Note With the exception of the **path** and **ini** arguments, all other command line arguments are ignored if you specify them in the command line in conjunction with the **causeEventOnSystem** argument.

You can optionally define a numeric value (in the range **16** through **2**G bytes) that specifies the user event caused on the **System** object.

If the **jadclient** program does not detect a user event number specified in the command line, it checks in the [JadeClient] section of the JADE initialization file for an **ExternalEventOnSystem** parameter. If neither of these parameters is found, the event defaults to the maximum value of 2G bytes.

If you specify a number for the user cause event, the specified event value must match the value specified in the **eventType** parameter in the **Object::beginNotification** method. (The **Object**::**causeEvent** method is on the **System** object and it is immediate.)

The signature of the **beginNotification** method must subscribe to the **System** object, in the following format.

```
beginNotification(system, cause-event-number, Response_Continuous, 1);
```

Tip If you use the **causeEventOnSystem** argument to shut down a system, your JADE system must also have a method that executes the **terminate** instruction when the user notification of the specified event is received.

host

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

port

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

interface

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

localport

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

service

The optional **service** argument enables you to install or remove the non-GUI client application as a service or it runs the non-GUI client application service from the command prompt even if the application is installed as a service.

The valid values for this argument are listed in the following table.

Value	Instructs the jadclient program to …
install	Install itself as a service, which can then be controlled by standard Windows features
remove	Remove itself as a service
cmdline	Run from the command prompt, even if the non-GUI client application is installed as a service

When the **jadclient** program is installed as a service, the **RunAsService** parameter in the [NonGuiClient] section of your JADE initialization file is set to **true** or it is set to **false** when the service is removed. By default, non-GUI client applications are not installed as services.

See also "Service Management", in the *JADE Platform Differences Guide*.

nodeName

The optional **nodeName** argument specifies the name given to the non-GUI client application when it is run as a service.

Note This argument is mandatory if you have specified **remove** in the **service** argument.

If this argument is not defined in the **NodeName** parameter in your JADE initialization file when you run a non-GUI client application with the **service** argument set to **install**, JADE creates the **jadclient** default value in your initialization file. This value updates the **NodeName** parameter in the [NonGuiClient] section of your JADE initialization file before it attempts to install or remove the service.

The installation of a non-GUI client service fails if the **NodeName** argument value is already in use; that is, each active service must have a unique **nodeName** argument value. If you want to run more than one non-GUI client service concurrently, you must therefore have a separate initialization file for each service.

The length of the service name cannot be greater than 100 characters.

nodeNameDescription

The optional **NodeNameDescription** argument specifies the description given to the non-GUI client application when it is run as a service. (This descriptive name is displayed in the Services dialog under a Windows operating system that supports services.)

This value updates the **NodeNameDescription** parameter in the [NonGuiClient] section of your JADE initialization file before it attempts to install or remove the service. If this parameter is not defined in your JADE initialization file when you run a non-GUI client application with the **service** argument set to **install**, JADE creates the **Jade nonguiClient –** ***node (path)*** default value in your initialization file, using the **nodeName** argument value for the *node* and the **path** argument value from the **jadclient** program command line for the *path*.

The installation of a non-GUI client service fails if the **NodeNameDescription** argument value is already in use; that is, each active service must have a unique **nodeNameDescription** argument value. If you want to run more than one non-GUI client service concurrently, you must therefore have a separate initialization file for each service.

The length of the service description cannot be greater than 256 characters.

noReorgRecovery

The optional **noReorgRecovery** argument specifies whether the creation of temporary backups (**.bak** files) of the original database files (**.dat** files) is disabled when a reorganization takes place.

For details, see "Replayable Reorganizations" under "Reorganization Options", in Chapter 14 of the *JADE Developer's Reference*. (The default value for this parameter is **false**, which means that the temporary backup files are created.)

When you set the value to **true**, temporary backup files are *not* created. With this setting, if a reorganization failed, you would need to restore the system from a backup taken before the reorganization.

The following is an example of this argument specified in the command line when initiating the reorganization.

```
jadclient.exe path=C:\Jade\system ini=C:\Jade\system\jade.ini server=multiUser
app=JadeReorgApp schema=RootSchema startAppParameters
action=initiateReorgAllSchemas waitForReorg=true initiateTransition=false
```

```
replayableReorg=true reorgAllowUpdates=false noReorgRecovery=true
fastBuildBTreeCollections=false
```

The **noReorgRecovery** argument applies only to reorganizations that mutate objects or move objects instances between map files. It has no effect for file compaction and for re-indexing operations.

Caution Your system will not be recoverable if the reorganization fails and you do not have a pre-deployment backup.

Roll-forward recovery fails if this argument is set to **true** and a reorganization that was aborted is replayed. Replay on an SDS secondary database fails if this argument is set to **true** and a reorganization that was aborted is replayed.

delta

The optional **delta** argument specifies the name of the delta used when methods are executed. If a method has a version checked out in the specified delta, then that version is executed; otherwise the unchecked out version is executed.

For details about change control, see Chapter 2 of the *JADE Development Environment Administration Guide*.

executeClass

The **executeClass** argument specifies the name of the existing class in which an existing method or script specified in the **executeMethod**, **executeTypeMethod**, or **executeScript** parameter is to be executed in the non-GUI application specified in the **app** argument.

The default value is **JadeScript**. (See also the **executeSchema** and **executeTransient** arguments.)

Note Although this argument is optional with the **executeScript** argument, you must specify it when you specify the **executeMethod** or **executeTypeMethod** argument, or when you are running a JADE script in a subclass of the **JadeScript** class.

executeMethod

The optional **executeMethod** argument specifies the name of the existing method that you want to execute in the non-GUI application specified in the **app** argument.

Notes When you specify the **executeMethod** argument, you must also specify the **executeClass** argument with the name of an existing class in which the specified method is defined. If you do not specify a valid value in the **executeClass** argument, exception 1407 is raised.

If the specified method is a type method, you must also specify the **executeTypeMethod** argument with a value of **true**.

The **executeMethod** argument takes precedence over the **executeScript** argument if that is also specified with a valid file name value.

See also the **executeParam** argument.

executeParam

The optional **executeParam** argument specifies the appropriate single-string argument that you want passed to the method or script being executed in the non-GUI application.

See also the **executeMethod** and **executeScript** arguments.

executeSchema

The optional **executeSchema** argument specifies the name of an existing schema in which the method or script value of the respective **executeMethod**, **executeTypeMethod**, or **executeScript** argument is defined if it is a schema other than that specified in the mandatory **schema** argument.

executeScript

The optional **executeScript** argument specifies the name of the existing valid file that you want to execute in the non-GUI application specified in the **app** argument.

Note The **jadclient** executable requires the **executeScript** argument with a specified valid file name if you execute a method by using the **Process** class **executeScript** method.

If you do not specify the **executeClass** argument with the name of a valid class in the application, the script is expected to be defined in the default **JadeScript** class.

If the **executeMethod** argument is also defined, that argument takes precedence over the **executeScript** argument.

See also the **executeParam** argument.

executeTransient

The optional **executeTransient** argument specifies whether a transient instance of the class specified in the **executeClass** argument is created.

Set the value of the **executeTransient** argument to **true** if you want to create and use a transient instance of the class specified in the **executeClass** argument, if it exists.

When you specify the **executeTransient** argument with a value of **false**, the method or script specified in the **executeMethod** or **executeScript** argument is executed on the first instance of the class defined in the **executeClass** argument, if it exists.

executeTypeMethod

The **executeTypeMethod** argument specifies that the method specified in the **executeMethod** argument is a type method.

The values for this argument are **false** (the method is not a type method) or **true** (the method is a type method). The default value is **false**.

signonPassword

The optional **signonPassword** argument specifies the sign-on password when executing existing methods or reading, compiling, and executing transient methods. (See also the **signonUser** argument.)

Note When you specify the **signonPassword** argument, you must also specify the **schema** and **app** arguments in the **jadclient** command line (and optionally the **executeSchema** argument), as they specify the schema and application to which the **jadclient** executable does a **jomSignOn** call.

Although the **signonUser** and **signonPassword** arguments are optional, your application logic (the appropriate authentication rules in the **Global** class **getAndValidateUser** or **isUserValid** method) may require these arguments for signing on to the application specified in the **app** argument. See also "Getting Non-GUI Client Sign-On Details".

Consider that the global validation methods will be called in non-GUI applications, in which an exception is raised when creating and attempting to show forms unless the application is in exception state. In the **getAndValidateUser** method, check if the type of application is non-GUI. If so, specify your required user code and password; for example, you could set the user code to the application name and the password to the current schema name.

Your **isUserValid** method can then check that the combination of non-GUI application type, user code, and password are valid, to protect against running non-GUI applications that are defined in the schema and are not intended to be run in production.

signonUser

The optional **signonUser** argument specifies the sign-on user code when executing existing methods or reading, compiling, and executing transient methods. (See also the **signonPassword** argument.)

Note When you specify the **signonUser** argument, you must also specify the **schema** and **app** arguments in the **jadclient** command line (and optionally the **executeSchema** argument), as they specify the schema and application to which the **jadclient** executable does a **jomSignOn** call.

Although the **signonUser** and **signonPassword** arguments are optional, your application logic (the appropriate authentication rules in the **Global** class **getAndValidateUser** or **isUserValid** method) may require these arguments for signing on to the application specified in the **app** argument. See also "Getting Non-GUI Client Sign-On Details".

Consider that the global validation methods will be called in non-GUI applications, in which an exception is raised when creating and attempting to show forms unless the application is in exception state. In the **getAndValidateUser** method, check if the type of application is non-GUI. If so, specify your required user code and password; for example, you could set the user code to the application name and the password to the current schema name.

Your **isUserValid** method can then check that the combination of non-GUI application type, user code, and password are valid, to protect against running non-GUI applications that are defined in the schema and are not intended to be run in production.

startAppParameters and endAppParameters

The optional **startAppParameters** and **endAppParameters** arguments specify the start and end of a collection of command line parameters into a huge string array that is passed to the **initialize** method of the application being run.

All subsequent parameters up to the **endAppParameters** argument or the end of the command line are collected into a **HugeStringArray** and passed to the **initialize** method.

Note When you specify the **startAppParameters** argument, you must also specify the **schema** and **app** arguments in the **jadclient** command line (and optionally the **executeSchema** argument), as they specify the schema and application to which the **jadclient** executable does a **jomSignOn** call.

endJade

Note Support for the **endJade** argument is maintained to provide backward compatibility for existing applications. It is recommended that your applications use the **startAppParameters** and **endAppParameters** arguments.

Specify the optional **endJade** argument if you want to specify your own parameters (that is, user-defined command line arguments) in addition to the JADE-specific ones that are required. If you specify this argument, it must be the last of the JADE arguments before the first of your own parameters.

Controlling Non-GUI Client Application Services

You can run a non-GUI client application as a service only under Windows Enterprise. See also "Service Management", in the *JADE Platform Differences Guide*. 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 a non-GUI application is started before the database, a message is recorded in the JADE log file.

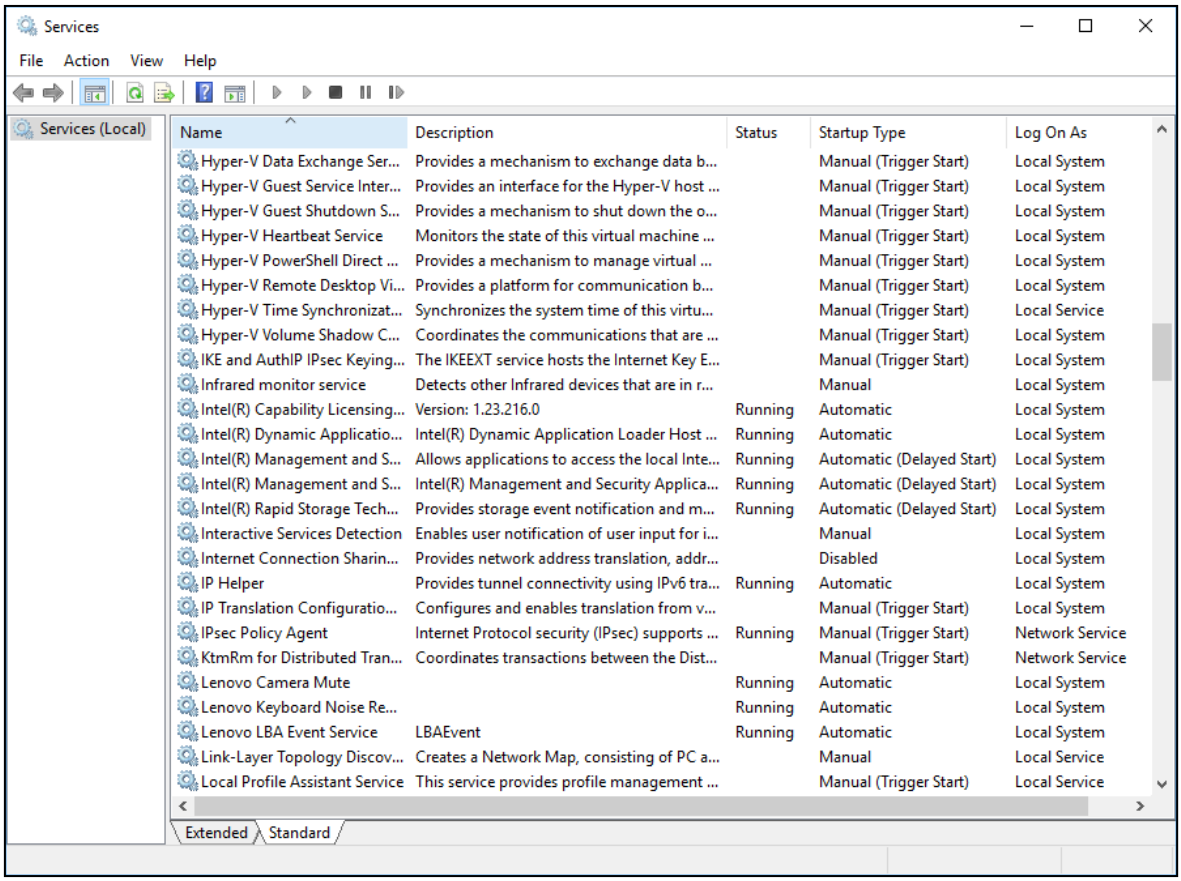
If you want to install multiple non-GUI client applications as services on the same host, you must have a separate JADE initialization file for each non-GUI client application service, because the service installation process must update a unique **NodeName**, **NodeNameDescription**, and **RunAsService**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

When you have installed a non-GUI client application as a service, you can then control the service by using standard Windows features.

To control an installed service using Windows Enterprise features

- Use the **Services** dialog to start and stop the service. In Windows Enterprise, select the **Administrative Tools** Control Panel applet and then select the **Services** applet from the Administrative Tools window that is displayed.

The following image shows an example of the Services window.



The non-GUI client application name is the value of the **NodeNameDescription** parameter in the [NonGuiClient] section of the JADE initialization file or specified in the **jadclient** program command.

- Use the standard Windows Enterprise Control Panel Services window to specify that you want the service to be started manually or to start or stop an installed service.
- Issue the **net start** or **net stop** command from a command prompt.

Passing Parameters to Non-GUI Applications using jadclient

For details about running the **jadclient** program to execute a non-GUI application within your JADE code, see "Running a Non-GUI Client Application using jadclient". (For details about collecting command line arguments into a huge string array that is passed to the **initialize** method of the application, see "Running a JADE Non-GUI Client Application with Parameters", in the following topic.)

To run a non-GUI application defined in your JADE schema, specify the following arguments in the **jadclient** program.

```
jadclient path=database-path
          ini=jade-initialization-file
          schema=schema-name
          app=application-name
          startAppParameters
          [command-line-arguments]
```

Enclose any command line arguments that contain spaces in double (**" "**) or single (**' '**) quotation marks.

Command line arguments specified after the **startAppParameters** argument are passed as a huge string array to the **initialize** method of your application, which must have a signature that contains only an **initializeParameter** parameter; that is:

```
method-name(initializeParameter: Object);
```

An exception is raised if the **initialize** method does not have one parameter only, of type **Object**.

If you do not specify any command line arguments, the **initialize** method signature of your application should not have any parameters. Each argument is limited to the maximum size of a string in a huge string array, which is currently **2047** characters.

Note The **jadclient** program treats processing arguments enclosed in double (**" "**) or single (**' '**) quotation marks after the **startAppParameters** argument as single-string entries in the huge string array. The handling of strings in this huge string array is application-specific. For example, **path= "program files"** is treated as a two-string entry and **"path= program files"** is treated as a one-string entry. How these entries are handled is determined by your application.

The following example shows an **Application** class initialization method that writes a **jadclient** command line to the Jade Interpreter Output Viewer.

```
jadclient(obj: Object);
vars
    cmdLine : HugeStringArray;
    count   : Integer;
begin
    if obj = null then
        write "No command line";
    else
        cmdLine := obj.HugeStringArray;
        foreach count in 1 to cmdLine.size do
            write "Arg[" & count.String & "] = '" & cmdLine[count] & "'";
        endforeach;
    endif;
    terminate;
end;
```

The following example shows the use of the **jadclient** program to output six arguments to the Jade Interpreter Output Viewer, based on the **Application**::**jadclient** method in the previous example.

```
c:\jade\bin\jadclient path=c:\jade\system ini=c:\jade\system\jade.ini
schema=TestClient app=TestClientA startAppParameters 1 2 three ix "a b" "Second
string but not in the violas "
```

The command line arguments in this example are passed as a huge string array to the **Application**::**jadclient** method in the example earlier in this subsection, resulting in the following displayed in the Jade Interpreter Output Viewer.

```
Arg[1] = '1'
Arg[2] = '2'
Arg[3] = 'three'
Arg[4] = 'ix'
Arg[5] = 'a b'
Arg[6] = 'Second string but not in the violas'
```

See also "Extracting Schemas as a Non-GUI Application", in Chapter 10 of the *JADE Development Environment User's Guide*.

Running a JADE Non-GUI Client Application with Parameters

You can run any application type run from the non-GUI client, including a standard (fat) client, a thin (presentation) client, non-GUI applications, and so on, using the **jadclient** executable.

Note Not all **RootSchema** applications can be invoked using this functionality.

When initiating a JADE client application using the non-GUI **jadclient** executable, you can collect strings from the command line and pass them to the initialization method of the application, by specifying the arguments after the **startAppParameters** or **endJade** argument in the command line. Only the application initiated using the command line argument uses these command line arguments.

When running a non-GUI JADE client application using the **jadclient.exe**:

- The arguments recognized in the command line are **startAppParameters** and **endAppParameters**.
- The elements between the **startAppParameters** and **endAppParameters** arguments are added to a transient **HugeStringArray** object that is created.

  The added elements are a unary string (for example, **DailyTests**) or a coupled pair separated by a **=** character (for example, **codeCoverage=true**). For example, **startAppParameters c:\jade\unittest.xml DailyTests codeCoverage=true profile=true c:\jade\logs\unittest.log endAppParameters** results in a **HugeStringArray** of:

  a. c:\jade\unittest.xml

  b. The test name (that is, DailyTests)

  c. codeCoverage=true

  d. profile=true

  e. c:\jade\logs\unittest.log

- The built **HugeStringArray** object is passed to the defined initialize method for the invoked application. That method must therefore expect a parameter of a **HugeStringArray** type. If not, exception 4027 *(Method called with incorrect number of parameters)* is generated for the parameter mismatch.

When processing the elements after the **startAppParameters** argument:

- Any double or single quotes around elements are dropped.
- Any spaces just before and directly after the **=** sign are dropped.
- All subsequent arguments up to the **endAppParameters** argument or the end of the command line if the **endAppParameters** argument is not present are collected into a **HugeStringArray** and passed to the initialize method of the application.

  Support for the **endJade** argument is maintained to provide backward compatibility for existing applications. It is recommended that your applications use the **startAppParameters** and **endAppParameters** arguments.
- The elements are separated by spaces.
- Leading and trailing spaces in each string element are removed.

Note also:

- If no such command line arguments are specified, the initialize method is passed a null value if it requires a parameter.
- The created transient object is deleted when the initialize method call completes.
- This feature applies only to the application being initiated from the command line of **jadclient**.

The following example specifies the non-GUI client JADE executable (**jadclient.exe**) command line for the non-GUI **JadeUnitTestBatch** application.

```
jadclient path=c:\jade\system schema=TestUnitTestSchema ini=c:\jade\system\jade.ini
app=JadeUnitTestBatch startAppParameters c:\jade\unittest.xml DailyTests
codecoverage=true profile=true output=true c:\jade\logs\unittest.log
endAppParameters
```

The following example specifies the non-GUI client JADE executable (**jadclient.exe**) command line for the **GUI No FormsJadeUnitTestGuiNoForms** application.

```
jadclient path=c:\jade\system schema=TestUnitTestSchema ini=c:\jade\system\jade.ini
app=JadeUnitTestGuiNoForms startAppParameters c:\jade\unittest.xml DailyTests
codecoverage=true output=true c:\jade\logs\unittest.log
```

Getting Non-GUI Client Sign-On Details

Although the **signonUser** and **signonPassword** command line arguments are optional when executing existing methods or reading, compiling, and executing transient methods, the appropriate authentication rules in the **Global**::**isUserValid** method may require them for signing on to the application specified in the **app** command line argument.

If you do not want to specify the user identifier and password on the command line, you can use the **GetSignonAuthDetails** parameter in the [NonGuiClient] section of the JADE initialization file to specify your own input DLL. Use this parameter (whose default value of **<default>** indicates that the **signonUser** and **signonPassword** command line arguments are used, and if they are not present, the **JadClientTaskUser** and **JadClientTaskPassword** values are used) to specify the name of the input library file and optionally the function name, which defaults to **getSignonDetails**.

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 was unable to obtain the password.

```
extern "C" int JOMAPI
authFunction(GETPASSWORD_UNION * authInfo)
{
    authInfo->GETPASSWORD_STRUCTv2.bValid = true;

    intlStrcpy(authInfo->GETPASSWORD_STRUCTv2.username,  TEXT("ExampleUserCode"));
    intlStrcpy(authInfo->GETPASSWORD_STRUCTv2.password,  TEXT("ExamplePassword"));

  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 a third-party DLL, see the **GETPASSWORD_STRUCT** type definition in the **jomtypes.h** file in your JADE **includes** directory.

Inserting a Schema into the Schema Hierarchy

You can use the **jadclient** executable to insert one schema into the schema hierarchy above another schema. This inserts (or moves) an existing schema that has no subschemas to a new position in the schema hierarchy, enabling you to create the new schema in the hierarchy and then use a standard schema load after the insertion processing has completed to deploy the required classes.

Although it is preferable to insert an "empty" schema (that is, one that contains only the **Application**, **Global**, and **WebSession** subclasses, with no properties, constants, or methods on any of these classes), you *can* insert a non-empty schema but any entities that conflict with the schema branch into which it is inserted cause the operation to be cancelled.

Caution As a precaution, you should backup your database before inserting a schema and then perform a meta certify of your database after the changes are committed. For details, see "Using the Backup Database Command", in Chapter 1 of the *JADE Database Administration Guide* and Chapter 5, "JADE Logical Certifier Diagnostic Utility", of the *JADE Object Manager Guide*.

To insert a schema into the schema hierarchy, specify the following arguments in the **jadclient** program.

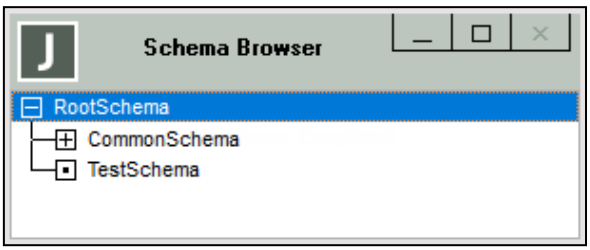
```
jadclient path=database-path
          ini=jade-initialization-file
          schema=RootSchema
          app=JadeInsertSchema
          server=singleUser
          startAppParameters
          superschema=superschema-name
          subschema=subschema-name
```

To insert a schema into the schema hierarchy, you must run the executable in single user mode. The **schema** command line argument value must be **RootSchema** and the **app** command line argument value **JadeInsertSchema**. For example, if the superschema is **TestSchema** and you want to insert it above **CommonSchema**, the **jadclient** command line is as follows.

```
jadclient path=c:\jade\system ini=c:\jade\system\jade.ini schema=RootSchema
app=JadeInsertSchema server=singleUser startAppParameters superschema=TestSchema
subschema=CommonSchema
```

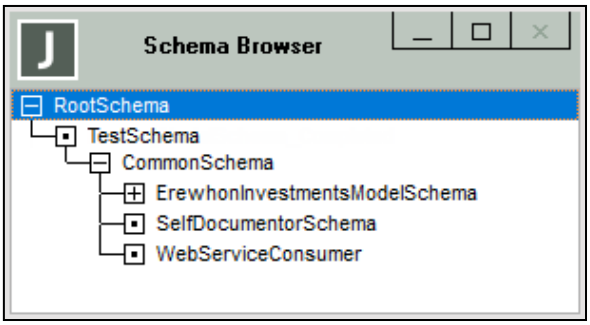
Note A reorganization is required as part of the schema insertion process.

Before inserting **TestSchema** above **CommonSchema**, the schema hierarchy would be like that shown in the following image.



The **JadeInsertSchema** application reports progress and errors to the **InsertSchema.log** file in your JADE logs folder. This file contains details of any conflicts that prevent the schema being moved.

After running the **JadeInsertSchema** application, the schema hierarchy would look like that shown in the following image.



The following restrictions apply to the current release of this feature.

- Neither the superschema nor the target schema can be versioned.
- If the superschema being inserted has subschema copy classes, the classes must exist in the new hierarchy at which it is being inserted.
- Where the superschema is not an "empty" schema, the following conflicts result in the operation being cancelled.
  - Class name conflicts
  - Constant category conflicts
  - Global constant conflicts
  - External function name conflicts
  - ActiveX library name conflicts
  - Database name conflicts
  - Delta id conflicts
  - Exported package name conflicts
  - Imported package name conflicts
  - External database name conflicts
  - Interface name conflicts
  - Method number conflicts (subschema copy class)
  - Library name conflicts
  - RPS database name conflicts
  - Relational view name conflicts
  - Schema view name conflicts
  - User format names conflicts

- Translatable string name conflicts
- Constant name conflicts (subschema copy class)
- Method signature conflicts (subschema copy class)
- Method name conflicts (subschema copy class)
- A package used by the inserted superschema that is defined by one of the schemas that will become a subschema of this schema
- A package defined in the superschema that is used by one of the schemas that will become a subschema of this schema

Reblocking Collection Class Maps

You can use the reblocking collection class maps facility after upgrading an environment from an earlier release that had a different file structure (for example, when upgrading JADE release 6.3 or 7.0 to 7.1), to reblock collections for the best fit to the new file structure.

You can reblock all collections in one or more map files, by using the **JadeCollectionReblocker** application in the **jadclient** executable in single user or multiuser mode; that is:

```
jadclient.exe path=database-path
              ini=jade-initialization-file
              server=SingleUser|MultiUser
              schema=user-schema-name
              app=JadeCollectionReblocker
              file=file-name|mask
              [file2=file-name [file3=file-name]...]
              [workers=number-of-worker-threads]
```

The *file-name* value is a complete map file name or a partial map file name with a trailing asterisk; for example, **cust*** means all map files that begin with the letters **cust**. Specify the *file-name* mask ***** (a single asterisk) to indicate reblocking of all user files, including **rootdef**. Specify the mask **_*** (underscore asterisk) to indicate the reblocking of all system files such as **_userscm**.

The following are examples of the reblocking action.

```
jadclient path=d:\jadeuser ini=d:\salesdb\jade.ini schema=Sales server=singleUser
app=JadeCollectionReblocker file=*

jadclient path=d:\jadeuser ini=d:\salesdb\jade.ini schema=Sales server=singleUser
app=JadeCollectionReblocker file1=testdb file2=banking file3=_userdev workers=5
```

The optional **workers** argument enables you to specify the number of multiple concurrent worker threads that are used. The default value of **1** is used if you do not specify the **workers** argument. You can specify a value in the range 1 through 16. If you specify a value greater than **16**, the maximum number of 16 worker threads is used.

Reblocking is done in normal update transactions, which are committed every 1,000 collections or 30 seconds.

Reblocking skips collections that do not require reblocking. The **JadeCollectionReblocker** application can be terminated part way through a file, in which case only uncommitted changes are discarded. When reblocking is restarted for the same file, collections that have already been reblocked are skipped.

Ad Hoc Index Batch Interface

You can use the **jadclient** executable RootSchema application **AdHocIndexBatchApp** to create indexes suitable for optimizing ad hoc queries without requiring database reorganization. For details about ad hoc indexes, see "Maintaining Ad Hoc Indexes", in Chapter 9 of the *JADE Development Environment User's Guide*.

The JADE initialization file can contain the [JadeAdHocIndex] section, which enables you to specify options for the worker applications that build, drop, and delete an ad hoc index, and for the controller application that starts worker applications when there is an ad hoc index maintenance operation to be performed. For details about the **BuildCommitPeriod** and **MaxBuildWorkers** parameters, see "JADE Ad Hoc Index Section [JadeAdHocIndex]", in the *JADE Initialization File Reference*.

Note The **AdHocIndexBatchApp** application specified in the **app** argument must be run in **multiUser** mode, because the commands initiate the controller application and its workers to perform the requests.

To execute the **AdHocIndexBatchApp** application from the non-GUI client program (for example, from a command script), specify the following.

```
jadclient schema=RootSchema
          app=AdHocIndexBatchApp
          server=multiUser
          path=database-path
          ini=JADE-initialization-file-path
          startAppParameters
          [command-line-arguments]
```

The list of command line arguments that you can define after the **startAppParameters** argument in ***command*=*value***; format are as follows.

- load=*file-name*

  Loads ad hoc index definitions from an Extensible Markup Language (XML) file. Ad hoc definitions cannot be changed if the collection has been built or it is being built. If the loaded definition is unchanged from an existing definition, the load is ignored.

- save=*file-name*

  Saves all ad hoc index definitions in all schemas in XML format to the specified file.

- build=*ad-hoc-index-name*|all

  Initiates the build of a specific ad hoc index or all definitions if **all** is specified. If the ad hoc index is already built, the request is ignored.

- cancel=*ad-hoc-index-name*|all

  Initiates the cancellation of a specific ad hoc index build or all builds that are in progress if **all** is specified. If the ad hoc index is not being built, the request is ignored.

- drop=*ad-hoc-index-name*|all

  Initiates the drop of a specific ad hoc index or all definitions if **all** is specified. If the ad hoc index is already dropped, the request is ignored.

- delete=*ad-hoc-index-name*|all

  Initiates the deletion of a specific ad hoc index or all definitions if **all** is specified.

- status=*schema-name*|all

Checks the available status of all ad hoc indexes for a specific schema or all schemas if **all** is specified. If all such ad hoc indexes are not available for use by ODBC, **jadclient** exits with an exit code of 1202. The request also fails if there are no defined ad hoc indexes for the specified schema.

Separate each command action with a semicolon followed by a space, as shown in the following example.

```
jadclient.exe schema=RootSchema app=AdHocIndexBatchApp server=multiUser
path=g:\jade\database\system ini=g:\jade\database\system\jade.ini
startAppParameters load=C:\adHoc Files\defn.xml; build=all;
```

The **jadclient** executable exits with exit code 1201 if the processing fails or is rejected or exit code 1202 if the **status** command is performed and the index or indexes are not active.

Stripping Method Source Code

The optional **executeMethod** argument enables you to specify the name of a user-defined schema whose JADE method source code you want stripped.

You must specify the **executeSchema** argument with a value of **RootSchema**, the **executeClass** argument with a value of **Schema**, the **executeMethod** argument with a value of **_removeSourceFromSchema**, and the **executeParam** argument with the name of your user schema from whose JADE methods you want source code stripped.

The syntax of the *execute* arguments in the **jadclient** command line is as follows.

```
jadclient path=database-path
          ini=initialization-file-name
          app=RootSchemaApp
          schema=RootSchema
          executeSchema=RootSchema
          executeClass=Schema
          executeMethod=_removeSourceFromSchema
          executeParam=schema-name
```

The following example strips the source from all JADE methods in the **TestBaseSchema** schema.

```
jadclient path=d:\jadesystems\jade\system ini=c:\jade\jade.ini app=RootSchemaApp
schema=RootSchema executeSchema=RootSchema executeClass=Schema
executeMethod=_removeSourceFromSchema executeParam=TestBaseSchema
```

An error is returned if the schema with the name specified in the **executeParam** argument does not exist or if it is a system schema.

Note As these arguments are case-sensitive, you must specify the JADE schema, class, method, and user schema names exactly. For example, **TestBaseSchema** is valid but if you were to specify **executeParam=Testbaseschema**, an error would be output.

You can specify only one user-defined schema for the removal of source code from all JADE methods. If you want to remove source code from methods in more than one schema (for example, subschemas and then their superschema), you must run separate copies of the **jadclient** program.

Unloading All Report Writer Data to a Single File

The optional **executeMethod** argument enables you to unload all JADE Report Writer data to a single flat file (for example, when upgrading from one JADE release to another from one JADE release to another JADE release in which the class number range has changed).

You must specify the **executeSchema** argument with a value of **JadeReportWriterSchema**, the **executeClass** argument with a value of **JadeReportWriterGlobal**, the **executeMethod** argument with a value of **unloadAllToFile**, and the **executeParam** argument with the name and output location of your report data file, which should have the default .**rwa** suffix, as shown in the following example.

```
jadclient path=d:\jade\system ini=d:\jade\myjade.ini app=RootSchemaApp
schema=RootSchema executeSchema=JadeReportWriterSchema
executeClass=JadeReportWriterGlobal executeMethod=unloadAllToFile
executeParam=d:\jade\rpts\alldata.rwa
```

The syntax of the *execute* arguments in the **jadclient** command line is as follows.

```
jadclient path=database-path
          ini=initialization-file-name
          app=RootSchemaApp
          schema=RootSchema
          executeSchema=JadeReportWriterSchema
          executeClass=JadeReportWriterGlobal
          executeMethod=unloadAllToFile
          executeParam=output-location\output-file-name-prefix.rwa
```

Note As these arguments are case-sensitive, you must specify the element values exactly, or an error is output.

Running a JADE Client Application using jade.exe

You can run any application type run from the client, including a standard (fat) client, a thin (presentation) client, non-GUI applications, and so on, using the **jade.exe** executable.

Note Not all **RootSchema** applications can be invoked using this functionality.

When initiating a JADE client application using the JADE executable (**jade.exe**), you can collect strings (arguments) from the command line and pass them to the initialization method of the application. Only the application initiated using the command line arguments uses these command line arguments.

» To run a JADE client application using the jade.exe

- The arguments recognized in the command lines are **startAppParameters** and **endAppParameters**.
- The elements between the **startAppParameters** and **endAppParameters** arguments are added to a transient **HugeStringArray** object that is created.

  The added elements are a unary string (for example, **DailyTests**) or a coupled pair separated by a **=** character (for example, **codeCoverage=true**). For example, **startAppParameters c:\jade\unittest.xml DailyTests codeCoverage=true profile=true c:\jade\logs\unittest.log endAppParameters** results in a **HugeStringArray** of:

  a. c:\jade\unittest.xml

  b. The test name (that is, DailyTests)

  c. codeCoverage=true

  d. profile=true

  e. c:\jade\logs\unittest.log
- The built **HugeStringArray** object is passed to the defined initialize method for the invoked application. That

method must therefore expect a parameter of a **HugeStringArray** type. If not, exception 4027 *(Method called with incorrect number of parameters)* is generated for the parameter mismatch.

When processing the elements after the **startAppParameters** argument:

- Any double or single quotes around elements are dropped.
- Any spaces just before and directly after the **=** sign are dropped.
- All subsequent arguments up to the **endAppParameters** argument or the end of the command line are collected into a **HugeStringArray** and passed to the initialize method.
- The elements are separated by spaces.
- Leading and trailing spaces in each string element are removed.

Note also:

- If no such command line arguments are specified, the initialize method is passed a null value if it requires a parameter.
- The created transient object is deleted when the initialize method call completes.
- This feature applies only to the application being initiated from the command line of **jade.exe**.

The following example specifies the JADE executable (**jade.exe**) command line for the non-GUI **JadeUnitTestBatch** application.

```
jade path=c:\jade\system schema=TestUnitTestSchema ini=c:\jade\system\jade.ini
app=JadeUnitTestBatch appServer=MyAppServer appServerPort=1234 startAppParameters
c:\jade\unittest.xml DailyTests codecoverage=true profile=true
c:\jade\logs\unittest.log endAppParameters
```

The following example specifies the JADE executable (**jade.exe**) command line for the **GUI No Forms JadeUnitTestGuiNoForms** application.

```
jade path=c:\jade\system schema=TestUnitTestSchema ini=c:\jade\system\jade.ini
app=JadeUnitTestBatchGuiNoForms appServer=MyAppServer appServerPort=1234
startAppParameters c:\jade\unittest.xml DailyTests codecoverage=true output=true
c:\jade\logs\unittest.log endAppParameters
```

The **JadeSchemaLoader** application enables you to deploy schema changes directly, without having to stop any applications that are running. For details and an example, see "JadeSchemaLoader Application", in the *JADE Schema Load User's Guide*.

Running JADE Production Mode Databases

Production mode is designed for use with JADE systems running in production where changes to the database occur only under managed conditions and performance may be improved at the cost of memory usage.

In production mode, internal structures for classes are not released when the class is no longer in use. Subsequent class usages reuse these structures, avoiding the overhead of allocating and opening the class. Classes in the schema remain in use on the node as long as an application running on that node has accessed the schema. For details about enabling or disabling production mode, see "Using the Production Mode Command", in Chapter 1 of the *JADE Database Administration Guide*. In production mode:

- Continuable exceptions cannot be ignored in the default Unhandled Exception dialog.

  The **Ignore** button is disabled on the default Unhandled Exception dialog (changing the behavior of the **Exception** class **showDialog** method).

- When performing a reorganization, the transition must be initiated in single user mode.

  When you start a production mode, single user, JADE development environment so that you can perform a reorganization, applications, **JadeScript** methods, and Workspaces should not be used before or after the reorganization, or exceptions may be raised.

  When the reorganization is complete, you can restart the JADE development environment.

- The Jade User Interrupt icon is not displayed.

Using the Jade User Interrupt

The Jade User Interrupt icon is created in the system tray at the right of the Taskbar by default when you run a JADE user application or **JadeScript**, to enable you to interrupt the application or script that is currently running.

Notes A Jade User Interrupt icon is created only for user-defined JADE applications (that is, a Jade User Interrupt icon is not created for system applications such as the **Jade** application itself or for the JADE Monitor) that are running in a Windows GUI environment. A Jade User Interrupt icon is not created if the database is running in production mode or if the **ShowUserInterrupt** parameter in the [Jade] section of the JADE initialization file is set to **false**.

If the **ShowUserInterrupt** and **UseSystemTrayIcon** parameters in the [Jade] section of the JADE initialization file are set to **true** and the **Group similar taskbar buttons** check box is checked on the **Taskbar** sheet of the Windows Taskbar and Start Menu Properties dialog, right-clicking on the Jade User Interrupt system menu within the group menu does nothing. You must use the left-click action instead.

The icon provides a menu only; it cannot be maximized to a window. The icon that is displayed is the icon of the earliest initiated application that is not using the default JADE icon. (The JADE icon is displayed if no icon is defined for the application.)

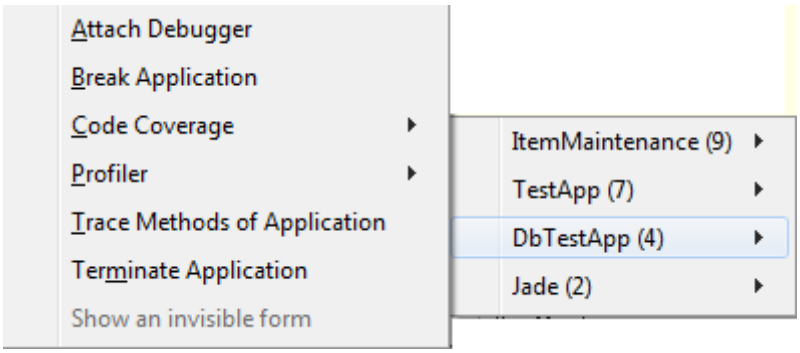
For details about the conditions under which this icon is displayed and configuring its location, see the JADE initialization file **ShowUserInterrupt** and **UseSystemTrayIcon** parameters, under "JADE Startup Section [Jade]", in the *JADE Initialization File Reference*.

Clicking this icon with a left or right mouse button displays the Jade User Interrupt system menu and when you move the mouse over the icon, bubble help is displayed that describes the database or application server that is in use.

In applications running on a standard (fat) client, the bubble help includes the database path. As the database path is not known on the presentation client, the bubble help for the Jade User Interrupt icon when an application is running in JADE thin client mode displays the following information.

```
Jade User Interrupt for: App Server:<name> App Server Port:<port>
```

The Jade Interrupt icon enables you to access the Jade User Interrupt system menu, shown in the following image.



The Jade User Interrupt menu lists the JADE applications that are currently running and the instance identifier (**instId**) of the associated JADE process, which enables you to distinguish between multiple copies of the same application running on the same client node (for example, is your want to determine the user who acquired a selected lock so that you can force off or interrupt the user). This identifier is the same as that shown in the JADE Monitor **Users** view.

Although the JADE application itself is listed, the only action that you can perform from the Jade User Interrupt menu is to terminate the JADE application. All other operations are disabled.

The Jade User Interrupt menu also contains submenus of the operations that you can perform on your applications.

Use the Jade User Interrupt to perform the actions described later in this section and listed in the following table.

Action	For details, see…
Attach Debugger	Attaching the Debugger to a Running Application
Break the process that is currently being executed in an application	Breaking a Process in an Application
Determines the degree to which the methods in your application have been executed	Determining Code Coverage
Profile an application	Profiling an Application
Enable or disable the tracing of application methods in an application	Tracing Methods in an Application
Terminate a selected application	Terminating an Application
Show an invisible form	Showing an Invisible Form

Notes The version of Windows that you are running determines the display of commands on the menu.

The icon is closed when there are no user applications currently running.

Accessing the Jade User Interrupt Menu

» To access the Jade User Interrupt menu

- Left-click or right-click on the Jade User Interrupt icon in the system tray at the right of the Taskbar.

The Jade User Interrupt menu is then displayed.

Selecting an Application

The Jade User Interrupt menu enables you to select the application that you want to administer. (As the JADE application is a system application, the only action that you can perform on the JADE application by using the Jade User Interrupt menu is to terminate it. All other operations are disabled.)

To select the application

- Select the application from the list in the Jade User Interrupt menu.

The submenu of operations that you can perform for the selected application is then displayed.

Attaching the Debugger to a Running Application

The **Attach Debugger** command in the Jade User Interrupt submenu enables you to dynamically attach the JADE debugger to a running application, provided that the JADE development environment is also being run by that client.

The Jade User Interrupt menu provides the **Attach Debugger** command for each application, which enables you to initiate the JADE debugger. The application execution halts in the debugger at the next debugged method logic statement.

By default, only methods that begin execution *after* the JADE debugger is attached can be debugged, because the JADE Interpreter executes methods using two different execution engines: normal execution and debugger execution. Normal execution is significantly faster and cannot be debugged. As a result, any methods already in execution will not halt on breakpoints or allow debug break and step. The exceptions to this are:

- If the debugger was attached to an application, the debugger is closed and then subsequently re-attached.
- Running the application in debug-ready mode; that is, by checking the **Run in Debug Ready Mode** check box on the JADE Run Application dialog, which runs the application in debug mode but without the debugger being initiated when this is checked. Attaching to that application then allows debugging of any method already on the execution stack. (Note that if the application is idle and no modal dialogs are displayed, attaching the debugger is equivalent to running in debug-ready mode.)

To attach a running application to the JADE debugger

- Select the **Attach Debugger** command in the Jade User Interrupt submenu.

If the application is in a loop, attaching the debugger breaks only if the application is debug-ready or if the logic begins execution of another user-defined method after the attach action.

If you use the Jade User Interrupt menu **Break Application** command and the logic being executed is debug-ready, it causes a debugger break if logic is being executed.

Breaking a Process in an Application

The **Break Application** command in the Jade User Interrupt menu enables you to select the application whose current process is to be broken, or interrupted; for example, to display an exception dialog to enable you to abort the action or to continue (resume) processing if the application is looping.

To break an application

1. Select the application whose processing is to be interrupted. The Jade User Interrupt submenu is then displayed.
2. Select the **Break Application** command from the submenu.

If the selected application is currently executing JADE logic, an exception dialog is displayed. If the selected application is not currently executing JADE logic, an exception dialog is displayed when the next JADE method starts execution.

The exception dialog enables you to abort the process or to continue processing.

Profiling an Application

The **Profiler** command in the Jade User Interrupt submenu enables you to record actual and total times spent in JADE and external methods in an application.

Note It is recommended that when investigating application performance, only one of the JADE Profiler, JADE Monitor, or method profiling is used at any one time, as the results reported when any of these are combined is undefined.

Selecting the **Start** command starts the recording of times spent in methods during the running of the selected application. These statistics continue to be recorded to memory until you terminate the application or you select the **Stop** command or the **Report** command.

The profile statistics are output to a file only when you select the **Report** command.

The statistics are cleared from memory when you perform one of the following actions.

- Select the **Reset** command.
- Report the profile statistics to file, by selecting the **Report** command.

» To access the Profiler submenu

- Select the **Profiler** command.

The Profiler submenu, containing commands that control the recording of your application profile statistics, is then displayed. Use these commands to optimize your JADE code, by analyzing the performance of your JADE and external methods.

The Profiler submenu commands are listed in the following table.

Method	Description	For details, see…
Start	Starts the capture of profile statistics for the selected application	Starting the Recording of Profile Statistics
Stop	Stops the capture of profile statistics for the selected application	Stopping the Recording of Profile Statistics
Reset	Clears profile statistics	Clearing Profile Statistics
Report	Outputs profile information about time spent in methods to a log file	Reporting Profile Statistics
Report CSV	Outputs profile information about time spent in methods to a comma-separated values file	Reporting Profile Statistics to a CSV File

For details, see the following subsections.

Note In the JADE development environment, you can use system methods provided by the **Process** class to control the recording of profile statistics in specific methods, if required. For details, see "Process Class", in Chapter 1 of the *JADE Encyclopaedia of Classes*.

Starting the Recording of Profile Statistics

The **Start** command in the Jade User Interrupt Profiler submenu starts recording profile statistics of the actual and total times spent in JADE and external methods during the running of the selected application. (To stop the recording of profile statistics, select the **Stop** command.)

These profile statistics are recorded until you output them to a file by selecting the **Report** or **Report CSV** command or you clear them by selecting the **Reset** command.

To start recording profile statistics

- Select the **Start** command from the Jade User Interrupt Profiler submenu.

The recording of the actual and total times spent in methods is then started.

Stopping the Recording of Profile Statistics

The **Stop** command in the Jade User Interrupt Profiler submenu stops the recording of the actual and total times spent in JADE and external methods for the selected application, started by the **Start** command.

These profile statistics are recorded until you perform one of the following actions.

- Output them to a file by selecting the **Report** or **Report CSV** command. (You must then select the **Start** command to restart the recording of profile statistics.)
- Clear them by selecting the **Reset** command. (Statistics then continue to be recorded until they are terminated by the **Stop** command or the **Report** or **Report CSV** command.)
- Terminate your application. (You must then select the **Start** command to start the recording of profile statistics when you next run that application.)

To stop recording profile statistics

- Select the **Stop** command from the Jade User Interrupt Profiler submenu.

The recording of profile statistics is then stopped.

Clearing Profile Statistics

The **Reset** command in the Jade User Interrupt Profiler submenu clears profile statistics of the actual and total times spent in JADE and external methods.

To clear profile statistics

- Select the **Reset** command from the Jade User Interrupt Profiler submenu.

Any profile statistics that have not yet been printed are then cleared from the output file (that is, they are deleted) and are unable to be printed. Recording of statistics continues until a **Start** command or a **Report** or **Report CSV** command is executed.

Note You can also clear profile statistics by selecting the **Report** or **Report CSV** command. (This action prints any statistics that are not yet printed, and then stops the recording of statistics.)

Reporting Profile Statistics

The **Report** command in the Jade User Interrupt Profiler submenu outputs profile statistics of times spent in JADE and external methods to a file. (In JADE thin client mode, profiler output is always output to the workstation that is running the JADE logic; that is, to the application server.)

To output profile statistics to a report file

- Select the **Report** command from the Jade User Interrupt Profiler submenu.

The profile statistics are then output to the**JadeProfiler_*application-name_yyyyMMdd_hhmmss*.log** file in the path specified by the **ResultsFile** parameter in the [JadeProfiler] section of the JADE initialization file, if any. A new report file is generated for each call to the **JadeProfiler** class **report** method. (For details about directing the report to a comma-separated values (.**csv**) file, see "Reporting Profile Statistics to a CSV File", in the following section.)

If a path is not specified in this parameter, the file is located in the log file directory, specified by using the **LogDirectory** parameter in the [JadeLog] section of the JADE initialization file. (For details, see Chapter 1 of the *JADE Initialization File Reference*, "JADE Initialization File".)

You can view your profile output by using a text editor or you can output it to a printer, if required.

The statistics provided by the profile report enable you to optimize the code of your methods in the JADE development environment by analyzing the performance of your methods at run time.

The report produced by this method can contain six sections, as follows.

- The "methods in actual time order" table lists all JADE and external methods that are called, in the order of the *actual* time spent in each method. This information contains:
  - The number of times that the method was called
  - The minimum, maximum, and average duration of each call (in milliseconds)
  - The percentage of the profiling time

  When running the application in multiuser mode, the schema, class, and method names are followed by the method execution location if the method executes on the server node; for example, **ErewhonInvestmentsViewSchema**::**Sale::loadData(server)**.

  No method location is output for methods executing locally. Details are output for both nodes if a method executes both locally and on the server.
- The "methods in total time order" table lists the methods that were called, in the order of the *total* elapsed time that was spent in each method.

  The time spent in methods called from each method is included in this table.

  Note It is recommended that when investigating application performance, only one of the JADE Profiler, JADE Monitor, or method profiling is used at any one time, as the results reported when any of these are combined is undefined.
- The "methods in total load time order" table lists the method *load* times, in the order of time each method took to load. This table lists the:
  - Total time taken to load the method
  - Number of times that the method was loaded

- Average load time (that is, total load time divided by the number of times that the method was loaded)
- Schema, class, and name of the method

Use this table to monitor the frequency of method loads so that you can increase the interpreter method cache size if methods are frequently being loaded and exceeding the method cache, by using the **MethodCacheLimit** parameter in the [JadeInterpreter] section of the JADE initialization file.

Note Methods already loaded in the cache before profiling started are not reported in this table. (You can use the **JadeProfiler** class **clearMethodCache** method to flush the cache.)

- The "methods ordered by method size" table lists the:
  - Size (in bytes) of the method in cache
  - Number of calls to the method
  - Number of times that the method was loaded
  - Hit rate percentage of the method (that is, the number of times the method was found in cache divided by the number of times the method was executed)
  - Schema, class, and name of the method

  In addition, this table contains the total size (in bytes) of all methods executed.

  On JADE client nodes, the interpreter method cache holds the code for methods executed on that node. Use the **MethodCache** parameter in the [JadeInterpreter] section of the JADE initialization file to allocate the number of method caches. (Multiple caches result in faster load and execution of JADE methods, especially on Symmetric Multiprocessing (SMP) nodes. However, this improved performance is achieved at the expense of an increased usage of physical memory.)
- The "cache statistics" section of profile information contains the method cache limit and the *total* maximum size to which the method cache grew during the profiling session.

  The cache statistics include the number of methods that were discarded from the cache to make room for new methods during the profiler run. If there were discarded methods, the total size of the methods discarded is also listed.

  If the method cache overflowed (that is, the cache size exceeds the maximum size specified and all methods in the cache were in use and could be discarded), a table lists the:
  - Amount by which the cache limit was exceeded, in ascending order of ten percentage points (for example, 10%, 20%, 30%, and so on up to 100+%, in units of 10 percentage points)
  - Size of the method cache at that level (that is, the cache limit plus the exceeded amount)
  - Number of times the cache was exceeded

  Tip When a JADE method is executed, the JADE interpreter must load the method code into cache for execution. If a method is called frequently, tuning the **MethodCacheLimit** parameter in the [JadeInterpreter] section of the JADE initialization file may result in substantial time savings.

  The "string pool statistics" provide the string pool limit, the maximum size to which the pool grew, and the number of method calls in which the limit was exceeded.

  For details about specifying the string pool limit, see the **StringPoolLimit** parameter in the [JadeInterpreter] section of the JADE initialization file, in the *JADE Initialization File Reference*.
- The "system statistics" table lists the global system-wide statistics for the duration of the profile session.

These values are those returned by the **System** class **getStatistics** or **getStatistics64** method and output in the Statistics window of the JADE Monitor. For details, see Chapter 1 of the *JADE Encyclopaedia of Classes*.

Note As statistics values are accumulated by the server, they include all system activity that occurred while the profiler was active.

If the application is running in multiuser mode and other users are accessing that application or any other application (regardless of the profiling setting), the system statistics therefore include all user operations for all applications that were running for the duration of the profile activity.

The system statistics are listed in the following table.

Statistic	Example
Committed transactions	64177
Aborted transactions	20
Get objects	201594
Queued locks	3
Objects created	532
Objects deleted	449
Objects updated	86609
Objects locked	226709
Objects unlocked	40647
Begin notifications	271
End notifications	183
Delivered notifications	597
Server method executions	3

When you select the **Report** command and there is an existing profile file, records are appended to existing records, indicated by start and finish times.

Reporting Profile Statistics to a CSV File

The **Report CSV** command in the Jade User Interrupt Profiler submenu outputs profile statistics of times spent in JADE and external methods to a comma-separated values (.**csv**) file. (In JADE thin client mode, profiler output is always output to the workstation that is running the JADE logic; that is, to the application server.)

» To output profile statistics to a CSV report file

- Select the **Report CSV** command from the Jade User Interrupt Profiler submenu.

The profile statistics are then output to the **JadeProfiler_<*application-name*>_*yyyyMMdd_hhmmss*.csv** file in the path specified by the **ResultsFile** parameter in the [JadeProfiler] section of the JADE initialization file, if any. A new report file is generated for each call to the **JadeProfiler** class **report** method.

If a path is not specified in this parameter, the file is located in the log file directory, specified by using the **LogDirectory** parameter in the [JadeLog] section of the JADE initialization file. (For details, see Chapter 1 of the *JADE Initialization File Reference*, "JADE Initialization File".)

You can view your profile output by using an Excel spreadsheet, from which you can output it to a printer, if required.

The statistics provided by the profile report enable you to optimize the code of your methods in the JADE development environment by analyzing the performance of your methods at run time.

For details about the report that is produced, see "Reporting Profile Statistics", in the previous section.

Determining Code Coverage

The **Code Coverage** command in the Jade User Interrupt submenu enables you to determine dynamically the degree to which the methods in your JADE applications have been executed without having to change application code. Use code coverage to:

- Discover methods and blocks of code that are not exercised by a set of tests
- Create tests that increase code coverage
- Quantify the overall code coverage of a system, which is one measure of quality
- Analyze JADE methods, to determine those lines of code that have been executed or not executed at all

Note When determining code coverage by using the Jade User Interrupt submenu, the application must already be running. Only those methods that are executed after code coverage has started are reported on.

Code coverage shows lines that have been executed to some degree (either fully or partially), or not executed at all.

The **Code Coverage** command dynamically enables code coverage, so that you do not have to change application code. You can then run the code coverage results application to load a code coverage output file, view the results, and optionally save the code coverage results text file as a comma-separated values (.**csv**) file that can be displayed in an Excel spreadsheet. (For details, see "Code Coverage", in Chapter 17 of the *JADE Developer's Reference*.)

Code coverage data is not recorded for server methods or system methods.

Selecting the **Start** command starts recording the blocks of code in JADE methods that are executed during the running of the selected application. These statistics continue to be recorded to memory until you terminate the application or you select the **Stop** command, the **Report** command, or the **View** command. The **View** command stops the code coverage session, automatically initiates the code coverage application, and displays the created code coverage result file.

The code coverage results are output to a file only when you perform one of the following actions.

- Select the **Report** command or the **View** command.
- Terminate your application.

The results are cleared from memory when you perform one of the following actions.

- Select the **Reset** command.
- Report the code coverage results to file, by selecting the **Report** command or the **View** command.

Note Code coverage is recorded in the method cache for each method when code coverage is enabled. Methods are not discarded from the cache if code coverage is present for the method unless code coverage is reset or a results file is created. Memory utilization for a process is greater if code coverage is enabled.

In addition, performance of a process is affected if code coverage is enabled.

To access the Code Coverage submenu

- Select the **Code Coverage** command in the Jade User Interrupt submenu.

The Code Coverage submenu is then displayed, containing the commands listed in the following table that control the recording of your application code coverage results.

Method	Description	For details, see…
Start	Starts the capture of code coverage results for the selected application	Starting Recording Code Coverage Results
Stop	Stops the capture of code coverage results for the selected application	Stopping Recording Code Coverage Results
Reset	Clears code coverage results	Clearing Code Coverage Results
Report	Outputs information about code coverage to a file	Reporting Code Coverage Results
View	Displays code coverage results in the application	Viewing Code Coverage Results

For details, see the following subsections.

Note In the JADE development environment, you can use system methods provided by the **JadeProfiler** class to control the recording of code coverage results, if required. For details, see Volume 1 of the *JADE Encyclopaedia of Classes* or "Code Coverage", in Chapter 17 of the *JADE Developer's Reference*.

Starting Recording Code Coverage Results

The **Start** command in the Jade User Interrupt Code Coverage submenu starts recording code coverage results the recording of times spent in JADE methods during the running of the selected application. (To stop the recording of code coverage results, select the **Stop** command.)

These code coverage results are recorded until you output them to a file by selecting the **Report** command, you display the results in the Code Coverage Results Browser by selecting the **View** command, or you clear them by selecting the **Reset** command or shutting down the application.

To start recording code coverage results

- Select the **Start** command from the Jade User Interrupt Code Coverage submenu.

The recording of code coverage is then started.

Stopping Recording Code Coverage Results

The **Stop** command in the Jade User Interrupt Code Coverage submenu stops the recording of times spent in JADE methods during the running of the selected application, started by the **Start** command.

These code coverage results are recorded until you perform one of the following actions.

- Output them to a file, by selecting the **Report** command or the **View** command. (You must then select the **Start** command to restart the recording of code coverage results.)
- Display the results in the Code Coverage Results Browser, by selecting the **View** command. (You must then select the **Start** command to restart the recording of code coverage results.)
- Clear them, by selecting the **Reset** command. (Results then continue to be recorded until they are terminated by the **Stop** command, the **Report** command, or the **View** command.)

- Terminate your application. (You must then select the **Start** command to start the recording of code coverage results when you next run that application.)

» To stop recording code coverage results, perform one of the following actions

- Select the **Stop** command from the Jade User Interrupt Code Coverage submenu.
- Shut down the application.

The recording of code coverage results is then stopped.

Clearing Code Coverage Results

The **Reset** command in the Jade User Interrupt Code Coverage submenu clears code coverage results for the selected application.

» To clear code coverage results

- Select the **Reset** command.

Any code coverage results that have not yet been printed or viewed are then cleared from the output file (that is, they are deleted) and are unable to be printed. In addition, the method cache is also cleared.

Recording of code coverage results continues until a **Start** command, a **Report** command, or a **View** command is executed.

Note You can also clear code coverage results by selecting the **Report** command or the **View** command, or by terminating the application.

Reporting Code Coverage Results

The **Report** command in the Jade User Interrupt Code Coverage submenu outputs code coverage results for the selected application to a file. In JADE thin client mode, code coverage results output is always output to the workstation that is running the JADE logic; that is, to the application server.

» To output code coverage results to a report file

- Select the **Report** command to output the results of the recording of times spent in JADE methods during the running of the selected application.

The code coverage results are then output to the code coverage file, which defaults to the ***application-name_timestamp*.ccd** file, where the *timestamp* value is in the **YYYYMMDD_hhmmss** format (for example, **erewhonshop_20090312_074611.ccd**). This generated file name is used if the value of the **codeCoverageFileName** property in the **JadeProfiler** class is null.

The report is output to the directory specified by the **CodeCoverageDirectory** parameter in the [JadeProfiler] section of the JADE initialization file, which defaults to **logs\CodeCoverage** (that is, the **CodeCoverage** subdirectory of **logs**). For details, see "JADE Initialization File" (that is, the *JADE Initialization File Reference*).

You can view your code coverage results output file by loading the report output file into the Code Coverage Results Browser. For details about viewing code coverage results, see "Viewing Code Coverage Results", in the next section, or "Code Coverage", in Chapter 17 of the *JADE Developer's Reference*.

Viewing Code Coverage Results

The **View** command in the Jade User Interrupt Code Coverage submenu enables you to view the contents of a code coverage results output file for the selected application. (For details about generating a code coverage results file, see "Reporting Code Coverage Results".)

The **View** command stops the code coverage session, automatically initiates the code coverage application, and displays the created code coverage result file.

Loading the results file into the Code Coverage Results Browser enables you to analyze the code coverage of your JADE methods. You can load one or more code coverage output files into the browser.

To display code coverage results in the Code Coverage Results Browser

- Select the **View** command from the Jade User Interrupt Code Coverage submenu.

The Code Coverage Results Browser is then displayed. For details, see "Code Coverage", in Chapter 17 of the *JADE Developer's Reference*.

Tracing Methods in an Application

The **Trace Methods of Application** command enables you to set or disable the tracing of methods entered, invoked, or exited from in the current application. The result of the method trace in the selected application is output to the Jade Interpreter Output Viewer.

To trace methods in an application

1. Select the application whose methods are to be traced. The Jade User Interrupt submenu is then displayed.
2. Select the **Trace Methods of Application** command from the submenu.

A check mark is then displayed to the left of the selected application in the Jade User Interrupt menu, indicating that methods in that application are being traced.

To disable (unset) the tracing of methods in an application

1. Select the application whose methods are no longer to be traced. The Jade User Interrupt submenu is then displayed.
2. Select the **Trace Methods of Application** command from the submenu.

The check mark is then removed from the left of the selected application in the Jade User Interrupt menu, indicating that methods are not traced when the application is run.

Viewing Traced Methods

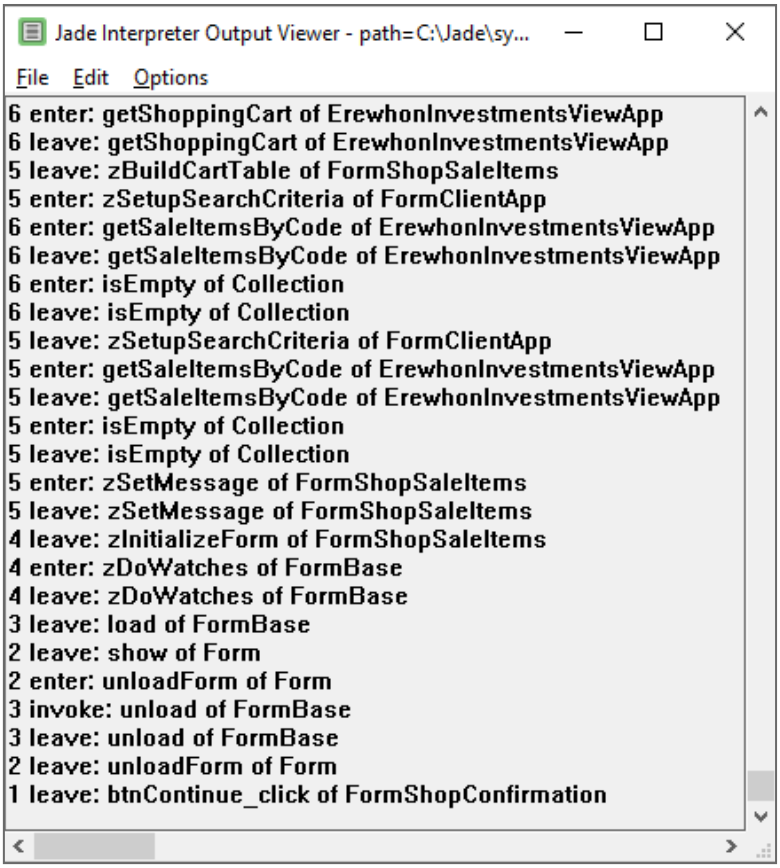
When you set the tracing of methods in an application, the Jade Interpreter Output Viewer is displayed when that application is running, recording:

- Millisecond timestamp
- Method invocation level (that is, method **1** calls method **2**, and so on)
- The methods and their class or classes that are entered, invoked, or left during the running of the selected application

Notes If your application code contains the **write** instruction, the result of tracing your methods may be intermingled with the **write** instruction results.

Although server methods are prevented from executing code that requires GUI objects (for example, forms and controls), the Jade Interpreter Output Viewer can be used in your JADE applications running on a server. (The Jade Interpreter Output Viewer is not implemented by the JADE GUI engine, as it is a separate module that talks directly to the operating system and does not require any GUI objects.)

The following image shows an example of traced methods output to the Jade Interpreter Output Viewer.



You can use the **WriteEnabled** parameter in the [JadeInterpreter] section of the JADE initialization file to disable output to the Jade Interpreter Output Viewer. For details, see "Interpreter Section [JadeInterpreter]", in the *JADE Initialization File Reference*.

Using the Jade Interpreter Output Viewer

The output from your traced methods (or **write** instructions) is displayed in the Jade Interpreter Output Viewer until you:

- Clear it manually
- Call the **Application** class **closeWriteWindow** method
- Close your current JADE session

The Jade Interpreter Output Viewer provides the:

- File Menu
- Edit Menu
- Options Menu

The text for each **write** instruction is accumulated and displayed in the Jade Interpreter Output Viewer. The text submitted by each **write** instruction is added to the end of any existing text on the form and the text box is scrolled to the end of the text to display the most-recent entries. When the accumulated text exceeds the maximum of 4M bytes of text, truncated to the nearest full line, the oldest write entries are dropped from the display until that size is no longer exceeded.

Tip If more than 4M bytes of text, truncated to the nearest full line, has been written to the viewer, you can display these by using a text editor if you have captured your output to the log file.

When the Jade Interpreter Output Viewer display is static (that is, there no are more **write** instructions and traced methods occurring), you can perform any available action in the text that is displayed. However, if **write** instructions continue to be output while you are attempting to interact with the text, there will be issues, as follows.

- New **write** statements add text to the end of the display. The initial text (possibly even all text) in the display can be removed if the maximum text size is exceeded.
- Any current selection will be lost and the display is scrolled to the end of the text.

To allow you to access displayed output, you can pause the display update of new **write** statements. For details, see "Options Menu".

File Menu

To clear output from the Jade Interpreter Output Viewer, perform one of the following actions

- Select the **Clear display** command from the Jade Interpreter Output Viewer File menu.
- Call the **Application** class **clearWriteWindow** method.

All output displayed in the window is then cleared.

To save the contents of the Jade Interpreter Output Viewer

- Select the **Save As** command from the Jade Interpreter Output Viewer File menu.

The common Set Save As File Name dialog is then displayed, to enable you to specify the location and name of the log file to which the contents are saved.

To set the capture file

- Select the **Set capture file** command from the Jade Interpreter Output Viewer File menu.

The common Set Capture File Name dialog is then displayed, to enable you to specify the location and name of your output file. By default, the output is directed to the **jadeout.log** file in your JADE **log** directory. If the capture file name does not include an absolute path, the directory is in the **logs** path.

You can view the output file by using a text editor; such as Notepad. (The output file is cumulative; that is, records are appended to any existing records in the output file.)

Note When you change the name of the capture file when capture to a file is set, all future **write** instructions and traced methods are output to this file, including **write** instructions and traced methods in your current session.

To delete the capture file

- Select the **Delete capture file** command from the Jade Interpreter Output Viewer File menu.

The capture file (defaulting to **jadeout.log** in your JADE **log** directory) is then deleted.

To annotate output to the Jade Interpreter Output Viewer

1. Select the **Annotate output** command from the Jade Interpreter Output Viewer File menu. The User Input dialog is then displayed.
2. In the **Enter text to add to output** text box, specify the text that you want to annotate to the Jade Interpreter Output Viewer.
3. Click the **OK** button. Alternatively, click the **Cancel** button to abandon your text.

The text that you specify is then displayed in the Jade Interpreter Output Viewer window (and written to the log file, if you have set capture to file). For example, you could add the following comment.

```
21 April 2016 - Start a new test now.
```

To print the output of the Jade Interpreter Output Viewer

1. Select the **Print** command from the Jade Interpreter Output Viewer File menu. The common Print dialog is then displayed, to enable you to specify your print options; for example, the printer name and the number of copies.
2. If text is selected in the Jade Interpreter Output Viewer, the **Selection** option button on the Print dialog is enabled.

   You can select the **All** or the **Selection** option button. The **All** option results in the entire current contents of the Jade Interpreter Output Viewer being printed. The **Selection** option results in only the selected text being printed.
3. Make the selections that you require.
4. Click the **OK** button. Alternatively, click the **Cancel** button to abandon your selections.

Any changes that you make to the printer options are retained for subsequent reuse, apart from the number of copies selected, which is reset to **1**.

Each page of the printed output has a heading of **Jade Interpreter Output Viewer – path=*database-path*** and a footer of **Page: *n* of *m date time***.

Long lines are split into multiple lines so that the entire line of text is visible on the printed output.

To change your Jade Interpreter Output View printer settings

1. Select the **Print Setup** command from the Jade Interpreter Output Viewer File menu. The common Print Setup dialog is then displayed. By default, printing is performed on the default printer using the default paper size and orientation that is set up for your system. Use the print set-up to change your printer settings.

   The **Print Setup** command uses common dialogs that enable you to specify advanced print options and printer connection options. For details, see your *Microsoft Windows User's Guide*.
2. Make the selections that you require.
3. Click the **OK** button. Alternatively, click the **Cancel** button to abandon the setting up of your print output.

Note The print setup dialogs and the options that these dialogs provide are printer-dependent and cannot be controlled from within JADE.

To exit from the Jade Interpreter Output Viewer, perform one of the following actions

- Select the **Exit** command from the Jade Interpreter Output Viewer File menu.
- Call the **Application** class **closeWriteWindow** method.

The Jade Interpreter Output Viewer is then closed, and the focus returns to the current application.

Edit Menu

To copy code selected in the Jade Interpreter Output Viewer, perform one of the following actions

- Select the **Copy** command from the Edit menu
- Press Ctrl+C
- Right-click in the Jade Interpreter Output Viewer and select **Copy** from the popup menu that is then displayed

The selected text is then copied to the clipboard. (These actions are disabled if there is no selected text.)

To cut code selected in the Jade Interpreter Output Viewer, perform one of the following actions

- Select the **Cut** command from the Edit menu
- Press Ctrl+X
- Right-click in the Jade Interpreter Output Viewer and select **Cut** from the popup menu that is then displayed

The selected text is then removed from the display and placed in the Windows clipboard. (The menu command is disabled if there is no selected text.)

To delete code selected in the Jade Interpreter Output Viewer, perform one of the following actions

- Select the **Delete** command from the Edit menu
- Press Delete

  Note The Delete key functions only when it is pressed on selected text.

- Right-click in the Jade Interpreter Output Viewer and select **Delete** from the popup menu that is then displayed

The selected text is then removed from the display but it does not get placed in the Windows clipboard. (The menu command is disabled if there is no selected text.)

To paste text from the clipboard, perform one of the following actions

- Select the **Paste** command from the Edit menu
- Press Ctrl+V
- Right-click in the Jade Interpreter Output Viewer and select **Paste** from the popup menu that is then displayed

The text is then pasted from the Windows clipboard, starting at the location of the caret. If text is selected in the Jade Interpreter Output Viewer, the contents of the Windows clipboard replaces the selected text. (The menu command is disabled if there is no text in the Windows clipboard.)

To select all text in the Jade Interpreter Output Viewer, perform one of the following actions

- Select the **Select All** command from the Edit menu
- Press Ctrl+A
- Right-click in the Jade Interpreter Output Viewer and select **Select All** from the popup menu that is then displayed

The entire contents of the Jade Interpreter Output Viewer text box is then selected. (The menu command is disabled if there is no text output to the Jade Interpreter Output Viewer.)

To undo the last paste, cut, or delete action, perform one of the following actions

- Select the **Undo** command from the Edit menu
- Press Ctrl+Z

Tip A stack of all paste, cut, and delete actions is maintained, which enables you to repeat undo actions.

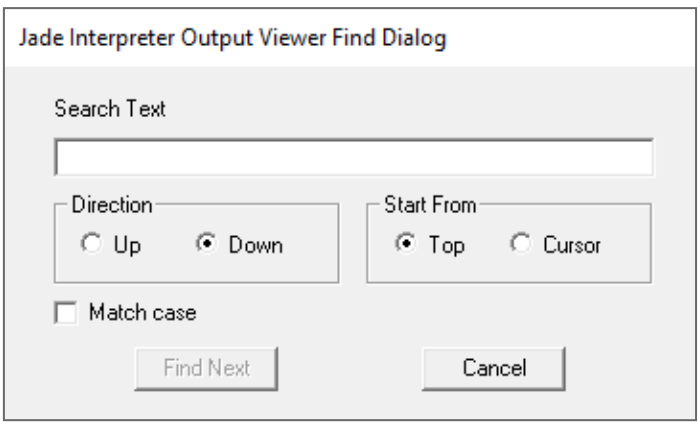
The **Undo** command is disabled if no undo actions are available.

If the target location of an undo action in the displayed text is removed (when the displayed text exceeds the maximum of 4M bytes of text, truncated to the nearest full line), the undo stack entry is also removed. That may mean that undo stack entries added after the removed action will still be available if the target text area is still present in the displayed text.

To find text in the Jade Interpreter Output Viewer

1. Select the **Find** command from the Edit menu or press Ctrl+F.

   The Jade Interpreter Output Viewer Find dialog, shown in the following image, is then displayed.



   If the Jade Interpreter Output Viewer has selected text, this text is displayed in the **Search Text** text box; otherwise the text box displays the text of the last find action that was performed or else the text box is empty.

2. In the **Search Text** text box, specify or change the text for which you want to search, if required.
3. If you want to search up the text box from the current caret position, select the **Up** option button in the

Direction group box. By default, the search is performed down the text box from the current position.

On subsequent searches, the value selected in the Direction group box is restored.

4. If you want the search to start from the current caret position, select the **Cursor** option button in the Start From group box. The search is then started at the caret position and works forwards or backwards, depending on the value of the Direction group box. By default, the search is performed from the top of the Jade Interpreter Output Viewer text box.

   When you set the Start From group box option to **Down**, the find action starts at the beginning of the text. If you set the direction to **Up**, the caption of the **Up** option button changes to **Bottom** and the search starts at the end of the text and searches backwards.

5. If you want to perform a case-sensitive search, check the **Match case** check box. By default, searches are case-insensitive.

   On subsequent searches, the value of the **Match case** check box is restored.

6. To abandon your selections, click the **Cancel** button.

7. To start your search of the Jade Interpreter Output Viewer, click the **Find Next** button.

The dialog remains open, regardless of whether the search text is found or not.

If a matching text occurrence is found, the found text will be highlighted and the text is scrolled into view, and the value of the option in the Start From group box is set to **Cursor**. If no matching text is found, a message box is displayed, stating that the find action failed to locate the specified text. (The displayed message includes the search text, the search case-sensitivity, and the search direction.)

You can place focus in the Jade Interpreter Output Viewer while the Find dialog is displayed, and any available edit action can be performed except for finding selected text; that is, the **Find Selected Text** command is disabled.

» To find the next occurrence of search text, perform one of the following actions

- Select the **Find Next** command from the Edit menu
- Press F3

The Jade Interpreter Output Viewer is then searched for the next occurrence of the text, in the currently set direction using the currently set case-sensitivity, without displaying the Jade Interpreter Output Viewer Find dialog.

If a matching text occurrence is found, that text is highlighted and the text is scrolled into view. If no matching text is found, a message box is displayed, stating that the find action failed to find any text. (The displayed message includes the search text, the search case-sensitivity, and the search direction.)

The **Find Next** command is disabled if no search text is set.

Although you can select the **Find Next** command when the Jade Interpreter Output Viewer Find dialog is open, the F3 shortcut key is available only when the Jade Interpreter Output Viewer has focus.

» To find the previous occurrence of search text, perform one of the following actions

- Select the **Find Prior** command from the Edit menu
- Press F3+Shift

The Jade Interpreter Output Viewer is then searched for the next occurrence of the text, in the opposite of the currently set direction using the currently set case-sensitivity.

If a matching text occurrence is found, that text is highlighted and the text is scrolled into view. If no matching text is found, a message box is displayed, stating that the find action failed to find any text. (The displayed message includes the search text, the search case-sensitivity, and the search direction.)

The **Find Prior** command is disabled if no search text is set.

Although you can select the **Find Prior** command when the Jade Interpreter Output Viewer Find dialog is open, the F3+Shift shortcut key combination is available only when the Jade Interpreter Output Viewer has focus.

To find selected text, perform one of the following actions

- Select the **Find Selected Text** command from the Edit menu
- Press Ctrl+F3

The Jade Interpreter Output Viewer is then searched for the next occurrence of the selected text, in the currently set direction using the currently set case-sensitivity.

Note This search action is the same as invoking the Jade Interpreter Output Viewer Find dialog and clicking the **Find Next** button without changing any search options.

If a matching text occurrence is found, that text is highlighted and the text is scrolled into view. If no matching text is found, a message box is displayed, stating that the find action failed to find any text. (The displayed message includes the search text, the search case-sensitivity, and the search direction.)

The **Find Selected Text** command is disabled if no search text is set.

Although you can select the **Find Next** command when the Jade Interpreter Output Viewer Find dialog is open, the F3 shortcut key is available only when the Jade Interpreter Output Viewer has focus.

Options Menu

To set the Jade Interpreter Output Viewer display on top of the current window

- Select the **Always on top** command from the Jade Interpreter Output Viewer Options menu.

A check mark is then displayed to the left of the command in the Options menu.

The Jade Interpreter Output Viewer is then always displayed on top of the current window unless it is minimized. To change the option back again, repeat the operation. (The check mark is then no longer displayed and the Jade Interpreter Output Viewer is no longer brought to the top.)

The **Always on top** command sets and unsets the [JadeInterpreterOutputViewer] section **WindowAlwaysOnTop** parameter in the JADE initialization file.

To capture your output to a file

- Select the **Capture to file** command from the Jade Interpreter Output Viewer Options menu.

A check mark is then displayed to the left of the command in the Options menu.

The output from traced methods and **write** instructions is written to the Jade Interpreter Output Viewer window by default. You can capture the output to a file as well as have it displayed in the Jade Interpreter Output Viewer. If you have specified that the output is also written to a file, the output is directed to the **jadeout.log** file in your JADE **log** directory, by default.

To change the option back again, repeat the operation. (The check mark is then no longer displayed.)

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

To change the screen font displayed in the Jade Interpreter Output Viewer

1. To display text in another font, select the **Font** command in the Options menu.

   The common Font dialog is then displayed, listing all screen fonts available for selection.

2. In the **Font Style** and **Size** combo boxes, select the font attributes that you require; for example, **Bold** and **12**.

3. Click the **OK** button. Alternatively, click the **Cancel** button to abandon your selections.

To set the font displayed in the Jade Interpreter Output Viewer to a fixed-pitch font

1. To display text in a fixed-pitch (monospaced) font, select the **Font (Fixed Pitch)** command in the Options menu. The common Font dialog is then displayed, listing all fixed-pitch fonts available for selection; for example, **Courier**.

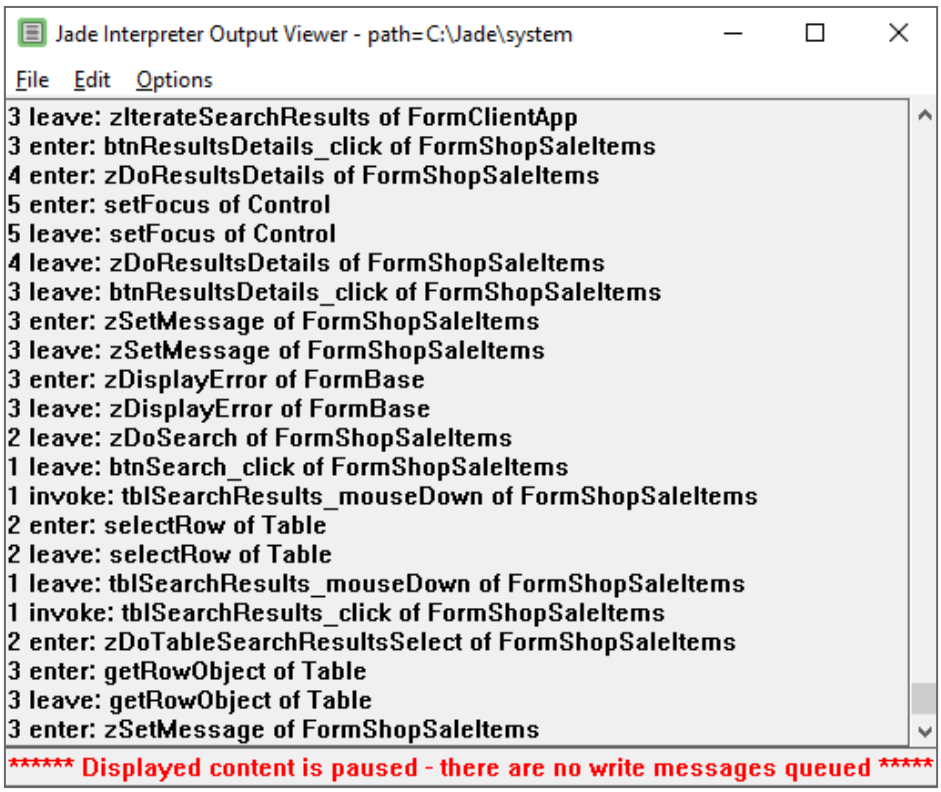
2. In the **Font Style** and **Size** combo boxes, select the font attributes that you require; for example, **Bold Italic** and **9**.

3. Click the **OK** button. Alternatively, click the **Cancel** button to abandon your selections.

To toggle the pausing of updates displayed in the Jade Interpreter Output Viewer

- To pause the output of text to the Jade Interpreter Output window, select the **Pause Display Update** command from the Options menu, or press the PAUSE BREAK key when the Jade Interpreter Output Viewer has focus.

When output to the Jade Interpreter Output View is paused, the Options menu command changes to **Resume Display Update** and a message is displayed in red in the status bar of the form, as shown in the following image.



Any write text that is subsequently output is queued and not added to the display until the Jade Interpreter Output Viewer until updating is resumed. The message in the status bar includes the number of queued write statements. When the display update is paused, you can interact with the displayed text without any interference.

Note Pausing the update does not apply to edit functionality (that is, cut, delete, paste, undo, and annotate actions).

The contents of any **write** statements that arrive while the display is paused continue to be written to the capture file, if you have selected that option.

When the amount of queued text exceeds the maximum size of text that the Jade Interpreter Output Viewer holds and displays (4M bytes of text, truncated to the nearest full line), the older entries are discarded. If messages have been discarded, the number of discarded message is displayed in the status bar at the bottom of the form. If file capture is on, all **write** statements continue to be output to that file.

- To resume the output of text to the Jade Interpreter Output Viewer, select the Options menu **Resume Display Update** command, or press the PAUSE BREAK key when the Jade Interpreter Output Viewer has focus.

  When the display update is resumed, the status bar at the bottom of the form is hidden, the Options menu command changes to **Pause Display Update**, and any queued write text is displayed.

To toggle how text output to the Jade Interpreter Output Viewer is displayed

- Select the **Word wrap** command from the Jade Interpreter Output Viewer Options menu.

By default, word wrapping is not set, so long lines of text are truncated on the right and the text box must be scrolled horizontally to view the extended text.

When you set word wrapping, a check mark symbol is displayed to the left of the **Word wrap** command in the Options menu, and word wrapping remains in effect until you select this command again. Any long lines of text are then wrapped so that additional lines are used to display the complete text and there is no horizontal scroll bar.

The **Word wrap** command sets and unsets the [JadeInterpreterOutputViewer] section **WordWrap** parameter in the JADE initialization file.

Terminating an Application

The **Terminate Application** command enables you to select the application that is to be terminated.

Note As an orderly termination is performed, you cannot terminate an application that is in exception state. You must first respond to the raised exception (that is, continue or abort) and then terminate the application by using this command.

To terminate an application from the Jade User Interrupt menu

1. Select the application that is to be terminated.

   The Jade User Interrupt submenu is then displayed.

2. Select the **Terminate Application** command from the submenu.

The selected application is then terminated. For more details about terminating a JADE application, see "Closing a JADE Application".

Showing an Invisible Form

When running multiple desktops on a workstation, switching to another desktop hides all forms on the prior desktop, which then generates a warning message box advising you that the GUI application (that is, **ApplicationType_GUI**) has no visible forms.

As GUI applications can run without a visible form, the **Show an invisible form** command in the Jade User Interrupt submenu enables you to terminate the application by using the system tray icon.

By default, the **Show an invisible form** command is disabled in the Jade Interrupt menu.

To display the first invisible form for a GUI application when running multiple desktops

1. Click on the system tray icon to display the application name followed by **- no visible forms** (for example, **LockTest (4) – no visible forms**).

   The Jade User Interrupt submenu is then displayed, with the **Show an invisible form** command enabled.

2. You can then perform one of the following actions.

   - Select the **Terminate Application** command, to terminate the application. (For more details, see "Terminating an Application".)
   - Select the **Show an invisible form** command, which displays the first invisible form for the application.

If the **ShowUserInterrupt** parameter in the [Jade] section of the JADE initialization file is set to **false**, this command is ignored and an icon is displayed in the system tray if the application does not have a visible form.

Closing a JADE Application

Use the application-specific means of exiting from your JADE runtime application (for example, your system may have an Exit menu, an **Exit** button, or a File menu that contains an **Exit** command).

When you exit from a runtime JADE application:

- Any current form is closed.
- All locks are released.
- All objects (including profile statistics) are cleared from memory.

Note Any current JADE development session is not terminated when you exit from a runtime JADE application.

To sign off from a JADE application, perform one of the following actions

- Select the application-specific means of exiting from your JADE application (for example, an **Exit** command from a File menu).
- Click the close icon at the top right corner of the window, or select the **Close** command from the Control-Menu.
- Press Ctrl+X.

Your JADE application is then closed down.

Shutting Down a JADE Session

JADE rejects Windows log-off or shut-down requests if a modal form or a Windows dialog (for example, a message box, a common dialog, and so on) is currently active, because in those cases, the calling method has been suspended until the dialog is closed.

The user must complete the required actions on that dialog before the suspended method logic can continue. Graceful shut down of the application can be accomplished only if the application is in a truly idle state.

If the JADE application does not have any suspended modal or dialog methods in progress, requesting a Windows log off or shut down is not rejected by the JADE application. JADE calls the **queryUnload** and **unload** methods on each form (in the correct order), and terminates gracefully.

If the JADE application has suspended modal or dialog methods in progress, requesting a Windows log off or shut down is rejected and the log off or shut down action will not occur. The user must complete or cancel the current modal dialog or dialogs and then request the log off or shut down action again.

The best policy for a user is to shut down all applications before requesting that Windows logs off or shuts down.

Chapter 2 Using Skins in Runtime Applications

This chapter covers the following topics.

Overview

Note The functionality documented in this section and its subsections was superseded from JADE release 6.0, and is retained for backward compatibility. For details about the enhanced skin functionality that enables you to define a category both for a skin and for a window (at the application, form, or control level), see "Maintaining Skins Using Extended Functionality", later in this document.

Your JADE systems can contain JADE skins, which are a series of images that are applied to the caption line, menu line, and border areas of each form to provide an enhanced look and feel to each form. The skin can also define images for button, check box, and option button controls to further enhance the appearance of forms.

By default, skins are not used in runtime applications but the developer of the application can specify and select the skin that is applied to all JADE forms displayed during the running of that application in the current work session.

You can use JADE skins in applications to enforce a specific skin (for example, a company logo, and so on) or the initialization of the runtime application can provide users with the ability to select a preferred skin that is set during the application's **initialize** method (by calling **app.setSkin**).

The global collection of skins is available to all runtime applications in all systems in your JADE database. In addition, you can enable runtime users to define or maintain JADE skins that you have provided or you can enable them to select from the global collection of skins. For details, see "Defining and Maintaining JADE Skins" or "Selecting a Skin to Use at Run Time", respectively.

Note As references to skins information is contained in the **_usergui.dat** system schema file, when the **ReadOnlySchema** parameter in the [JadeClient] or [JadeServer] section of the JADE initialization file is set to **true**, skins cannot be loaded.

For details about extracting JADE skins for runtime applications from your JADE development environment, see "Specifying Your Schema Options" under "Extracting Your Schema", in Chapter 10 of the *JADE Development Environment User's Guide*.

Defining and Maintaining JADE Skins

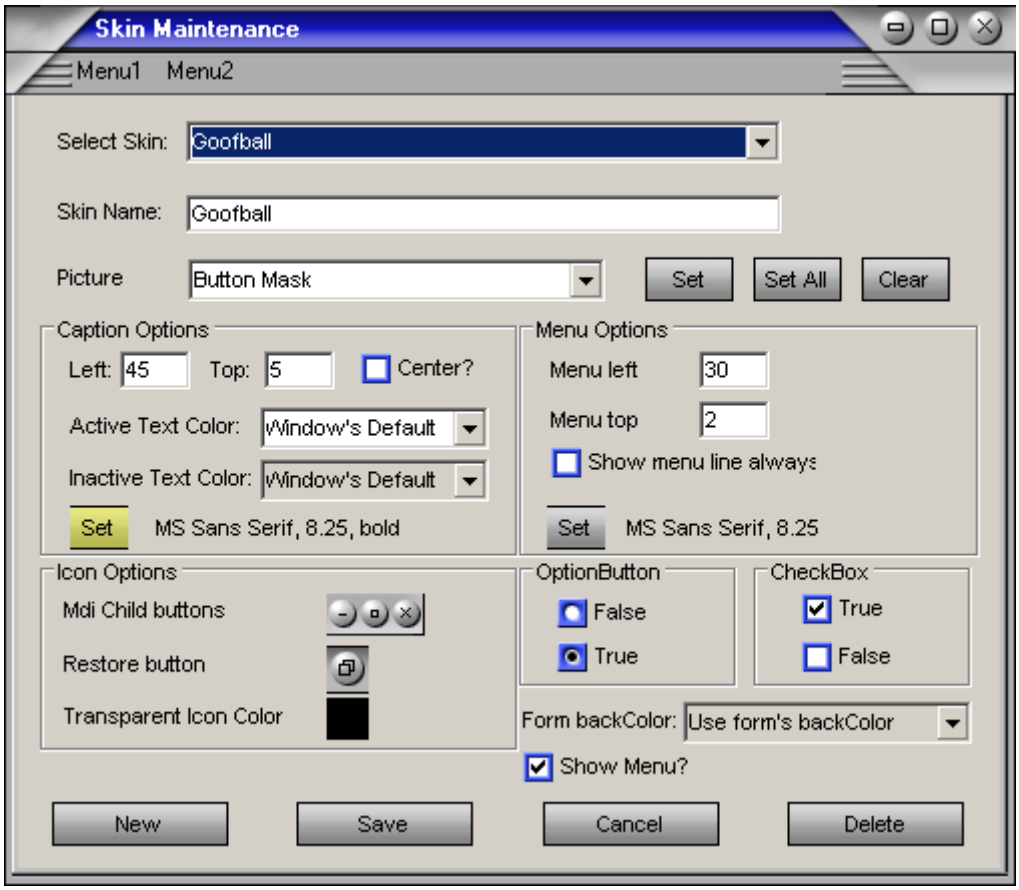
You can create your own skin images for definition and maintenance by users of the runtime applications, if required. For details, "JadeSkin Class", in Chapter 1 of the *JADE Encyclopaedia of Classes*.

Users can define or maintain skins based on the picture files that you provide for each required image only when logic in your runtime application invokes the **JadeSkinMaint** form provided by JADE, as shown in the following example.

```
maintainSkin_click(menuItem: MenuItem input) updating;
vars
    form : JadeSkinMaint;
begin
    create form;
    form.showModal;
epilog
    delete form;
end;
```

Note Before you can define a new skin for your applications, a picture file (for example, a .**gif**, .**png**, .**bmp**, or .**jpg**) must exist for each of the images that you want to specify.

The following image shows the Skin Maintenance form (with a skin applied) that is displayed when you invoke the **JadeSkinMaint** form in your runtime applications, to enable a runtime user to define or maintain a skin that is available for selection in the application (based on the picture files that you have provided for each of the images).



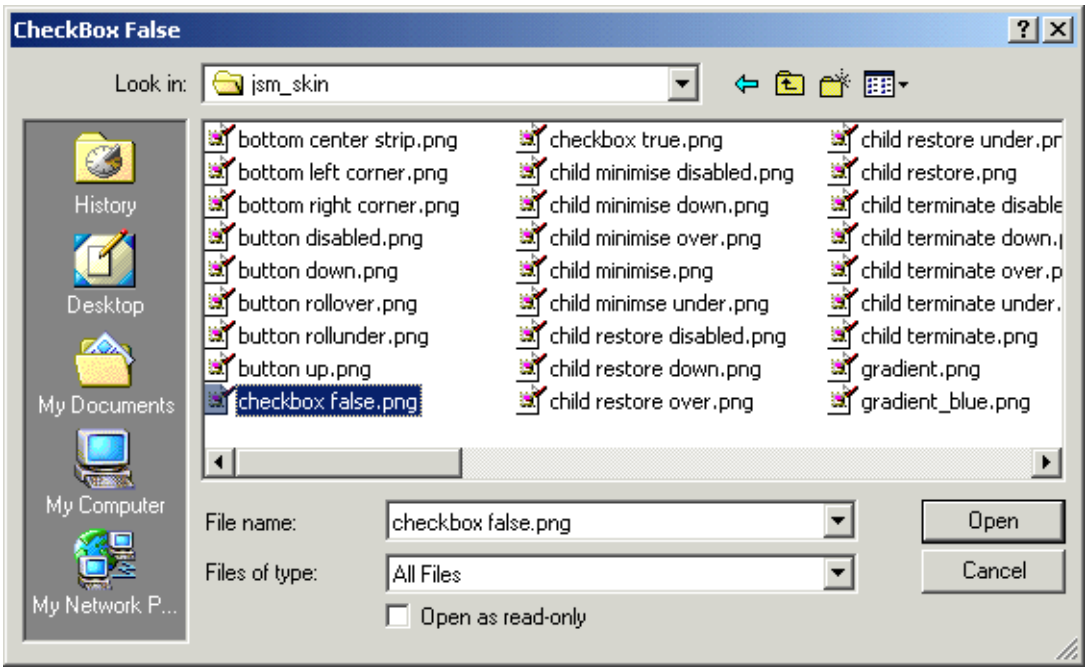
Defining a New Skin Based on Supplied Image Picture Files

To define a new skin

1. Click the **New** button and then enter the name of the skin in the **Skin Name** text box. This name is a description of the skin only, and no check is performed to ensure that the name is unique.

2. To request the loading of all the pictures that describe the skin, click the **Set All** button.

   A series of common File dialogs like that shown in the following image is then displayed, to enable you to locate each of the pictures required in turn.



   The title of the common dialog displays the required image. If you do not have a file for a specific image or you do not want to specify a file for that image, simply click the **Cancel** button to enable you to select the picture file for the next image. If you cancel a common dialog display for any picture, the default image is used for that property.

   When you have specified (or cancelled) the appropriate picture file for each image that you want to define, the common dialog is then closed and focus returns to the Skin Maintenance dialog.

3. Use the text box or check box controls in the Caption Options group box to position the form captions to meet your requirements.

   If you want to center the caption, click the **Center?** check box. Any value specified in the **Left** text box is ignored unless the caption is too large for the available area.

   The **Left** and **Top** text boxes indicate the number of pixels to offset the caption start position (for example, 49 and 6, respectively).

4. Use the **Active Text Color** and **Inactive Text Color** drop-down list boxes in the Caption Options group box to select a color for text on active and inactive captions on your skinned forms if you do not want your Windows default values applied (specified on the **Appearance** sheet of the Display dialog, accessed from the **Display** icon in the Control Panel for your workstation).

When you select the **Select Color** option in the list box, the common Color dialog is then displayed, to enable you to select one of the basic colors or to define a custom color to meet your requirements.

5. Use the text box or check box controls in the Menu Options group box to position menus on your forms to meet your requirements. The **Menu left** and **Menu top** text boxes indicate the number of pixels to offset the menu start position (for example, 43 and 4, respectively).

6. When the **Show Menu?** check box is checked (that is, the value is set to **true**), the caption line and the menu line are always displayed on each form regardless of whether the form has a menu. If this check box is unchecked (that is, it is set to **false**), the menu line is displayed only on forms that have a menu.

   Tip Use the **Show Menu?** check box at the lower left of the dialog to test the skin without a menu line.

7. Use the **Set** button in the Caption Options or Menu Options group box to change the font if you do not want Tahoma, regular, 8.25 points to be used for your captions or menus. The common Font dialog is then displayed, to enable you to select the required font and its attributes. User font preferences are not used when displaying the caption and menus, as they may be inappropriate for the skin.

8. If the form icons have been created with a transparent color surround, click the box at the right of the **Transparent Icon Color** label to set the transparent color. The common Color dialog is then displayed, to enable you to select or define the transparent color that you require. The OptionButton and CheckBox group boxes provide an example of the appearance of option buttons and check boxes with the skin applied.

9. If you want to set the background color for forms, select the **Selected Color** item in the **Form backColor** combo box. The common Color dialog is then displayed, to enable you to select the background color that you require.

   Note This background color applies only to forms that use the **Windows** class constant **Color_3DFace** default form background setting.

10. When you have finished specifying the settings that you require for the skin, click the **Save** button to store the skin definition in the corresponding properties of the **JadeSkin** class.

For details about maintaining skins or selecting the skin for use in the runtime application, see "Maintaining an Existing Skin" or "Selecting a Skin to Use at Run Time", respectively.

Maintaining an Existing Skin

To change an existing skin

1. Select the skin whose settings you want to change from the **Select Skin** combo box. You can then change any setting for the selected skin. For details, see "Defining a New Skin Based on Supplied Image Picture Files".

2. To set a specific skin image, select the image that you want to define or change from the list displayed in the **Picture** combo box and then click the **Set** button. The common File dialog is then displayed, to enable you to locate the picture file that you want to define. Alternatively, you can clear a picture image by selecting the image name in the **Picture** combo box and then clicking the **Clear** button.

3. When you have modified all settings to meet your requirements, click the **Save** button. The definition of the skin, stored in the **JadeSkin** class, is then updated with your changes. Alternatively, click the:

   - **New** button, to clear the Skin Maintenance dialog so that you can define a new skin.
   - **Cancel** button, to discard your changes and close the Skin Maintenance dialog.

   If you have changed the current skin and not yet saved your changes, you are prompted to confirm whether you want to save or discard your changes before the dialog is cleared or closed.

Note Any changes made to a skin do not affect any current users of that skin. For details about selecting the skin for use in the runtime application, see "Selecting a Skin to Use at Run Time".

Selecting a Skin to Use at Run Time

The Skin Selection dialog enables users of runtime applications to select the skin that they want to use during the current runtime work session. For details about enabling runtime users to define or maintain JADE skins that you have provided, see "Defining and Maintaining JADE Skins". The following example shows the creation and use of the **JadeSkinSelect** form from a **click** event method in user application logic.

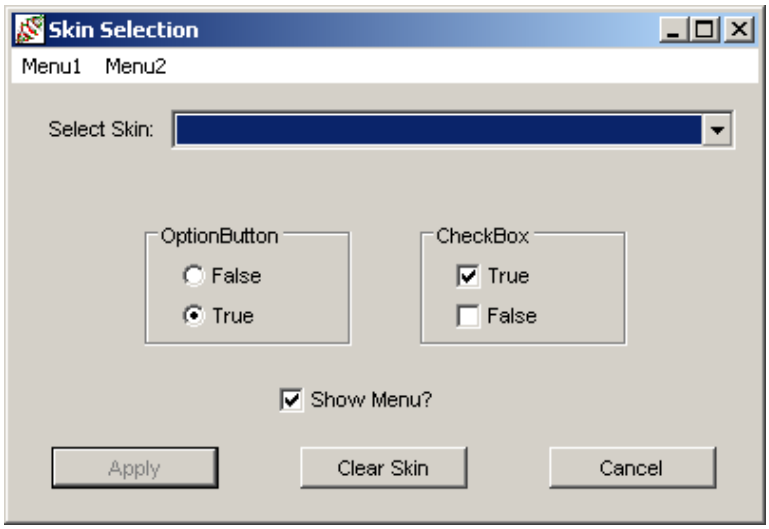
```
selectSkin_click(menuItem: MenuItem input) updating;
vars
    form : JadeSkinSelect;
begin
    create form;
    form.showModal;
epilog
    delete form;
end;
```

When using the Skin Selection dialog:

- The form is displayed using the requested skin.
- The option buttons, check boxes, and buttons display the skin presentations.
- The **Show Menu?** check box enables users to view the effects of a skin with and without a menu.
- The **Apply** button causes that skin to be applied to the forms of the current application.
- The **Clear Skin** button removes the use of a skin from the current application.

  The **Cancel** button leaves the skin usage unchanged.
- Programmatically, if the user:
  - Selects a skin, **modalResult = JadeSkinSelect.Skin_Chosen(1)** and the **userObject** property contains the selected skin.
  - Clicks the **Cancel** button, **modalResult = JadeSkinSelect.Skin_Cancel(0)**.
  - Clicks the **Clear Skin** button, **modalResult = JadeSkinSelect.Skin_Cleared(2)**.

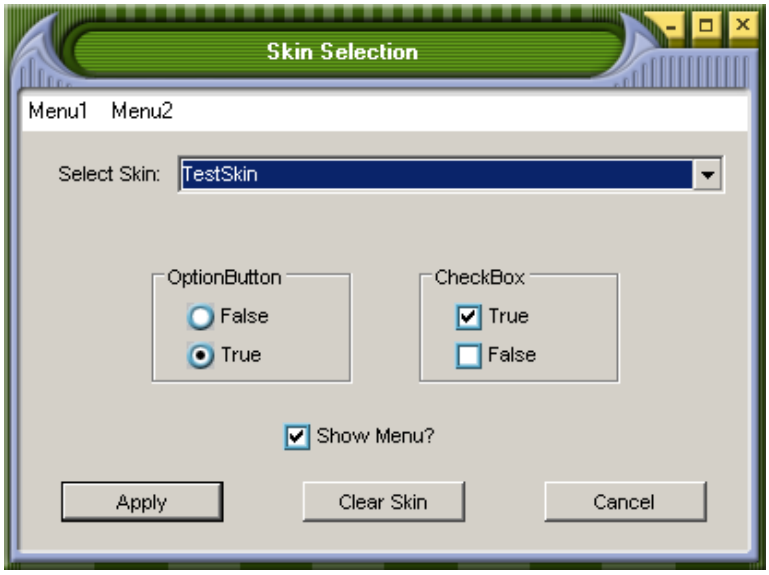
Use the **app.getSkinCollection** method to obtain the collection of skins and display a list of the names of each skin object if you want to implement your own selection scheme in your applications. Your user application needs only to create and modally show this form, shown in the following image.



To select a skin for use in the current work session

1. Select the skin from the list that is displayed in the **Select Skin** combo box. The option buttons, check boxes, and buttons then display the presentation offered by the selected skin.
2. Click the **Show Menu?** check box to view the effects of a skin without a menu. By default, a check mark symbol (✓) is displayed in this control, indicating that the menu line is displayed on the Skin Selection dialog.
3. To apply the selected skin to all forms in the current application during the current work session, click the **Apply** button. Alternatively, perform one of the following actions.
   - Click the **Clear Skin** button to remove the use of a skin from the current application.
   - Click the **Cancel** button to abandon your selections and leave the skin usage unchanged.

The following image shows an example of the Skin Selection dialog using a skin.



The JADE executable calls the **DisableProcessWindowsGhosting()** Microsoft API on initiation, which disables Windows' ghosting so that a non-responsive form does not show *Not Responding* and have the ghosting effect applied by Windows.

The form will still not automatically paint itself when the presentation thread is busy processing JADE logic. Windows automatically redraws that part of the form or forms that need refreshing from a saved copy of the previously painted image or images.

You can call the following methods while performing a long processing loop to address the repainting issue.

- **Application**::**paintIfRequired**;

  The **Application** class **paintIfRequired** method causes all forms of the application to be repainted if a paint is required. A **refreshNow** event is performed on that part of the form that needed refreshing. If **paint** events are not required, no action is performed. This method performs any repainting required without having to perform an **app.doWindowEvents** method call, and therefore does not allow the user interface to be active.

  Other than any **paint** events, no other events, notifications, or timer events will be processed as a result of this **paintIfRequired** method call.

- **Form**::**paintIfRequired**;

  The **Form** class **paintIfRequired** method performs the same action as the **Application** class **paintIfRequired** method except that only the form on which the call is made will be affected.

Notes After a repaint, any clicked button that initiated the processing loop will be drawn in the up position, so it will be important that the user is given a visual indication that the processing is still progressing by some other means; for example, using the **app.mousePointer := 11** (busy) property value.

You will need to add a call to your logic loop that is regularly performed; for example, call it when the **Cancel** button is checked for a **click** event, when a progress bar update ticks over a percentage, or at a specified number of seconds, as shown in the following code fragment.

```
cancelled := false;
    while not cancelled do
        .... logic
    // the click event sets the cancelled property
    btnCancel.doWindowEvents(0);
    app.paintIfRequired();
endwhile;
```

Maintaining Skins Using Extended Functionality

By default, skins are not used in runtime applications. You can specify and select the skin that is applied to all JADE applications, forms, or controls during the running of an application in the current work session.

You can use JADE skins in applications to enforce a specific skin (for example, a company logo) or the initialization of the runtime application can provide users with the ability to select a preferred skin that is set during the application's **initialize** method (by calling **app.setApplicationSkin**).

The global collection of skins is available to all runtime applications in all systems in your JADE database. In addition, you can enable runtime users to define or maintain JADE skins that you have provided or you can enable them to select from the global collection of skins. For details, see "Selecting a Skin to Use in Runtime Applications".

Note As references to skins information is contained in the **_usergui.dat** system schema file, when the **ReadOnlySchema** parameter in the [JadeClient] or [JadeServer] section of the JADE initialization file is set to **true**, skins cannot be loaded into a production database.

For details about extracting JADE skins for runtime applications from your JADE development environment, see "Extracting and Loading Skins", in Chapter 9 of the *JADE Developer's Reference* and to "Specifying Your Schema Options" under "Extracting Your Schema", in Chapter 10 of the *JADE Development Environment User's Guide*.

You can create your own skin images so that users of the runtime application can maintain the skin definition, if required. For details, "JADE Development Environment Skin Classes", in Chapter 9 of the *JADE Developer's Reference*.

Defining and Maintaining JADE Skins at Run Time

Users can define or maintain skins based on the picture files that you provide for each required image only when logic in your runtime application invokes the **JadeSkinMaintenance** form provided by JADE, as shown in the following example of a **click** event in user application logic.

```
maintainSkin_click(menuItem: MenuItem input) updating;
vars
    form : JadeSkinMaintenance;
begin
    create form;
    form.showModal;
epilog
```

```
        delete form;
    end;
```

Notes Before you can define a new skin for your applications, a picture file (for example, a .**gif**, .**png**, .**bmp**, or .**jpg**) must exist for each of the images that you want to specify.

If you have changed the current skin and not yet saved your changes, you are prompted to confirm whether you want to save or discard your changes before the dialog is cleared or closed. Any changes made to a skin do not affect any current users of that skin. (See also "Selecting a Skin to Use in Runtime Applications".)

The Jade Skin Maintenance dialog maintains the definition of a skin. The dialog displays a folder with a sheet for all of the entities that can be defined and an additional sheet that shows the users of each skin entity. These sheets are as follows.

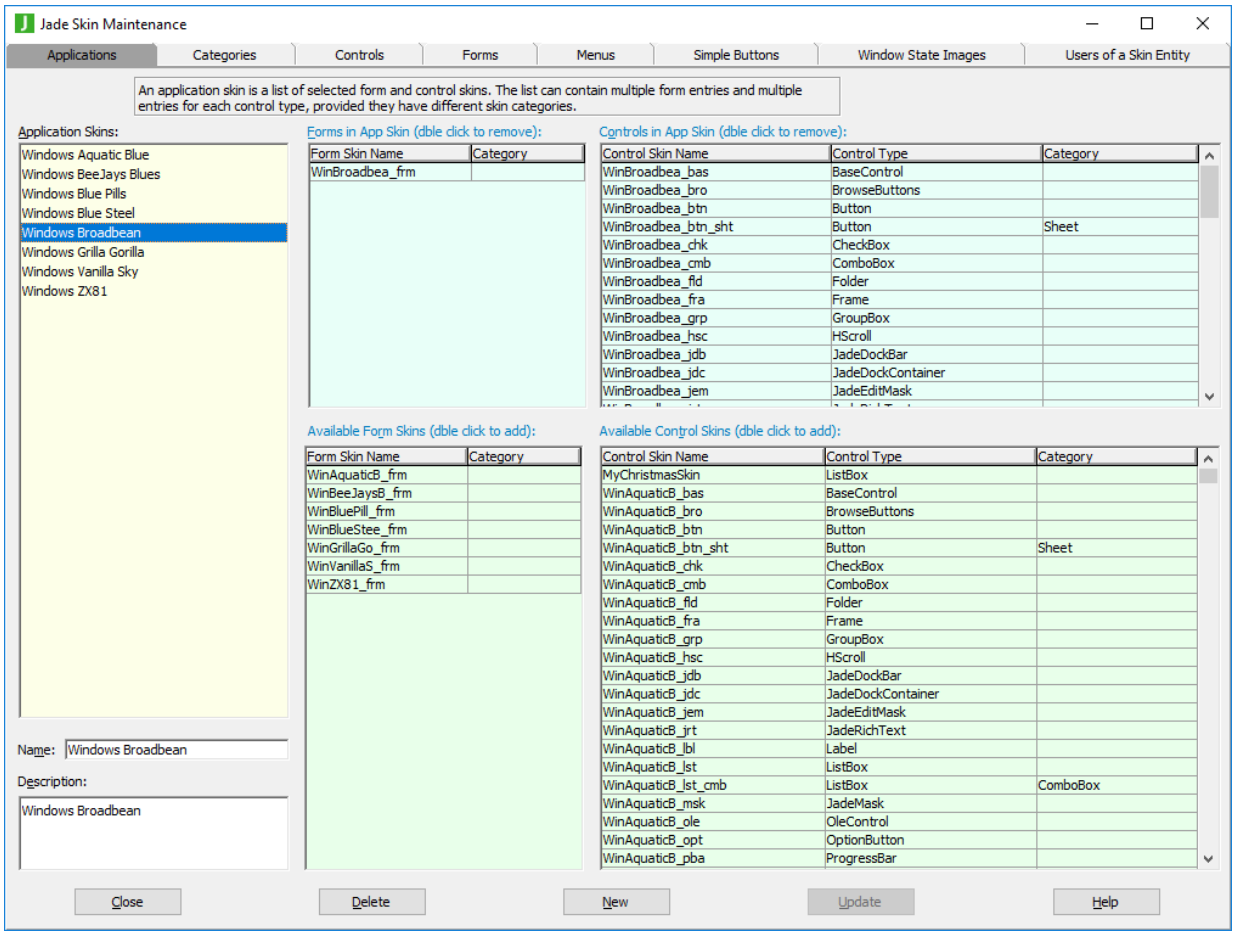
- Applications
- Categories
- Controls
- Forms
- Menus
- Simple Buttons
- Window State Images
- Users of a Skin Entity

To add or maintain a skin for the required entity

- Click on the appropriate tab to display that sheet. Alternatively, click the **Close** button or the Esc key to close the form.

Using the Applications Sheet

The **Applications** sheet, shown in the following image, is displayed by default when you access the Jade Skin Maintenance dialog.



The left of the **Applications** sheet displays the list of defined application skins. Application skins that are in different categories can contain one or more form entries and control entries.

To add or maintain an application skin

- To add a new skin to the application:

  a. Click the **New** button if another application skin is selected.

  b. Enter the name of the application skin in the **Name** text box or select the skin from the **Application Skins** list box.

  Note Application skin names must be unique.

  c. Enter a description of the skin in the **Description** text box, if required.

  d. Select the list of form skins that make up the application skin by double-clicking on each form skin name in the **Available Form Skins** table that you want to add to the application skin.

The selected form skin is then moved to the **Forms in App Skin** table. If the form skin exists in the same category, a message box prompts you to confirm that you want to replace the existing form skin with your selected skin.

Conversely, to remove a form skin from the application, double-click on each form skin in the **Forms in App Skin** table that you want to remove from the application. The form skin is then moved to the **Available Form Skins** table.

e. Select the list of control skins that make up the application skin by double-clicking on each control skin name in the **Available Control Skins** table that you want to add to the application skin. The selected control skin is then moved to the **Controls in App Skin** table. If the control skin exists in the same category, a message box prompts you to confirm that you want to replace the existing control skin with your selected skin.

Conversely, to remove a control skin from the application, double-click on each control skin in the **Controls in App Skin** table that you want to remove from the application. The control skin is then moved to the **Available Control Skins** table.

f. Click the **Update** button.

- To update an existing application skin:

a. In the **Application Skins** list box, select the application skin that you want to change. The name of the selected application skin is then displayed in the **Name** text box.

b. Edit the appropriate data to meet your requirements. (For details, see steps d and e in the previous action.)

c. Click the **Update** button.

- To delete an existing application skin:

a. In the **Application Skins** list box, select the application skin that you want to delete. The name of the selected application skin is then displayed in the **Name** text box.

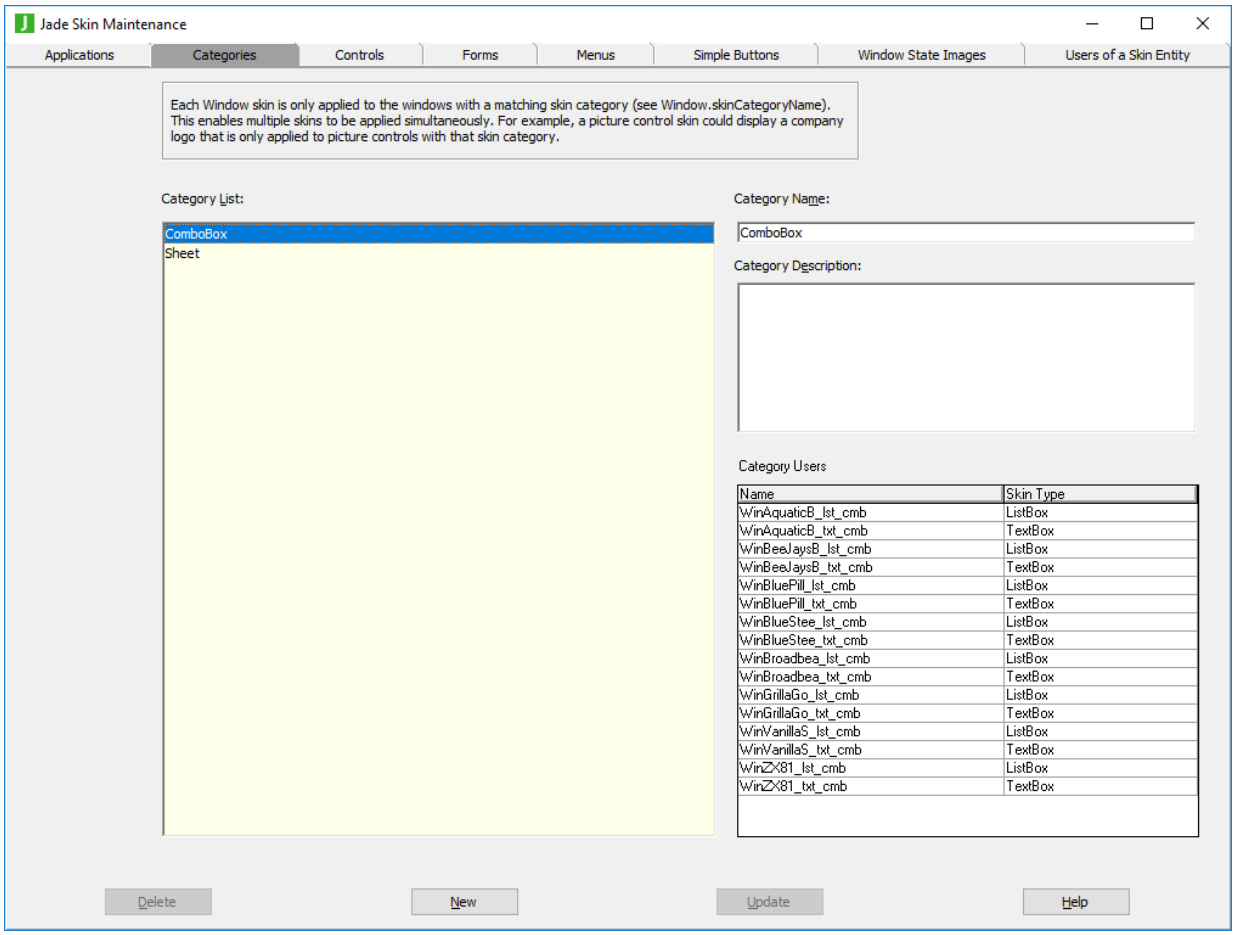
b. Click the **Delete** button.

c. A message box advises you that the skin may be referenced externally and prompts you to confirm that the skin can be safely deleted.

d. Click the **Yes** button to confirm the deletion. Alternatively, click the **No** button to abandon the deletion.

Using the Categories Sheet

The **Categories** sheet, shown in the following image, is displayed when you select the **Categories** tab in the Jade Skin Maintenance dialog.



The left of the **Categories** sheet displays the list of defined skin categories.

The lower right of the sheet displays a table of the skin entities that use the currently selected category.

As each skin is applied only to windows with a matching skin category, you can apply multiple skins to a window simultaneously. For example, you could use a picture control skin to display your company logo so that it is displayed only on picture controls that have the appropriate skin category.

» To add or maintain a skin category, perform one of the following actions

- To add a new skin category:

  a. Click the **New** button if another category is selected.

  b. Enter the name of the skin category in the **Category Name** text box or select the skin from the **Category List** list box.

  Note Category names must be unique.

c. Enter a description of the category in the **Category Description** text box, if required.

d. Click the **Update** button.

- To update an existing skin category:

  a. In the **Category List** list box, select the category that you want to change. The name of the selected category is then displayed in the **Category Name** text box.

  b. Edit the appropriate data to meet your requirements. (For details, see steps b and c in the previous action.)

  c. Click the **Update** button.

- To delete an existing skin category that is not currently referenced:

  a. In the **Category List** list box, select the skin category that you want to delete. The name of the selected category is then displayed in the **Category Name** text box.

  b. Click the **Delete** button.

  The **Delete** button is disabled if the category is referenced by any window skin entry.

Using the Controls Sheet

The **Controls** sheet is displayed when you select the **Controls** tab in the Jade Skin Maintenance dialog. The appearance of the **Controls** sheet and the edit actions that you can perform is determined by the current control in the **Control Type** combo box.

You can assign a control skin to a specific control or to an application skin. You can use skin categories to define different skins for your control subclasses. (For details, see "Using the Applications Sheet" and "Using the Categories Sheet".)

» To add a new skin for the current control

1. Click the **New** button if another control skin is selected in the **Control Type** combo box.
2. Enter the name of the control skin in the **Name** text box. (Control skin names must be unique.)
3. Enter a description of the skin in the **Description** text box, if required.
4. Specify the rest of the information for that skin. (For details, see the standard actions that you can perform for any type of control and the following subsections.)
5. Click the **Update** button.

» To update an existing skin for the current control

1. In the **Control type skin List** table, select the skin that you want to change. The name of the selected skin is then displayed in the **Name** text box.
2. Edit the appropriate data to meet your requirements. (For details, see the actions required to add or maintain standard functionality for skin controls and the following subsections.)
3. Click the **Update** button.

» To delete a control skin

1. In the **Control type skin List** table, select the skin that you want to delete. The name of the selected skin is then displayed in the **Name** text box.

2. Click the **Delete** button. The **Delete** button is disabled if the control skin is referenced by any other skin entity.

» To perform a standard edit action on the Controls sheet for any type of control

- In the **Skin Category** combo box, select the skin category that applies to the control skin or select the default **<none>** entry. A control skin is applied only to controls of the same type with the same defined skin category.
- In the **Apply when** combo box, specify when the **border** and **three-dimensional** styles of the target control affect whether the skin is applied to that control.

  Although the skin is always applied to the control type by default, it can be displayed only when the control has a border or when it is a three-dimensional control.
- In the **BorderStyle** combo box, specify the type of border that applies to the skin. Although border images are used by default, you can specify that no border is applied to the control skin or that a single border, three-dimensional sunken, or three-dimensional raised border is applied.

  If the border style is not set to the default **4 – use border images** value, the selected border is displayed and border images are ignored. If the border style is set to the default **4 – use border images** value, the border is drawn using the supplied images. If no images are supplied, the control has no border.
- If the border style is set to the default **4 – use border images** value in the **BorderStyle** combo box, you can set these images by clicking the **Set All** button that requests the loading of all pictures that describe the skin.

  A series of common File dialogs is then displayed, to enable you to locate each of the pictures required in turn. The title of the common dialog displays the required image. If you do not have a file for a specific image or you do not want to specify a file for that image, simply click the **Cancel** button to enable you to select the picture file for the next image. If you cancel a common dialog display for any picture, no image is assigned to that property.

  When you have specified (or cancelled) the appropriate picture file for each image that you want to define, the common dialog is then closed and focus returns to the **Controls** sheet of the Jade Skin Maintenance dialog. Alternatively, you can perform one of the following actions.

  - Set a specific control skin image by selecting the image type in the combo box (for example, **Right Strip**) and then clicking the **Set** button.

    The common File dialog is then displayed, to enable you to locate the picture file that you want to define or change.
  - Clear an existing image by selecting the image type in the combo box and then clicking the **Clear** button.
- If the border style is set to the **1 – single** value in the **BorderStyle** combo box, you can use the **Border color single** property to set the color of the border. Select a border color by clicking the **Set** button. The common Color dialog is then displayed, to enable you to select the border color that you require.
- Use the **The inner image is a brush?** check box to specify whether the inner image is treated as a brush to be repeatedly drawn over the whole area of the control or whether it is an image that is drawn centered in the inner area of the control.
- To select a background color for the control, check the **Default backColor** check box. The common Color dialog is then displayed, to enable you to select the background color that you require. The background color is used only when an inner image is not set or it is not a brush.

  If you want to restore the default background color of the control when you have selected a custom color, check the **Default backColor** check box so that a check mark symbol (✓) is displayed and the default color is restored.

- To select a foreground color for the control, check the **Default foreColor** check box. The common Color dialog is then displayed, to enable you to select the foreground color that you require.

  If you want to restore the default foreground color of the control when you have selected a custom color, check the **Default foreColor** check box so that a check mark symbol (✓) is displayed and the default color is restored.

- To select a foreground color for the control when it is disabled, check the **Default disabled foreColor** check box. The common Color dialog is then displayed, to enable you to select the foreground color that you require for the disabled control.

  If you want to restore the default disabled foreground color of the control when you have selected a custom color, check the **Default disabled foreColor** check box so that a check mark symbol (✓) is displayed and the default disabled foreground color is restored.

- To select a background color for when the control has focus, check the **Default focusBackColor** check box. The common Color dialog is then displayed, to enable you to select the background color that you require for the focused control.

  If you want to restore the default focus background color of the control when you have selected a custom color, check the **Default focusBackColor** check box so that a check mark symbol (✓) is displayed and the default focus background color is restored.

- To select a foreground color for when the control has focus, check the **Default focusForeColor** check box. The common Color dialog is then displayed, to enable you to select the foreground color that you require for the focused control.

  If you want to restore the default focus foreground color of the control when you have selected a custom color, check the **Default focusForeColor** check box so that a check mark symbol (✓) is displayed and the default focus foreground color is restored.

- In the **Image Mask** combo box, select a non-rectangular region mask image to be applied to the control skin. To remove an applied mask, select the **<None>** value (the default).

- If you do not want the default **Default_Color** to be used for your control skin, click the font **Set** button. The common Font dialog is then displayed, to enable you to select the required font and its attributes.

  Alternatively, click the font **Clear** button to restore the default font of the control.

Note An example of the appearance of the control skin is displayed in the **Example** group box.

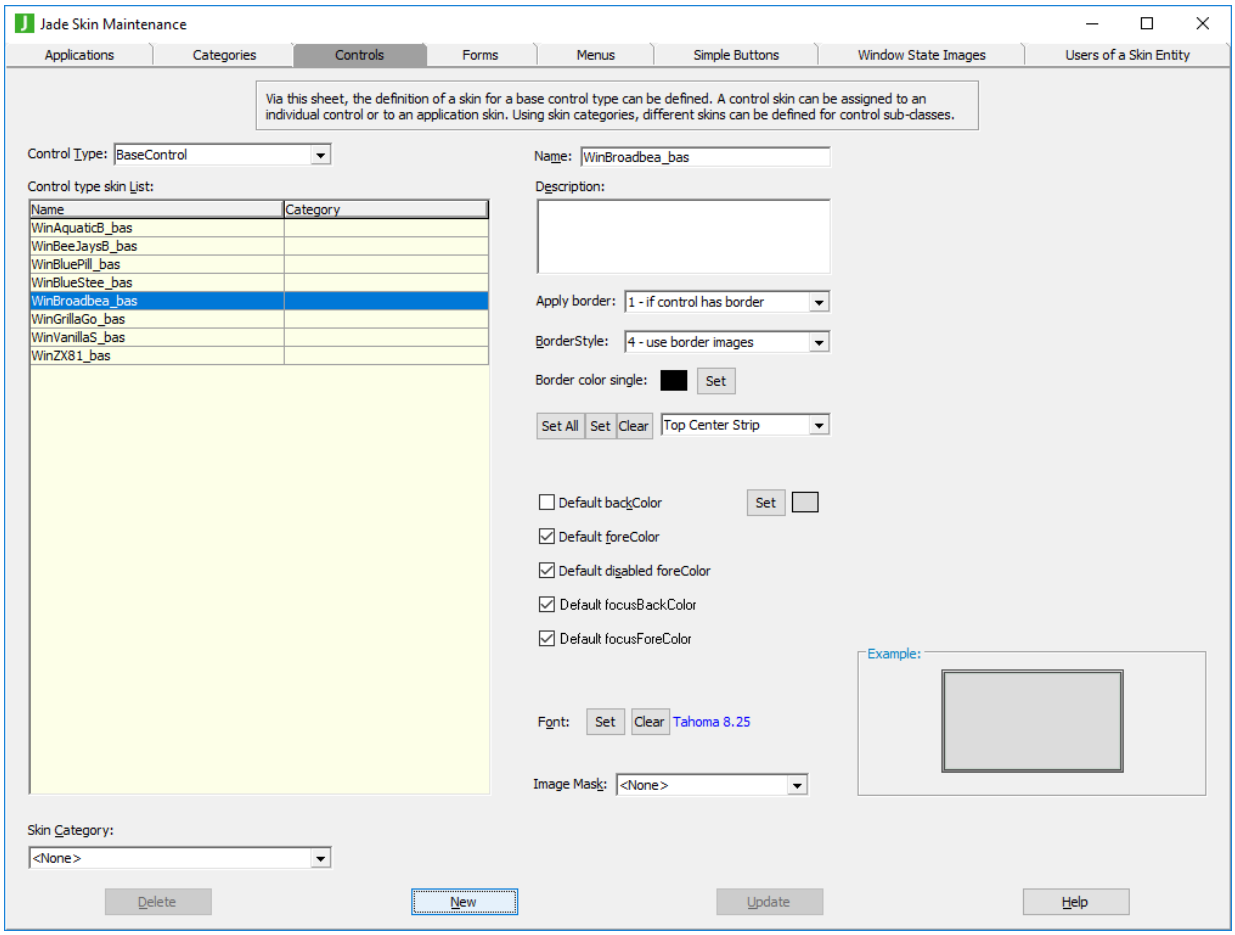
For details about defining or maintaining control skins, see the following subsections.

- Defining and Maintaining BaseControl Skins
- Defining and Maintaining BrowseButtons Control Skins
- Defining and Maintaining Button Control Skins
- Defining and Maintaining CheckBox Control Skins
- Defining and Maintaining ComboBox Control Skins
- Defining and Maintaining Folder Control Skins
- Defining and Maintaining Frame Control Skins
- Defining and Maintaining GroupBox Control Skins
- Defining and Maintaining HScroll Control Skins

- Defining and Maintaining JadeDockBar Control Skins
- Defining and Maintaining JadeDockContainer Control Skins
- Defining and Maintaining JadeEditMask Control Skins
- Defining and Maintaining JadeMask Control Skins
- Defining and Maintaining JadeRichText Control Skins
- Defining and Maintaining Label Control Skins
- Defining and Maintaining ListBox Control Skins
- Defining and Maintaining OleControl Control Skins
- Defining and Maintaining OptionButton Control Skins
- Defining and Maintaining Picture Control Skins
- Defining and Maintaining ProgressBar Control Skins
- Defining and Maintaining Sheet Control Skins
- Defining and Maintaining StatusLine Control Skins
- Defining and Maintaining Table Control Skins
- Defining and Maintaining TextBox Control Skins
- Defining and Maintaining VScroll Control Skins

Defining and Maintaining BaseControl Skins

The **Controls** sheet for base controls, shown in the following image, is displayed when the **BaseControl** control is selected in the **Control Type** combo box (the default selection).



The list of control skins for the **BaseControl** control is then displayed in the **Control type skin List** table.

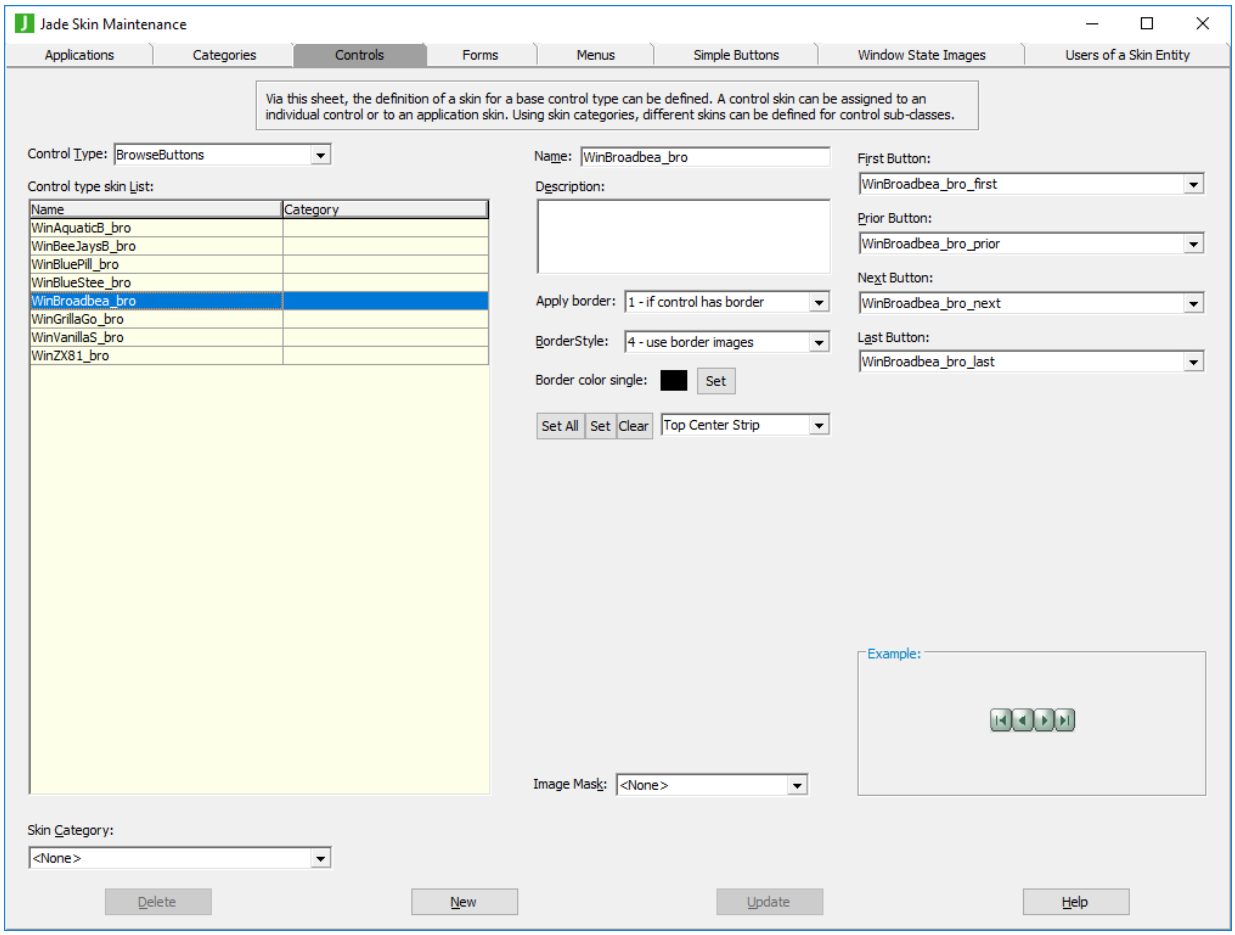
To define or maintain a base control skin

- Perform the actions that you require to define a base control skin or change an existing skin.

For details about the areas of the **Controls** sheet that are common to all control types, see "Using the Controls Sheet".

Defining and Maintaining BrowseButtons Control Skins

The **Controls** sheet, shown in the following image, is displayed when the **BrowseButtons** control is selected in the **Control Type** combo box.



The list of control skins for the **BrowseButtons** control is then displayed in the **Control type skin List** table.

Note Before you can specify the first, prior, next, and last buttons for your browse buttons skin, you must first have defined these buttons by using the **Simple Buttons** sheet. For details, see "Using the Simple Buttons Sheet".

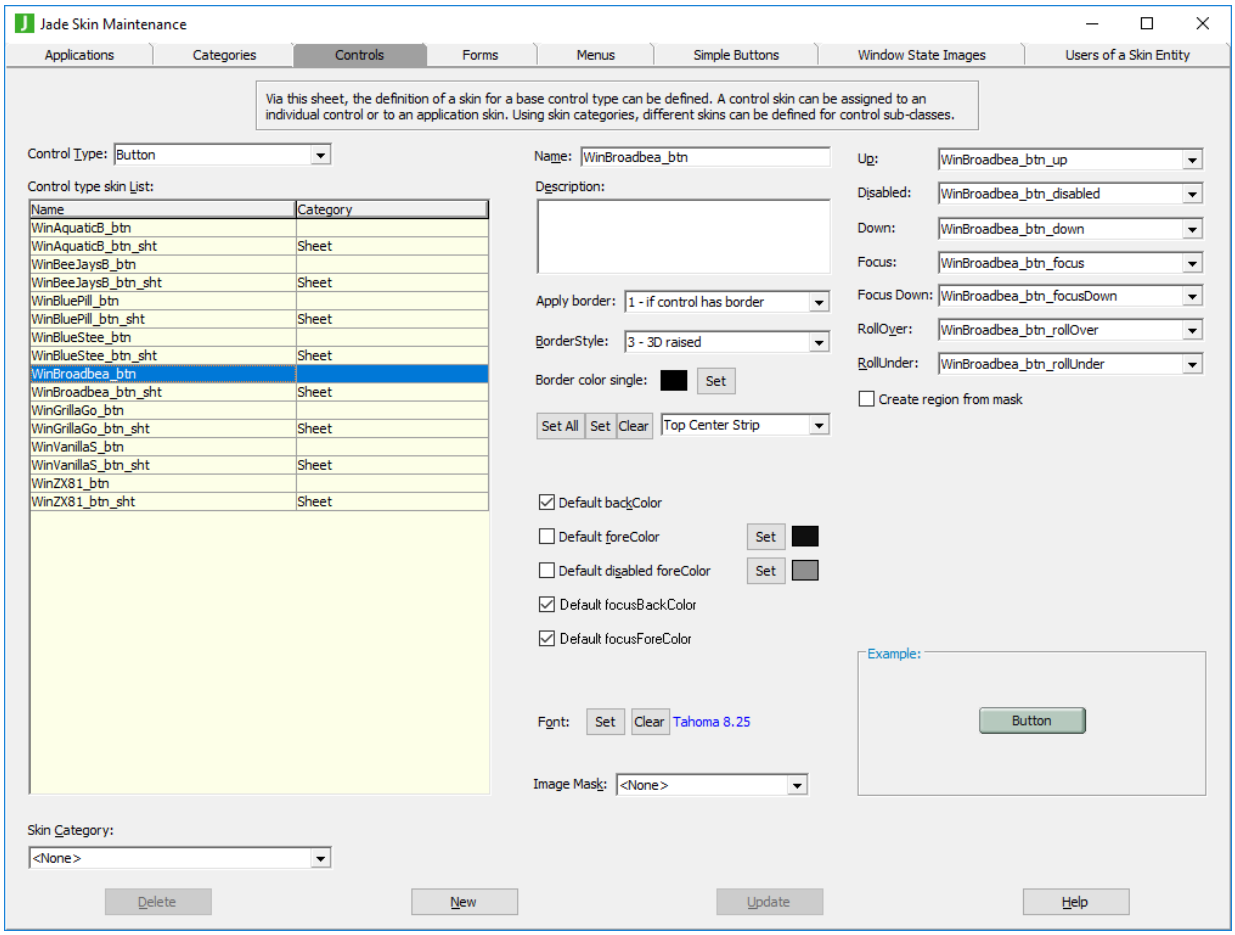
To define or maintain a browse buttons control skin

1. Perform the actions that you require to define a browse buttons control skin or change an existing skin. For details about the areas of the **Controls** sheet that are common to all control types, see "Using the Controls Sheet".

2. In the **First Button**, **Prior Button**, **Next Button**, and **Last Button** combo boxes, select the defined simple button that you want to apply to your browse buttons skin.

Defining and Maintaining Button Control Skins

The **Controls** sheet, shown in the following image, is displayed when the **Button** control is selected in the **Control Type** combo box.



The list of control skins for the **Button** control is then displayed in the **Control type skin List** table.

Note Before you can specify the image states for button controls, you must first have defined these images by using the **Window State Images** sheet. For details, see "Using the Window State Images Sheet".

To define or maintain a button control skin

1. Perform the actions that you require to define a button control skin or change an existing skin. For details about the areas of the **Controls** sheet that are common to all control types, see "Using the Controls Sheet".

2. In the **Up**, **Disabled**, **Down**, **Focus**, **Focus Down**, **RollOver**, and **RollUnder** combo boxes, select the images drawn for each button state.

   If you do not select an image for a button state, the default value of **<None>** indicates that the background color is used to fill the non-border area of the button in the up state.

   Button images are drawn inside any defined border area.

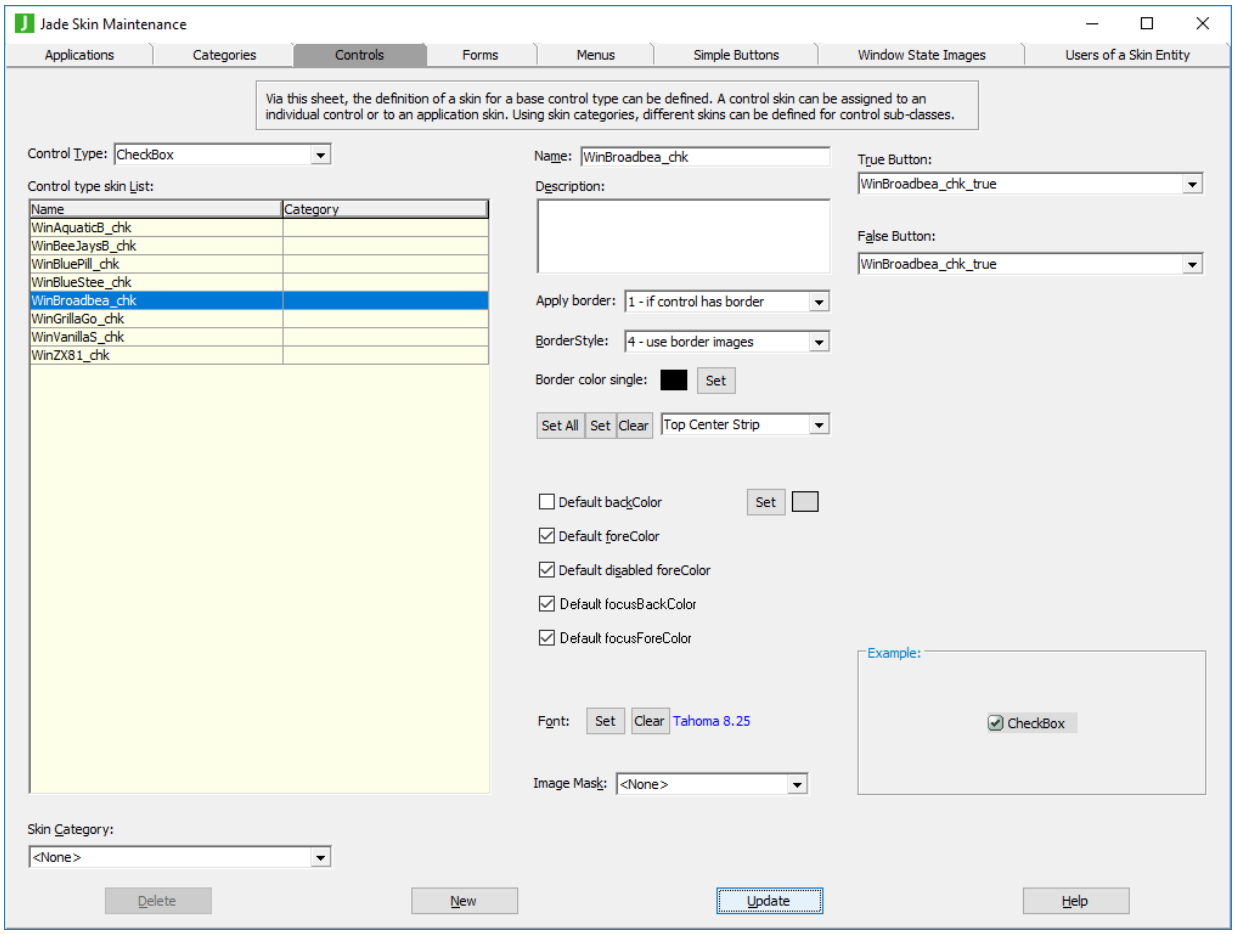
The following image on the left is an example of a button with a raised three-dimensional effect. The image on the right is an example of a button with a sunken three-dimensional effect.

3. Check the **Create region from mask?** check box if you want the image selected in the **Image Mask** check box used to create a non-rectangular region for painting the button.

   When this check box is unchecked (the default value), the full rectangular button area is drawn using the skin. The image mask then applies only to any mouse actions. For example, if the button is an unusual shaped image on a background, the button then displays only the rollover and click images when the mouse is over that special area.

Defining and Maintaining CheckBox Control Skins

The **Controls** sheet, shown in the following image, is displayed when the **CheckBox** control is selected in the **Control Type** combo box.



The list of control skins for the **CheckBox** control is then displayed in the **Control type skin List** table.

Note Before you can specify the **true** and **false** check box buttons for your check box skin, you must first have defined these check box buttons by using the **Simple Buttons** sheet. For details, see "Using the Simple Buttons Sheet".

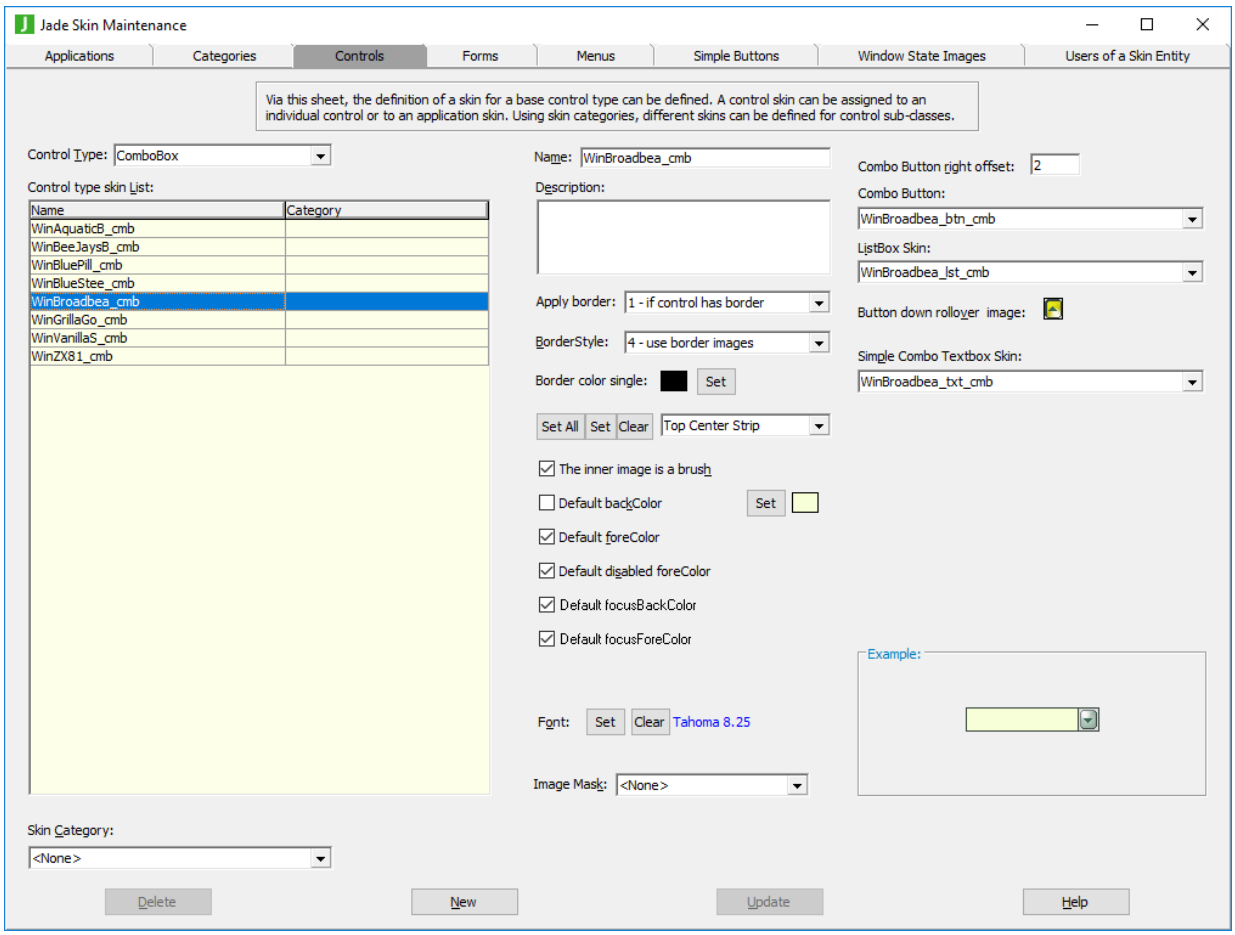
To define or maintain a check box control skin

1. Perform the actions that you require to define a check box control skin or change an existing skin. For details about the areas of the **Controls** sheet that are common to all control types, see "Using the Controls Sheet".
2. In the **True Button** and **False Button** combo boxes, select the **true** and **false** check box button images for the **true** and **false** check box states.

   If you do not select an image, the default check box image is drawn.

Defining and Maintaining ComboBox Control Skins

The **Controls** sheet, shown in the following image, is displayed when the **ComboBox** control is selected in the **Control Type** combo box.



The list of control skins for the **ComboBox** control is then displayed in the **Control type skin List** table.

To define or maintain a combo box control skin

1. Perform the actions that you require to define a combo box control skin or change an existing skin. For details about the areas of the **Controls** sheet that are common to all control types, see "Using the Controls Sheet".
2. In the **Combo Button right offset** text box, specify the number of pixels from the right-hand edge to position the right-hand edge of the combo box button. By default, the combo box button is not offset and it is centered

vertically.

3. In the **Combo Button** and **ListBox Skin** combo boxes, select the images that you want to apply to the button and list box areas of the combo box, respectively.

4. If you want to specify an image for the combo box button rollover state in the down position, click the **Button down rollover image** text box.

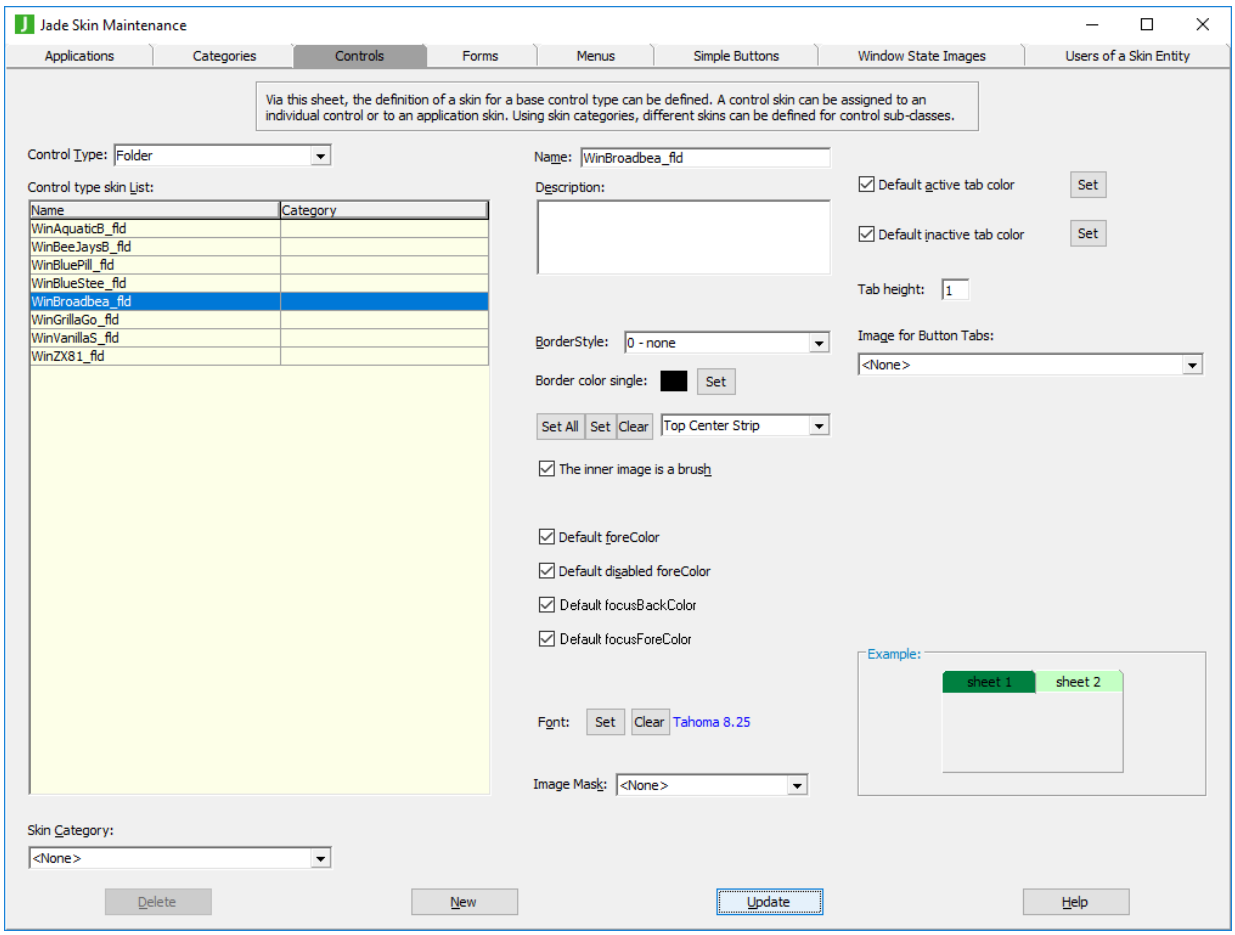
   The common File dialog is then displayed, to enable you to select the existing image that you want displayed. To apply the image, click the **Open** button in the common dialog. If you want to remove a rollover state down position image for the combo box, click the **Cancel** button in the common dialog.

   When you have specified (or cancelled) the appropriate picture file for the combo box button image, the common dialog is then closed and focus returns to the **Controls** sheet of the Jade Skin Maintenance dialog.

5. In the **Simple Combo Textbox Skin** combo box, select the image that you require for the text box area of a simple combo box.

Defining and Maintaining Folder Control Skins

The **Controls** sheet, shown in the following image, is displayed when the **Folder** control is selected in the **Control Type** combo box.



The list of control skins for the **Folder** control is then displayed in the **Control type skin List** table.

To define or maintain a folder control skin

1. Perform the actions that you require to define a folder control skin or change an existing skin. For details about the areas of the **Controls** sheet that are common to all control types, see "Using the Controls Sheet".
2. If you do not want the default active or inactive tab color used for your folder skin, uncheck the **Default active tab color?** or **Default inactive tab color?** check box.

   The common Color dialog is then displayed, to enable you to select the active or inactive color that you require for your folder tabs.

   Alternatively, click the **Set** button at the right of these check boxes if you want to change the default active or inactive tab color by selecting the required value in the common Color dialog.
3. If you want to change the height of the tabs in a folder that uses the default tab height, specify the number of pixels in the **Tab height** text box.

   If the height of the tabs for the folder has been specifically set, the specified value is ignored.

   By default, the tab height is set to zero (**0**), indicating that the default height is the calculated text height using the font of the folder. If you set the tab height to a positive value, each tab is drawn the specified number of pixels high.
4. In the **Image for Button Tabs** combo box, select the image that you require for the tabs in your folder skin when the tabs have the button style applied.

Defining and Maintaining Frame Control Skins

The **Controls** sheet displayed when the **Frame** control is selected in the **Control Type** combo box displays no controls other than those documented under "Using the Controls Sheet".

The list of skins for the frame control is then displayed in the **Control type skin List** table.

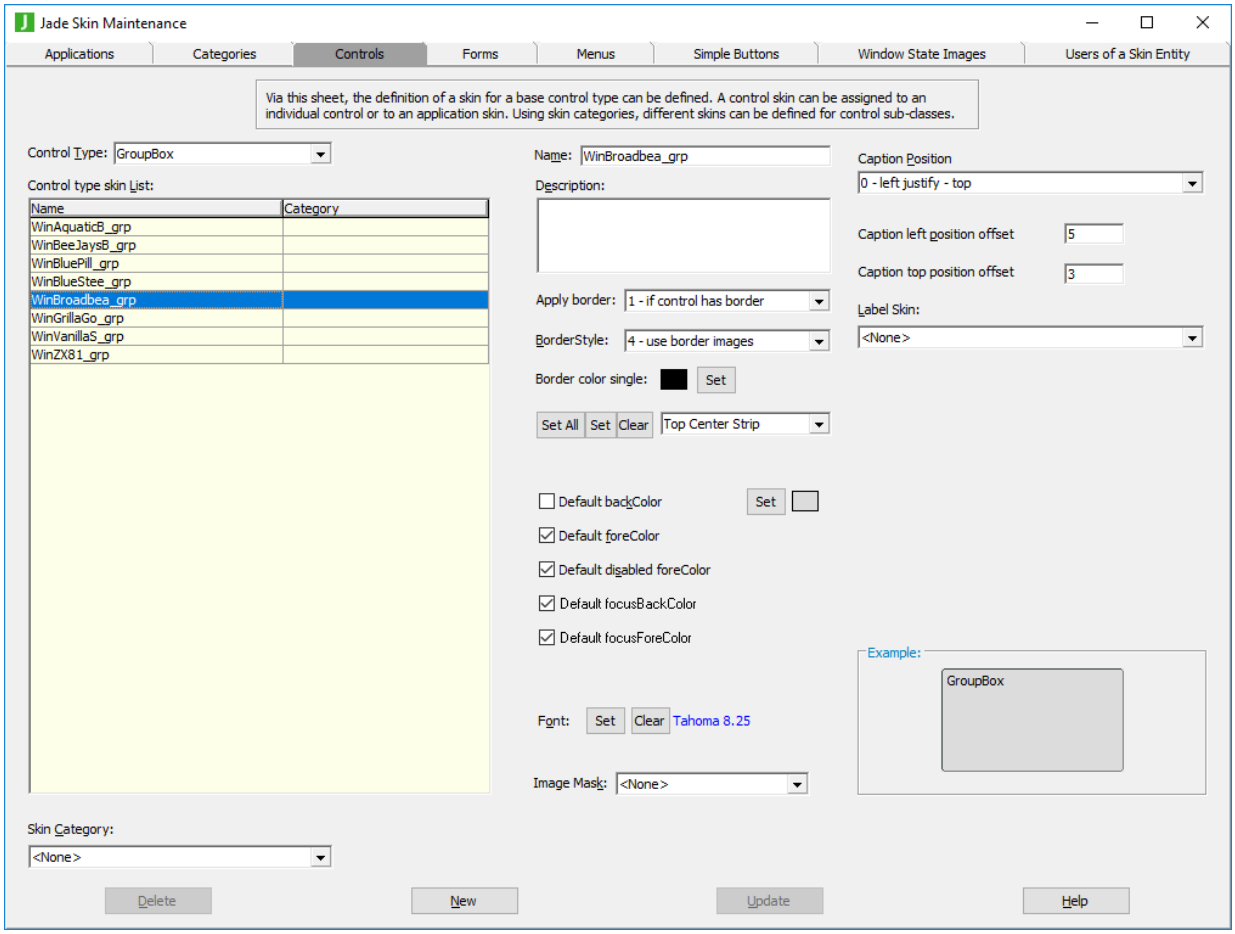
To define or maintain a frame control skin

- Perform the actions that you require to define a frame control skin or change an existing skin.

For details about the areas of the **Controls** sheet that are common to all control types, see "Using the Controls Sheet".

Defining and Maintaining GroupBox Control Skins

The **Controls** sheet, shown in the following image, is displayed when the **GroupBox** control is selected in the **Control Type** combo box. The example in this image shows a group box that has only the top and left strips of the border set and a label skin that draws a border around the text and uses a background brush.



The list of control skins for the **GroupBox** control is then displayed in the **Control type skin List** table.

To define or maintain a group box control skin

1. Perform the actions that you require to define a group box control skin or change an existing skin. For details about the areas of the **Controls** sheet that are common to all control types, see "Using the Controls Sheet".

2. In the **Caption Position** combo box, select the position that you require for the group box caption.

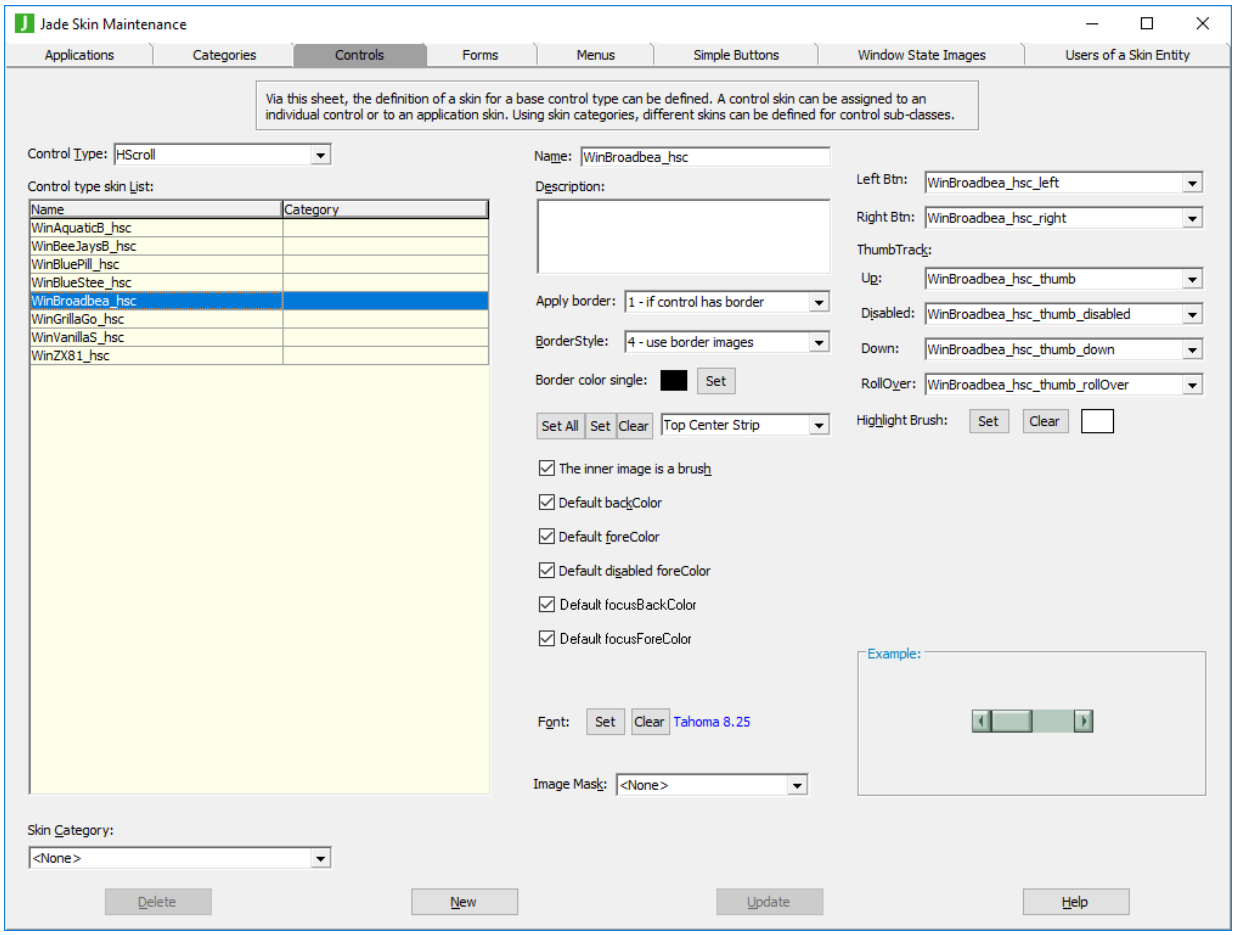
3. In the **Caption Position Left Offset** text box, specify the signed integer modifier for the selected left position of the caption. For example, to position the text nine pixels in the top left corner, select the **0 – left justify – top** caption position in step 2 and then set the caption left offset to **9**. To position the text nine pixels indented from the top right-hand corner, select the **3 – right justify – top** caption position in step 2 and then set the caption left position to **–9** (minus 9).

4. In the **Caption Position Top Offset** text box, specify the signed integer modifier for the selected top position of the caption. For example, to position the text nine pixels from the bottom right of the group box control, select the **5 – right justify – bottom** caption position in step 2 and then set the caption position top offset to **–9** (minus 9).

5. In the **Label Skin** combo box, select the image that you require for the group box label if you want the text drawn as though it was a label so that the text portion of the group box can have its own border and background color, brush, or image.

Defining and Maintaining HScroll Control Skins

The **Controls** sheet, shown in the following image, is displayed when the **HScroll** control is selected in the **Control Type** combo box.



The list of control skins for the horizontal scroll bar control is then displayed in the **Control type skin List** table.

To define or maintain a horizontal scroll bar control skin

1. Perform the actions that you require to define a horizontal scroll bar control skin or change an existing skin. For details about the areas of the **Controls** sheet that are common to all control types, see "Using the Controls Sheet".

2. In the **Left Btn** and **Right Btn** combo boxes, select the images that you require for the left and right buttons on horizontal scroll bars.

3. In the **Up**, **Disabled**, **Down**, and **Rollover** thumb track combo boxes, select the images that you require for the up, down, disabled, and rollover states of the scroll bar thumb track, respectively.

4. Click the **Set** button for the highlight brush if you want to select the image for the brush to be used when the

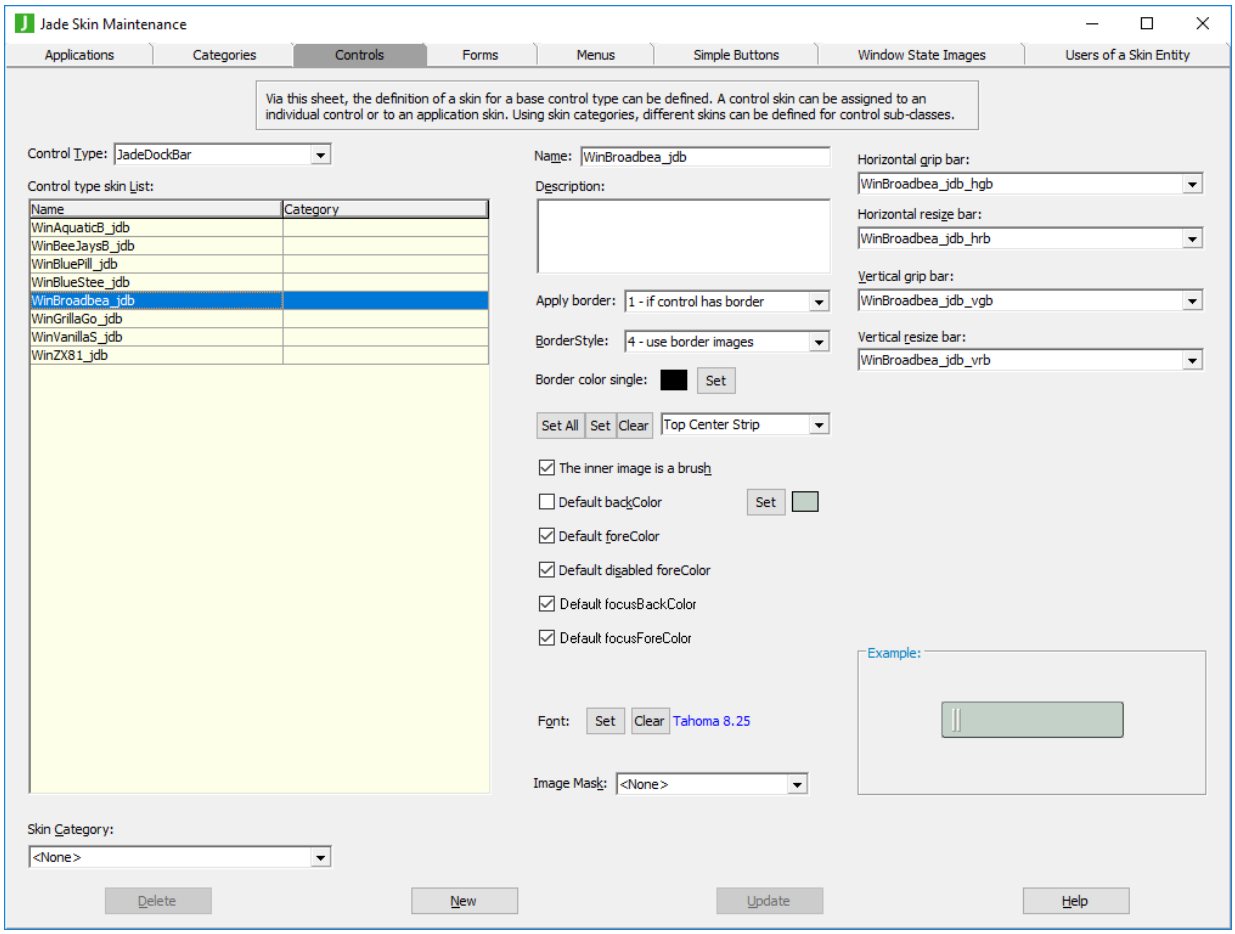
scroll bar stem itself (that is, *not* the thumb track or the arrows) is clicked. When the mouse is down in this situation, that portion of the scroll bar is highlighted. If you select this brush image, highlighting is drawn using this brush or it is drawn with a black brush if no highlight brush is provided.

The common File dialog is then displayed, to enable you to select the existing image that you want displayed for the highlight brush. To apply the image, click the **Open** button in the common dialog. When you have selected the appropriate picture file for the image, the common dialog is then closed and focus returns to the **Controls** sheet of the Jade Skin Maintenance dialog. In addition, your selected image is displayed in the text box at the right of the **Highlight Brush** caption.

5. If you want to remove the current highlight brush image, click the **Clear** button at the right of the **Highlight Brush** caption. The highlight brush image is then removed from the text box at the right of the **Highlight Brush** caption on the **Controls** form.

Defining and Maintaining JadeDockBar Control Skins

The **Controls** sheet, shown in the following image, is displayed when the **JadeDockBar** control is selected in the **Control Type** combo box.



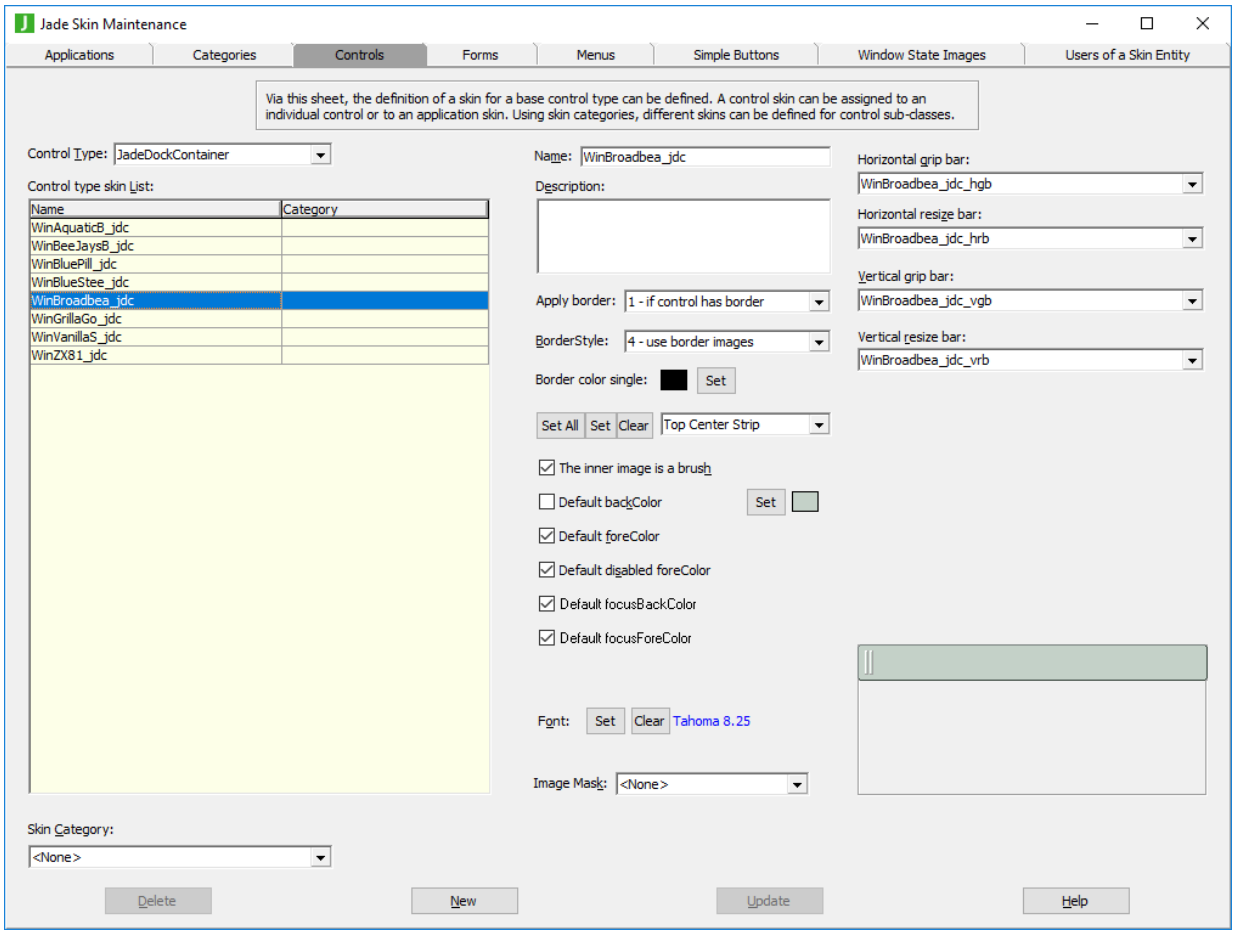
The list of control skins for the **JadeDockBar** control is then displayed in the **Control type skin List** table.

To define or maintain a dock bar control skin

1. Perform the actions that you require to define a dock bar control skin or change an existing skin. For details about the areas of the **Controls** sheet that are common to all control types, see "Using the Controls Sheet".
2. If you want to select the window state image that is displayed for the required horizontal grip bar, horizontal resize bar, vertical grip bar, or vertical resize bar, select the required image from the appropriate combo box at the upper right of the **Controls** sheet for the **JadeDockBar** control.
3. If you want to remove the current grip or resize bar image, click the **Clear** button at the right of the appropriate caption. The image is then removed from the text box at the right of the caption on the **Controls** form.

Defining and Maintaining JadeDockContainer Control Skins

The **Controls** sheet, shown in the following image, is displayed when the **JadeDockContainer** control is selected in the **Control Type** combo box.



The list of control skins for the **JadeDockContainer** control is then displayed in the **Control type skin List** table.

To define or maintain a dock container control skin

1. Perform the actions that you require to define a dock container control skin or change an existing skin. For details about the areas of the **Controls** sheet that are common to all control types, see "Using the Controls Sheet".

2. If you want to select the window state image that is displayed for the required horizontal grip bar, horizontal resize bar, vertical grip bar, or vertical resize bar, select the required image from the appropriate combo box at the upper right of the **Controls** sheet for the **JadeDockContainer** control.

3. If you want to remove the current grip or resize bar image, click the **Clear** button at the right of the appropriate caption. The image is then removed from the text box at the right of the caption on the **Controls** form.

Defining and Maintaining JadeEditMask Control Skins

The **Controls** sheet displayed when the **JadeEditMask** control is selected in the **Control Type** combo box displays no controls other than those documented under "Using the Controls Sheet". The list of skins for the edit mask control is then displayed in the **Control type skin List** table.

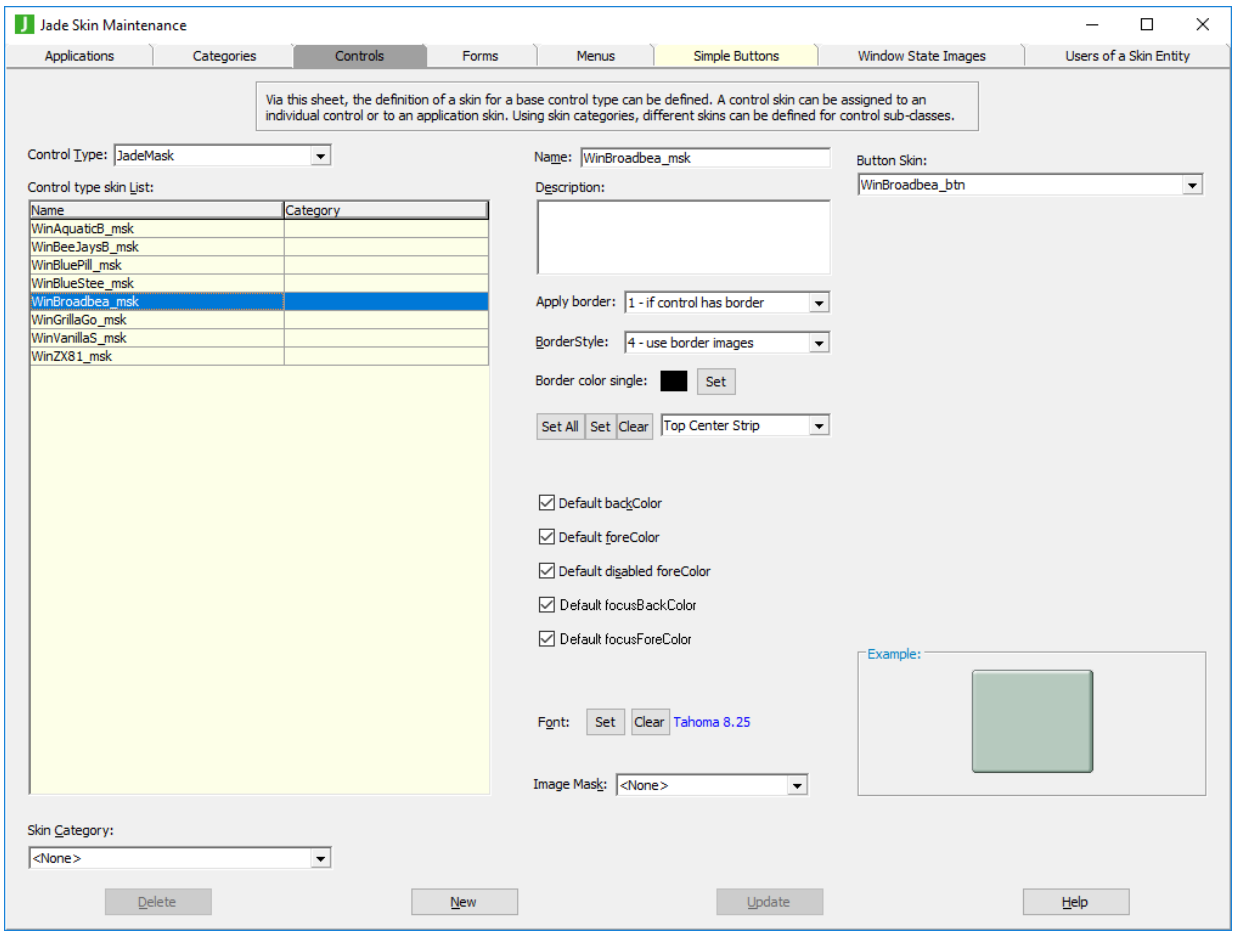
To define or maintain an edit mask control skin

- Perform the actions that you require to define an edit mask control skin or change an existing skin.

For details about the areas of the **Controls** sheet that are common to all control types, see "Using the Controls Sheet".

Defining and Maintaining JadeMask Control Skins

The **Controls** sheet, shown in the following image, is displayed when the **JadeMask** control is selected in the **Control Type** combo box.



The list of control skins for the **JadeMask** control is then displayed in the **Control type skin List** table.

To define or maintain a mask control skin

1. Perform the actions that you require to define a mask control skin or change an existing skin. For details about the areas of the **Controls** sheet that are common to all control types, see "Using the Controls Sheet".
2. In the **Button Skin** combo box, select the image that you want to apply when the **JadeMask** control acts as a button.

Defining and Maintaining JadeRichText Control Skins

The **Controls** sheet displayed when the **JadeRichText** control is selected in the **Control Type** combo box displays no controls other than those documented under "Using the Controls Sheet".

The list of skins for the JADE rich text control is then displayed in the **Control type skin List** table.

To define or maintain a JADE rich text control skin

- Perform the actions that you require to define a JADE rich text control skin or change an existing skin.

For details about the areas of the **Controls** sheet that are common to all control types, see "Using the Controls Sheet".

Defining and Maintaining Label Control Skins

The **Controls** sheet displayed when the **Label** control is selected in the **Control Type** combo box displays no controls other than those documented under "Using the Controls Sheet".

The list of skins for the label control is then displayed in the **Control type skin List** table.

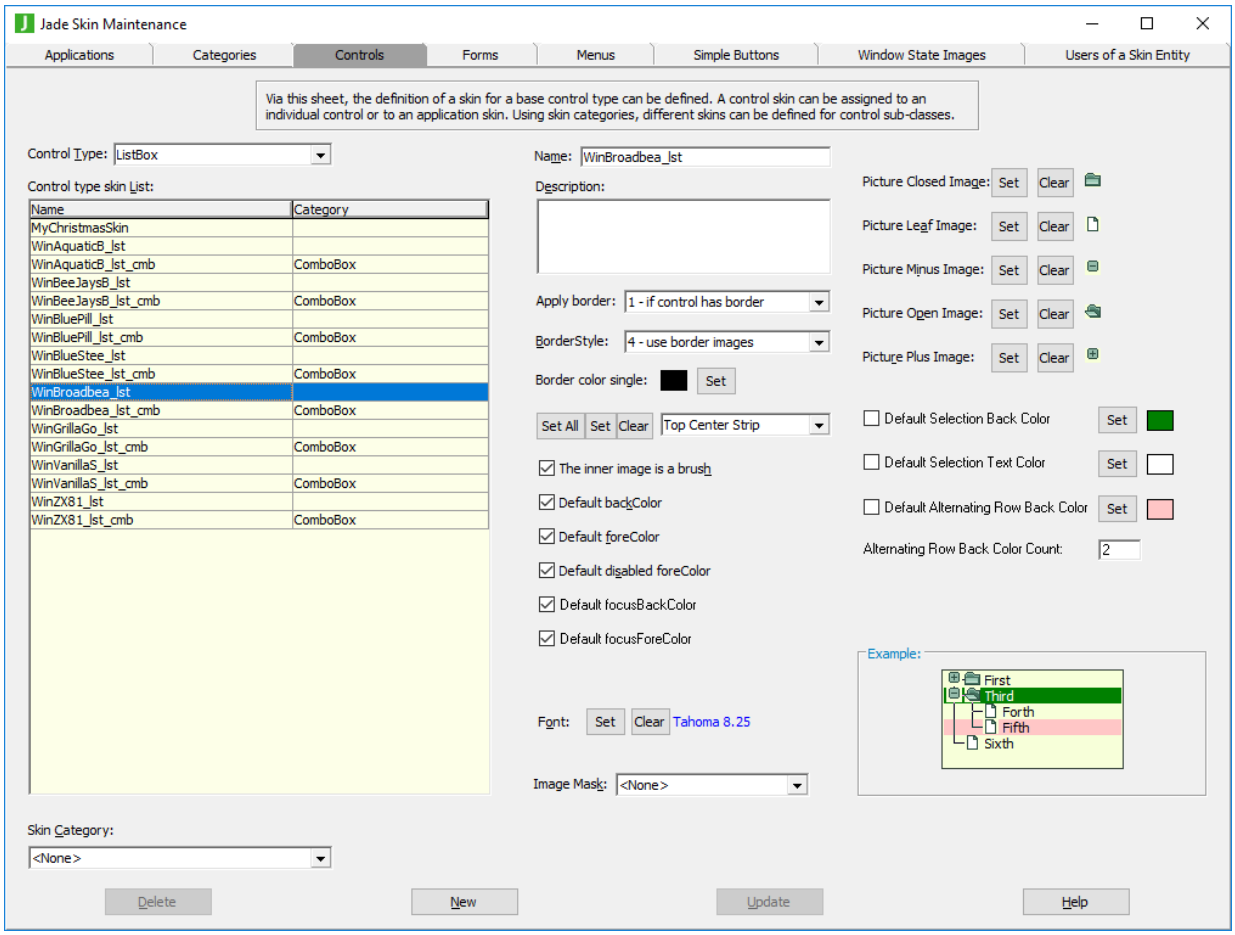
To define or maintain a label control skin

- Perform the actions that you require to define a label control skin or change an existing skin.

For details about the areas of the **Controls** sheet that are common to all control types, see "Using the Controls Sheet".

Defining and Maintaining ListBox Control Skins

The **Controls** sheet, shown in the following image, is displayed when the **ListBox** control is selected in the **Control Type** combo box.



The list of control skins for the **ListBox** control is then displayed in the **Control type skin List** table.

To define or maintain a list box control skin

1. Perform the actions that you require to define a list box control skin or change an existing skin. For details about the areas of the **Controls** sheet that are common to all control types, see "Using the Controls Sheet".

2. If you want to select the image that is displayed for the closed, leaf, minus, open, or plus picture of the list box control that supports pictures, click the appropriate **Set** button at the right of the required caption.

   The common File dialog is then displayed, to enable you to select the existing image that you want displayed for the list box picture. To apply the image, click the **Open** button in the common dialog.

   When you have selected the appropriate picture file for the image, the common dialog is then closed and focus returns to the **Controls** sheet of the Jade Skin Maintenance dialog. In addition, your selected image is displayed in the text box at the right of the appropriate caption.

3. If you do not want a default background color for selected items in a list box, uncheck the **Default Selection Back Color** check box.

The default value of **#80000000** (that is, transparent) means that the default selection background color defined by Windows is used, but you can select your required selected item background color by unchecking this check box and then selecting a color or defining your own custom color in the common Color dialog that is then displayed.

4. If you do not want a default text color for selected items in a list box, uncheck the **Default Selection Text Color** check box.

   The default value of **#80000000** (that is, transparent) means that the default selection text color defined by Windows is used, but you can select your required selected item background color by unchecking this check box and then selecting a color or defining your own custom color in the common Color dialog that is then displayed.

5. If you want a default background color for alternating list box rows, check the **Default Alternating Row Back Color** check box. The default color is **Azure**, but you can select your required alternating background color by clicking this check box and then selecting a color or defining your own custom color in the common Color dialog that is then displayed.

   If the value of the **Alternating Row Back Color Count** text box is zero (**0**) or less, the value of the **Default Alternating Row Back Color** check box is ignored and does not apply.

6. If you checked the **Default Alternating Row Back Color** check box in the previous step of this instruction, specify the number of visible entry list rows at which the alternating background color is displayed. If this text box contains the default value of zero (**0**) or less, the background color of each visible list box entry defaults to the value of the **backColor** property of the **ListBox** control and the **Default Alternating Row Back Color** check box is ignored.

   For example, if the count is **2**, the first, third, fifth, and so on visible entries in the list default to the value of the **backColor** property of the **ListBox** control, while the second, fourth, sixth, and so on visible entries default to the value of the **Default Alternating Row Back Color** check box.

7. If you want to remove the current list box picture image, click the **Clear** button at the right of the appropriate caption.

The image is then removed from the text box at the right of the caption on the **Controls** form.

Defining and Maintaining OleControl Control Skins

The **Controls** sheet displayed when the **OleControl** control is selected in the **Control Type** combo box displays no controls other than those documented under "Using the Controls Sheet". The list of **OLE** control skins is then displayed in the **Control type skin List** table.

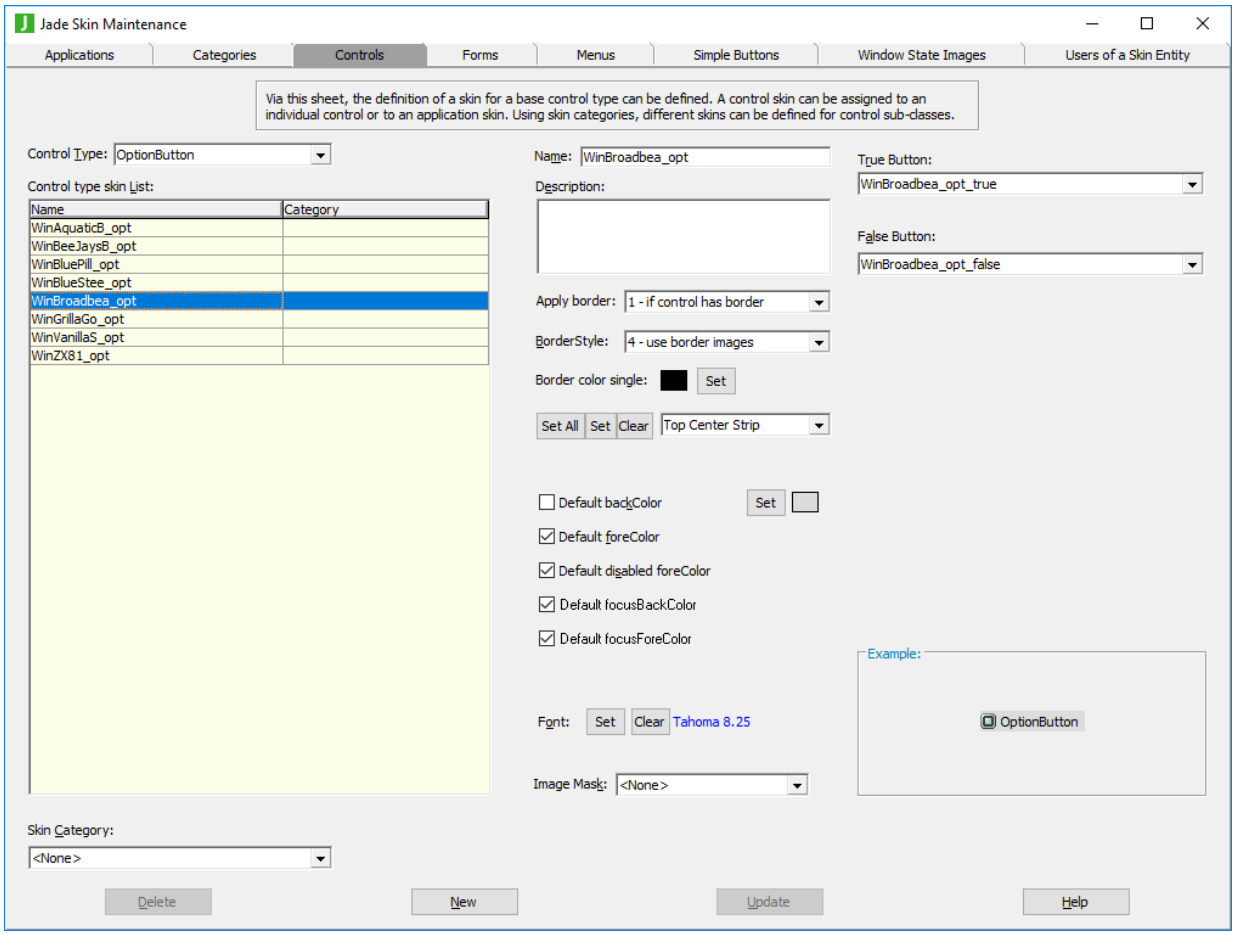
To define or maintain an OLE control skin

- Perform the actions that you require to define an **OleControl** control skin or change an existing skin.

For details about the areas of the **Controls** sheet that are common to all control types, see "Using the Controls Sheet".

Defining and Maintaining OptionButton Control Skins

The **Controls** sheet, shown in the following image, is displayed when the **OptionButton** control is selected in the **Control Type** combo box.



Control skins for the **OptionButton** control is then displayed in the **Control type skin List** table.

Note Before you can specify the **true** and **false** option buttons for your option button skin, you must first have defined these option buttons by using the **Simple Buttons** sheet. For details, see "Using the Simple Buttons Sheet".

To define or maintain an option button control skin

1. Perform the actions that you require to define an option button control skin or change an existing skin. For details about the areas of the **Controls** sheet that are common to all control types, see "Using the Controls Sheet".

2. In the **True Button** and **False Button** combo boxes, select the **true** and **false** option button images for the **true** and **false** option button states. If you do not select an image, the default option button image is drawn.

Defining and Maintaining Picture Control Skins

The **Controls** sheet displayed when the **Picture** control is selected in the **Control Type** combo box displays no controls other than those documented under "Using the Controls Sheet". The list of picture control skins is then displayed in the **Control type skin List** table.

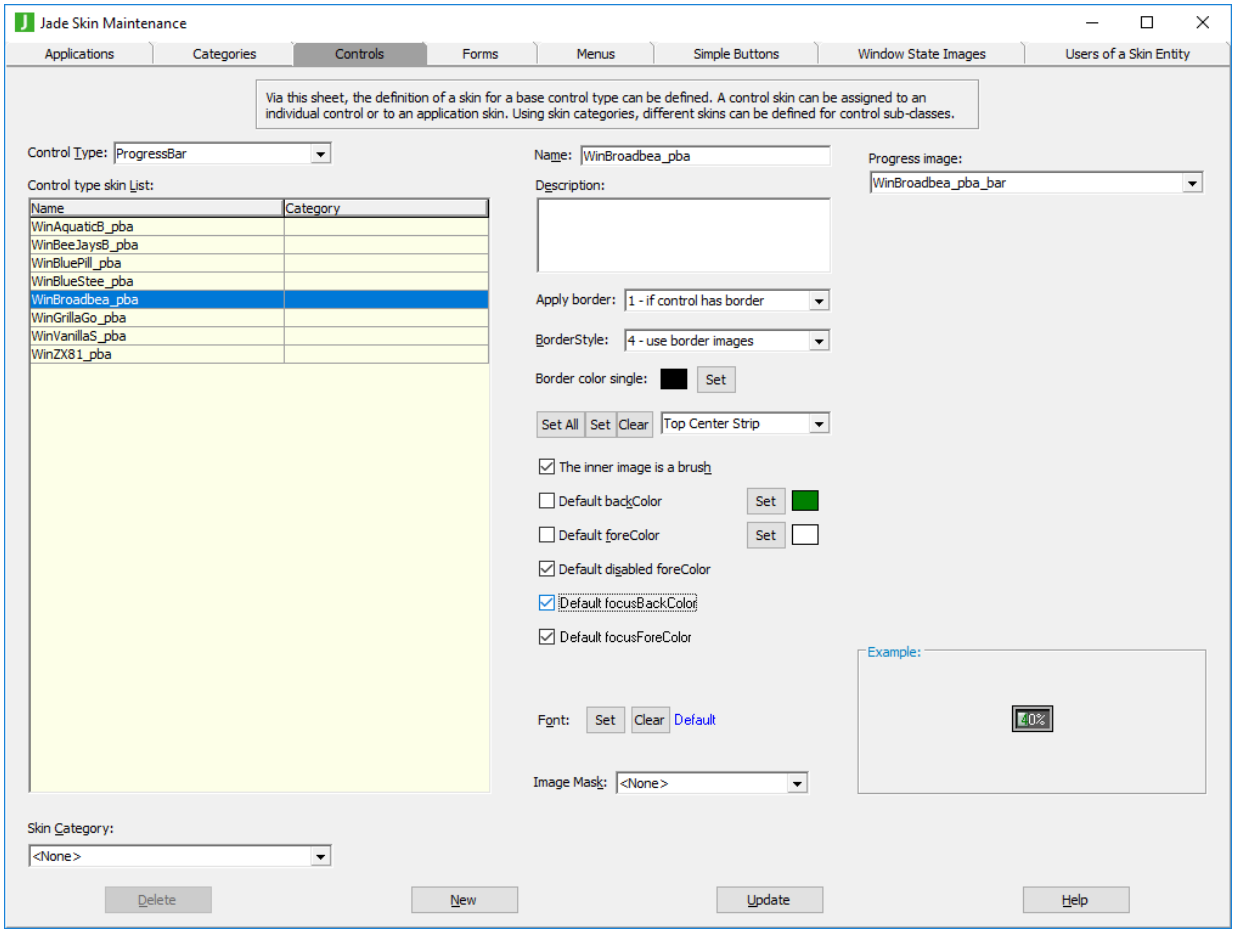
To define or maintain a picture control skin

- Perform the actions that you require to define a **Picture** control skin or change an existing skin.

For details about the areas of the **Controls** sheet that are common to all control types, see "Using the Controls Sheet".

Defining and Maintaining ProgressBar Control Skins

The **Controls** sheet, shown in the following image, is displayed when the **ProgressBar** control is selected in the **Control Type** combo box.



The list of control skins for the **ProgressBar** control is then displayed in the **Control type skin List** table.

To define or maintain a progress bar control skin

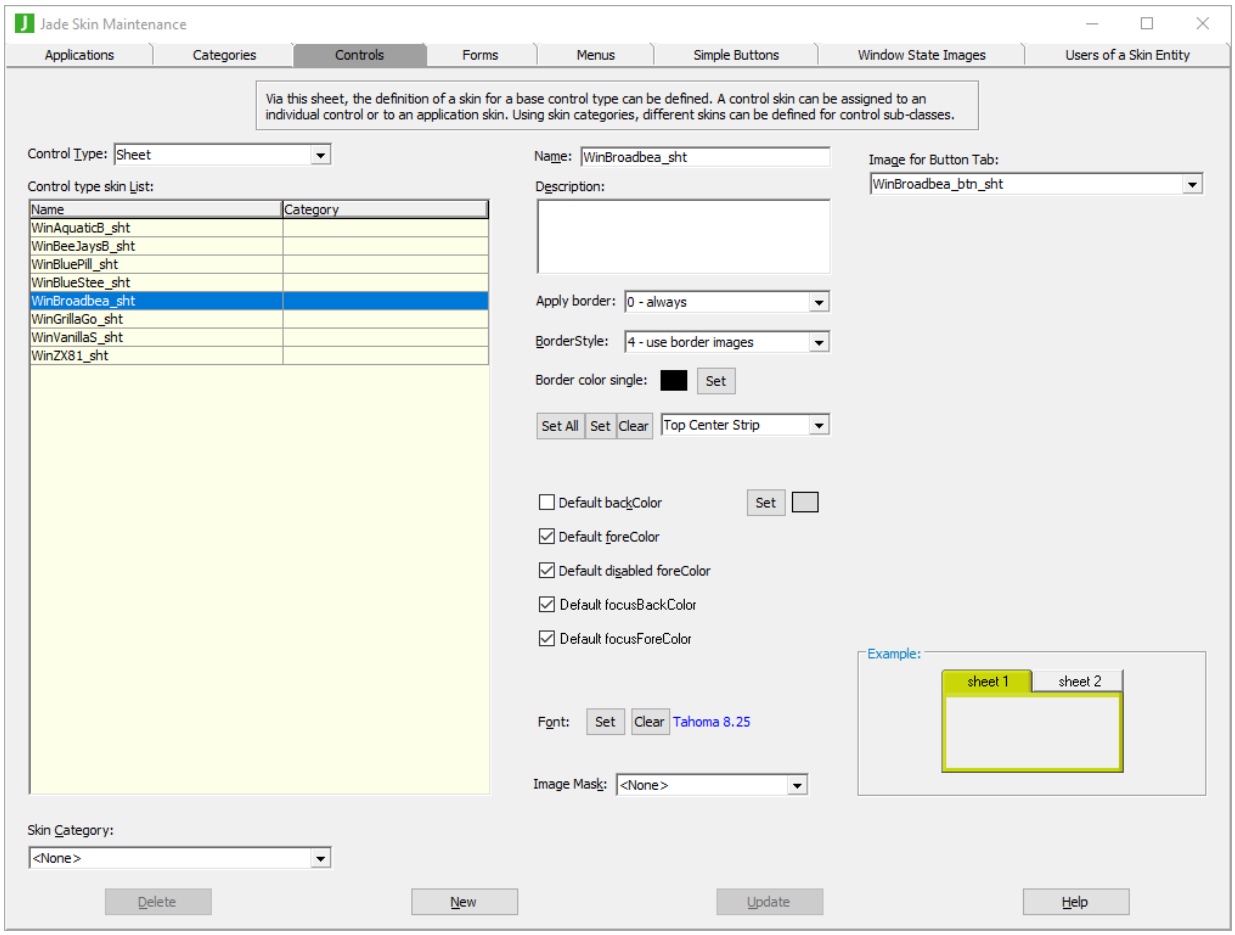
1. Perform the actions that you require to define a progress bar control skin or change an existing skin. For details about the areas of the **Controls** sheet that are common to all control types, see "Using the Controls

Sheet".

2. In the **Progress image** combo box, select the image that you want to apply to **ProgressBar** controls.

Defining and Maintaining Sheet Control Skins

The **Controls** sheet, shown in the following image, is displayed when the **Sheet** control is selected in the **Control Type** combo box.



The list of controls skins for the **Sheet** control in then displayed in the **Control type skin List** table.

To define or maintain a sheet control skin

1. Perform the actions that you require to define a **Sheet** control skin or change an existing skin.

   For details about the areas of the **Controls** sheet that are common to all control types, see "Using the Controls Sheet".

2. If you want to set the skin used to draw the tab of a folder for a sheet, select the appropriate button image in the **Image for Button Tab** combo box. If you set a tab button for the sheet, any skin button setting of the folder is ignored.

The main use of defining a button skin for a sheet is to enable each tab of a folder to be drawn with different images and colors. To achieve this, define several sheet skins with different categories and then set the appropriate category on the sheets that you want to use each specific sheet skin.

Defining and Maintaining StatusLine Control Skins

The **Controls** sheet displayed when the **StatusLine** control is selected in the **Control Type** combo box displays no controls other than those documented under "Using the Controls Sheet". The list of status line control skins is then displayed in the **Control type skin List** table.

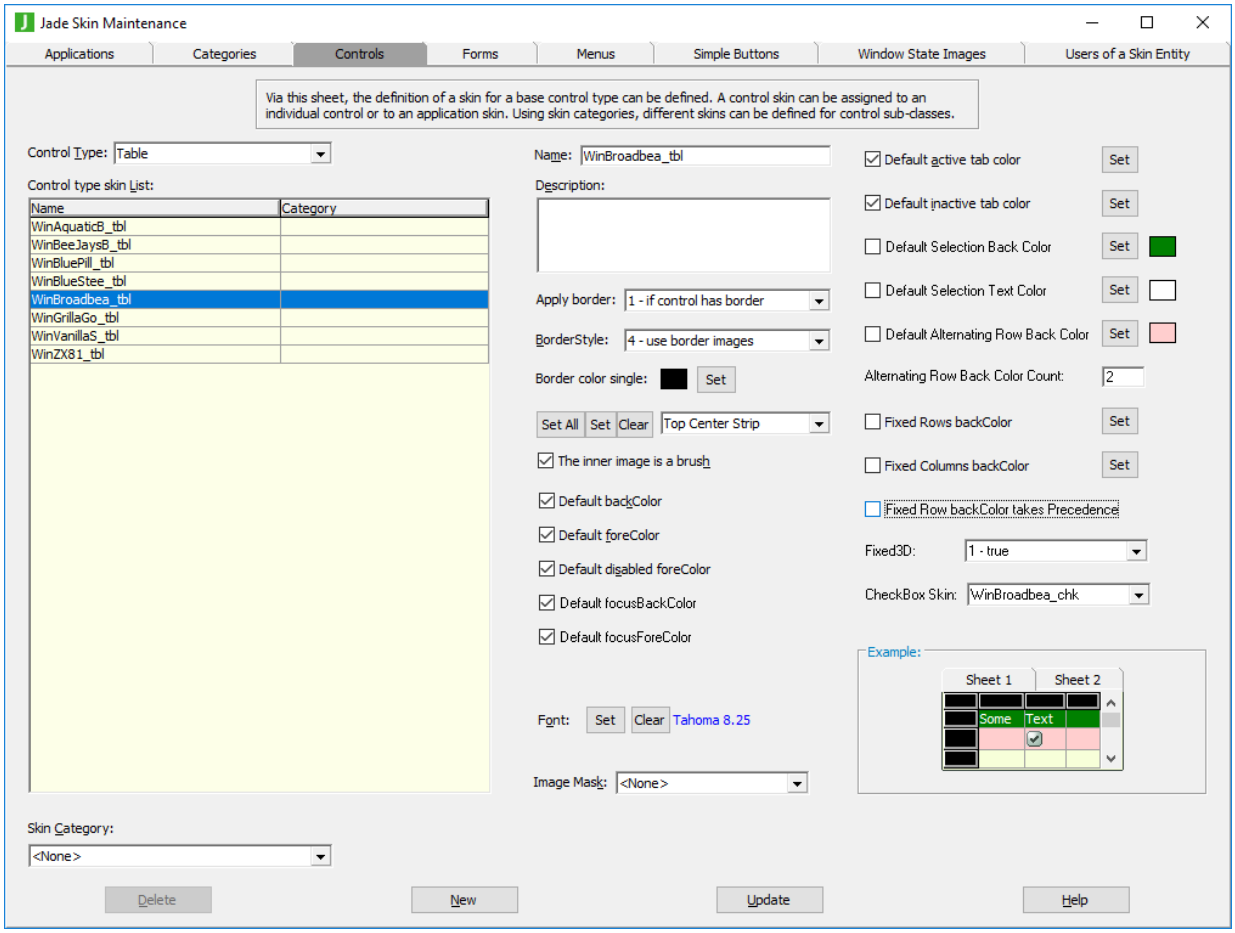
To define or maintain a status line control skin

- Perform the actions that you require to define a **StatusLine** control skin or change an existing skin.

For details about the areas of the **Controls** sheet that are common to all control types, see "Using the Controls Sheet".

Defining and Maintaining Table Control Skins

The **Controls** sheet, shown in the following image, is displayed when the **Table** control is selected in the **Control Type** combo box.



The list of control skins for the **Table** control is then displayed in the **Control type skin List** table.

To define or maintain a table control skin

1. Perform the actions that you require to define a **Table** control skin or change an existing skin.

   For details about the areas of the **Controls** sheet that are common to all control types, see "Using the Controls Sheet".

2. If you do not want the default active or inactive tab color used for your table skin, uncheck the **Default active tab color?** or **Default inactive tab color?** check box, respectively.

   The common Color dialog is then displayed, to enable you to select the active or inactive color that you require for your table tabs. Alternatively, click the **Set** button at the right of these check boxes if you want to change the default active or inactive tab color by selecting the required value in the common Color dialog.

3. If you do not want a default background color for selected items in a table, uncheck the **Default Selection Back Color** check box.

   The default value of **#80000000** (that is, transparent) means that the default selection background color defined by Windows is used, but you can select your required selected item background color by unchecking this check box and then selecting a color or defining your own custom color in the common Color dialog that is then displayed.

4. If you do not want a default text color for selected items in a table, uncheck the **Default Selection Text Color** check box.

   The default value of **#80000000** (that is, transparent) means that the default selection text color defined by Windows is used, but you can select your required selected item background color by unchecking this check box and then selecting a color or defining your own custom color in the common Color dialog that is then displayed.

5. If you want a default background color for alternating table rows, check the **Default Alternating Row Back Color?** check box. The default color is **Azure**, but you can select your required alternating background color by clicking this check box and then selecting a color or defining your own custom color in the common Color dialog that is then displayed.

   If the value of the **Alternating Row Back Color Count** text box is zero (**0**) or less, the value of the **Default Alternating Row Back Color** check box is ignored and does not apply.

6. If you checked the **Default Alternating Row Back Color** check box in the previous step of this instruction, specify the number of table rows at which the alternating background color of each visible non-fixed row and non-fixed cell is displayed. If this text box contains the default value of zero (**0**) or less, the background color of each non-fixed cell defaults to the value of the **backColor** property of the sheet and the **Default Alternating Row Back Color** check box is ignored.

   For example, if the count is **2**, the first, third, fifth, and so on non-fixed rows and the non-fixed cells in that row default to the value of the **backColor** property of the sheet, while the second, fourth, sixth, and so on non-fixed rows and the non-fixed cells in that row default to the value of the **alternatingRowBackColor** property.

7. If you want to specify the background color of fixed rows or columns, check the **Fixed Rows backColor** or **Fixed Columns backColor** check box, respectively.

   The common Color dialog is then displayed, to enable you to select the background color that you require for your fixed rows or columns. Alternatively, click the **Set** button at the right of these check boxes if you want to change the background color or rows or columns by selecting the required value in the common Color dialog.

8. If you want the background color of both a fixed row and a fixed column drawn using the value of the **fixedRowsBackColor** property, check the **Fixed Row backColor takes Precedence** check box (that is,

determine the color to be used for a cell that is in both a fixed row and a fixed column).

By default, cells that are in both a fixed row and a fixed column are drawn using the value of the **fixedColumnsBackColor** property.

Note If a fixed cell has a specific **backColor** set via a cell, row, or column **backColor** property value, the skin back color values that you set in steps 5 and 6 of this instruction are ignored.

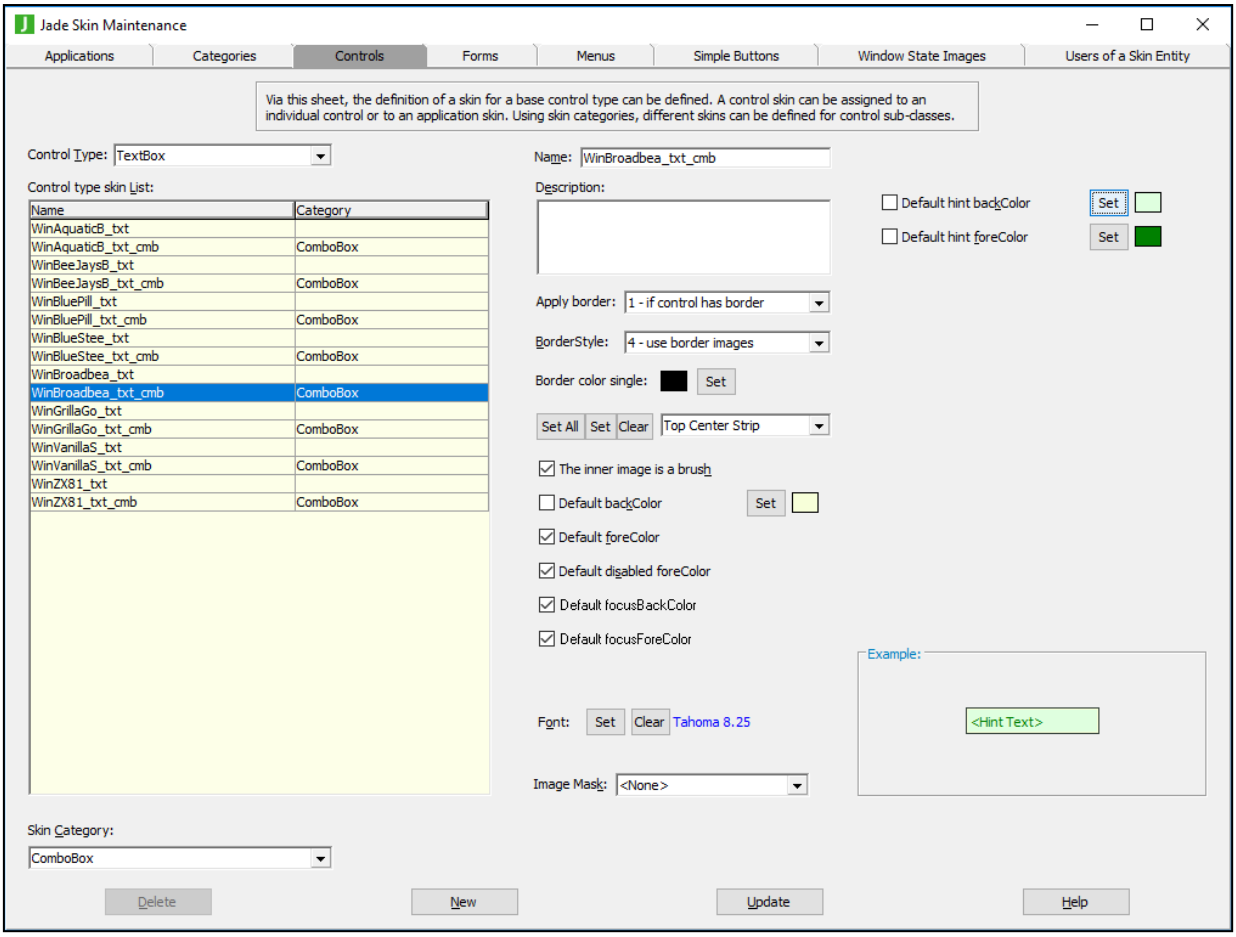
9. If you want fixed three-dimensional rows in your tables, in the **Fixed3D** combo box (which defaults to zero (**0**) **- false**), select **1 - true** or **2 - use control setting**.

10. If you want a check box skin to be used when drawing a cell that has the **inputType** property set to **InputType_CheckBox** or a cell control set to a check box control, select the check box skin that you required from the list of all available check box skins displayed in the list portion of the **CheckBox Skin** combo box.

    The default value of **<none>** indicates that check boxes in table cells are drawn without a skin.

Defining and Maintaining TextBox Control Skins

The **Controls** sheet, shown in the following image, is displayed when the **TextBox** control is selected in the **Control Type** combo box.



The list of controls for the **TextBox** control is then displayed in the **Control type skin List** table.

To define or maintain a text box control skin

1. Perform the actions that you require to define a **TextBox** control skin or change an existing skin. For details about the areas of the **Controls** sheet that are common to all control types, see "Using the Controls Sheet".

2. If you do not want a default background color for hint text in a text box, uncheck the **Default hint backColor** check box. The common Color dialog is then displayed, to enable you to select the hint background color that you require for your text box skin. Alternatively, click the **Set** button at the right of the check box if you want to change the default hint background color by selecting the required value in the common Color dialog.

   The default value of **#80000000** means that the default text box hint background color defined by the **TextBox** class **hintBackColor** property value is used, but you can select your required hint background color by unchecking this check box and then selecting a color or defining your own custom color in the common Color dialog that is then displayed.

   The value applies only if the **TextBox** class **hintText** property is not null (**""**), the hint text is displayed (that is, the text box is empty), and the **JadeSkinTextBox** class hint value is not **#80000000** (the default). The property value is also ignored when the hint text is displayed and the text box text is disabled, in which case the text box is drawn in its disabled state (with the hint text still displayed).

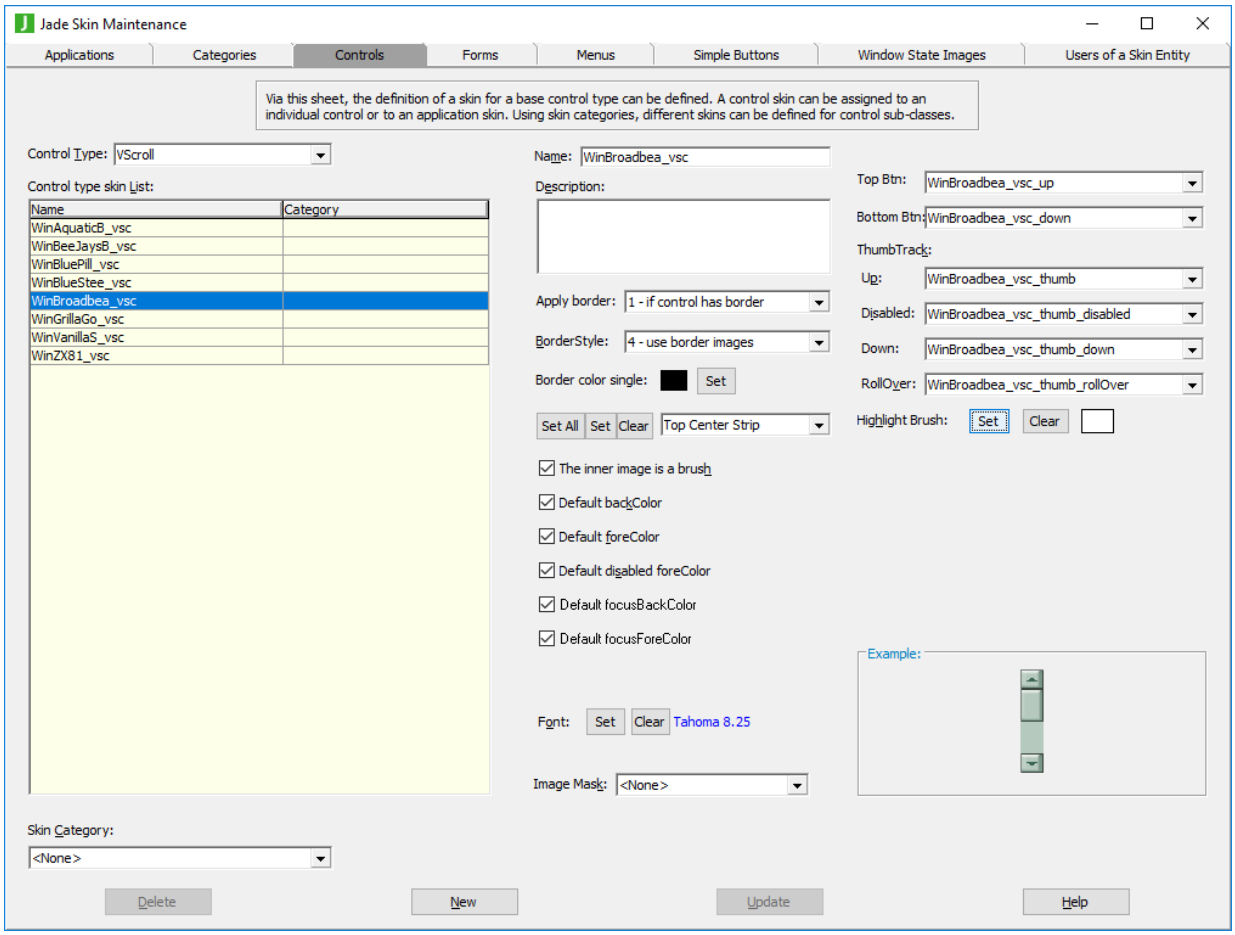
3. If you do not want a default foreground (text) color for hint text in a text box, uncheck the **Default hint foreColor** check box. The common Color dialog is then displayed, to enable you to select the hint foreground color that you require for your text box skin. Alternatively, click the **Set** button at the right of the check box if you want to change the default foreground hint text color by selecting the required value in the common Color dialog.

   The default value of **#80000000** means that the default text box hint foreground color defined by the **TextBox** class **hintText** property value is used, but you can select your required hint foreground color by unchecking this check box and then selecting a color or defining your own custom color in the common Color dialog that is then displayed.

   The value applies only if the **TextBox** class **hintText** property is not null (**""**), the hint text is displayed (that is, the text box is empty), and the **JadeSkinTextBox** class hint value is not **#80000000** (the default). The property value is also ignored when the hint text is displayed and the text box text is disabled, in which case the text box is drawn in its disabled state (with the hint text still displayed).

Defining and Maintaining VScroll Control Skins

The **Controls** sheet, shown in the following image, is displayed when the **VScroll** control is selected in the **Control Type** combo box.



The list of control skins for the vertical scroll bar control is then displayed in the **Control type skin List** table.

To define or maintain a vertical scroll bar control skin

1. Perform the actions that you require to define a vertical scroll bar control skin or change an existing skin.

   For details about the areas of the **Controls** sheet that are common to all control types, see "Using the Controls Sheet".

2. In the **Left Btn** and **Right Btn** combo boxes, select the images that you require for the left and right buttons on vertical scroll bars.

3. In the **Up**, **Disabled**, **Down**, and **Rollover** thumb track combo boxes, select the images that you require for the up, down, disabled, and rollover states of the scroll bar thumb track, respectively.

4. Click the **Set** button for the highlight brush if you want to select the image for the brush to be used when the scroll bar stem itself (that is, *not* the thumb track or the arrows) is clicked.

When the mouse is down in this situation, that portion of the scroll bar is highlighted. If you select this brush image, highlighting is drawn using this brush or it is drawn with a black brush if no highlight brush is provided.

The common File dialog is then displayed, to enable you to select the existing image that you want displayed for the highlight brush. To apply the image, click the **Open** button in the common dialog.

When you have selected the appropriate picture file for the image, the common dialog is then closed and focus returns to the **Controls** sheet of the Jade Skin Maintenance dialog.

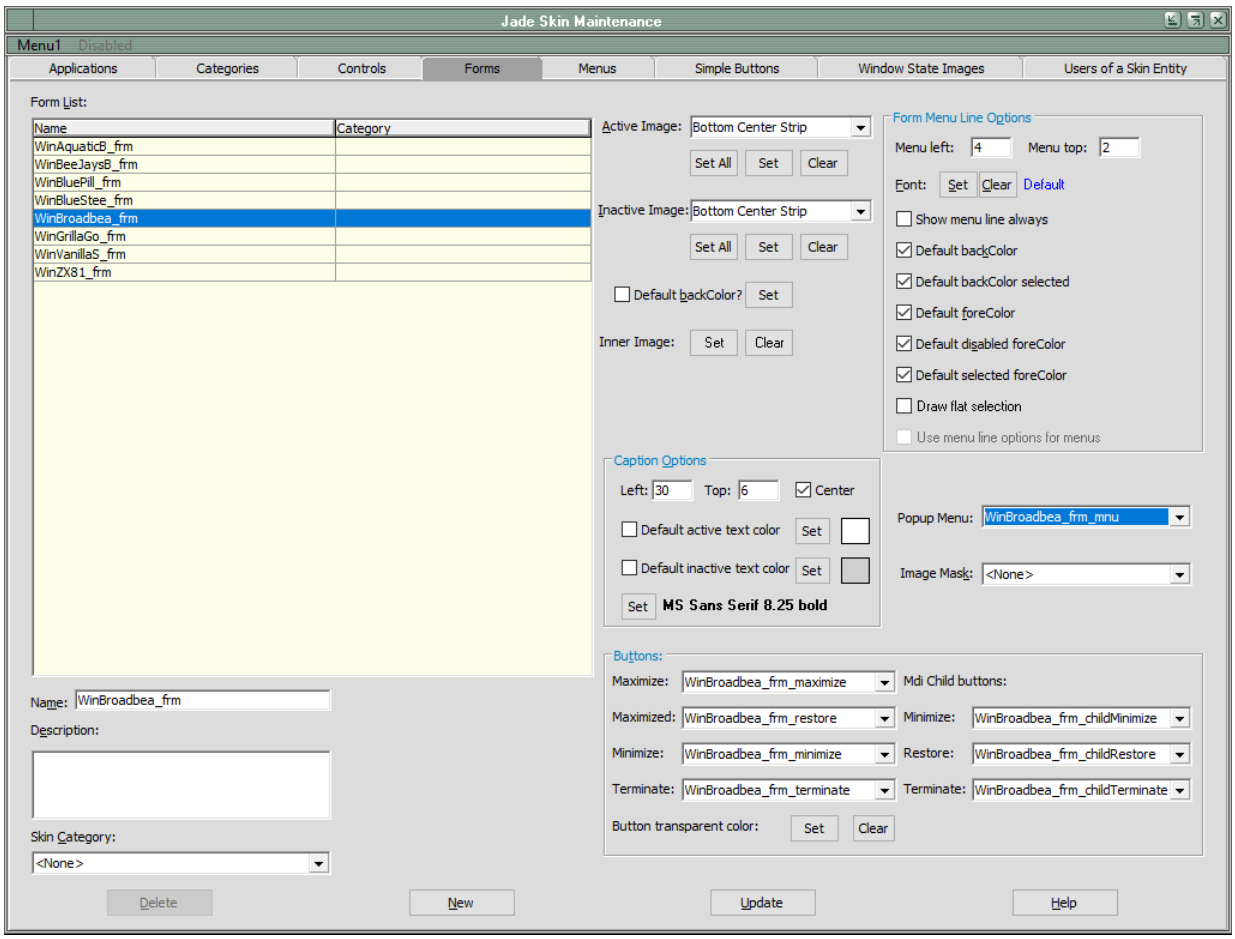
In addition, your selected image is displayed in the text box at the right of the **Highlight Brush** caption.

5. If you want to remove the current highlight brush image, click the **Clear** button at the right of the **Highlight Brush** caption.

   The highlight brush image is then removed from the text box at the right of the **Highlight Brush** caption on the **Controls** form.

Using the Forms Sheet

The **Forms** sheet, shown in the following image, is displayed when you select the **Forms** tab in the Jade Skin Maintenance dialog.



To add a form skin

1. Click the **New** button if another form skin is selected.
2. Enter the name of the form skin in the **Name** text box or select the skin from the **Form List** table.

   Note Form skin names must be unique.

3. Enter a description of the skin in the **Description** text box, if required.
4. In the **Skin Category** combo box, select the skin category that applies to the control skin or select the default **<none>** entry.

   A control skin is applied only to controls of the same type with the same defined skin category.
5. In the **Active Image** and **Inactive Image** combo boxes, you can set all active and inactive form images by clicking the appropriate **Set All** button below the respective combo boxes to request the loading of all pictures that describe the active or inactive form skin.

   A series of common File dialogs is then displayed, to enable you to locate each of the pictures required in turn. The title of the common dialog displays the required image. If you do not have a file for a specific image or you do not want to specify a file for that image, simply click the **Cancel** button to enable you to select the picture file for the next image. If you cancel a common dialog display for any picture, the default image is used for that property.

   When you have specified (or cancelled) the appropriate picture file for each image that you want to define, the common dialog is then closed and focus returns to the **Forms** sheet of the Jade Skin Maintenance dialog. Alternatively, you can perform one of the following actions.

   - Set a specific form skin image by selecting the image type in the combo box (for example, **Right Strip**) and then clicking the **Set** button. The common File dialog is then displayed, to enable you to locate the picture file that you want to define or change.
   - Clear an existing image by selecting the image type in the combo box and then clicking the **Clear** button.
6. To select a background color for the form, check the **Default backColor** check box. The common Color dialog is then displayed, to enable you to select the background color that you require. The background color is used only when an inner image is not set or it is not a brush. If you want to restore the default background color of the control when you have selected a custom color, check the **Default backColor** check box so that a check mark symbol (✓) is displayed and the default color is restored.
7. If you want to set the inner image of the form, click the **Set** button at the right of the **Inner Image** caption. The common File dialog is then displayed, to enable you to locate the picture file that you want to define or change.

   If you set an inner image, the **Is brush?** check box is then displayed at the right of the **Clear** button. By default, this check box is unchecked (that is, the image is centered in the inner area of the form rather than repeatedly drawn over the whole area of the form).

   Alternatively, to clear an existing inner image for the form, click the **Clear** button.
8. Use the text box or check box controls in the Caption Options group box to position the form captions to meet your requirements. If you want to center the caption, check the **Center?** check box. Any value specified in the **Left** text box is ignored unless the caption is too large for the available area. The **Left** and **Top** text boxes indicate the number of pixels to offset the caption start position (for example, **49** and **6**, respectively).
9. To select a color for text on active and inactive captions on your skinned forms, uncheck the **Default active text color** and **Default inactive text color** check boxes in the Caption Options group box.

The common Color dialog is then displayed, to enable you to select the active and inactive text colors that you require.

If you want to restore the default color (specified on the **Appearance** sheet of the Display dialog, accessed from the **Display** icon in the Control Panel for your workstation) of text on active or inactive forms when you have selected a custom color, check the **Default active text color** or **Default inactive text color** check box so that a check mark symbol (✓) is displayed and the default text color restored.

10. Use the **Set** button in the Caption Options or Form Menu Line Options group box to change the font if you do not want Tahoma 8.25 regular font to be used for your form captions or the default font for your menus. The common Font dialog is then displayed, to enable you to select the required font and its attributes.

    User font preferences are not used when displaying the caption and menus, as they may be inappropriate for the skin.

11. Use the text box or check box controls in the Form Menu Line Options group box to position menus on your forms to meet your requirements. The **Menu left** and **Menu top** text boxes indicate the number of pixels to offset the menu start position (for example, **43** and **4**, respectively).

12. Check the **Show menu line always** check box in the Form Menu Line Options group box if you want to specify that both the caption line and the menu line are always displayed on each form, regardless of whether the form has a menu. When you uncheck this check box (that is, it is set to **false**), the menu line is displayed only on forms that have a menu.

13. In the Form Menu Line Options group box, uncheck the following check boxes if you do not want your forms skin to use the:

    - Default **backColor**
    - Default **backColor** selected
    - Default **foreColor**
    - Default disabled **foreColor**
    - Default selected **foreColor**

14. Check the **Draw flat selection** check box if you want a selected menu item always drawn in a flat manner using the defined menu line selection colors (with no surrounding border). In addition, the menu line item is not drawn as though it is part of the popup menu.

    This check box is unchecked by default; that is, the selected menu line item is drawn using the effective menu item selection background and foreground colors with a border that is the same color as the text of the menu item. In addition, when a menu is dropped down directly below menu item (the menu does not need to be moved to fit on the current display), the selected menu item is drawn as though it is part of the dropped-down menu using the same background color and text color defined for the popup menu.

15. When **<None>** is displayed in the **Popup Menu** combo box, the **Use Menu Line Options For Menus** check box is enabled, and unchecked by default. This controls whether the menu line definition of a form skin is used to draw menus when no popup menu is selected (that is, **<None>**). When the **Use Menu Line Options For Menus** check box is unchecked and the **Popup Menu** combo box displays **<None>**, menus are not skinned.

    When this check box is checked and the **Popup Menu** combo box displays **<None>**, menus are drawn using the menu line properties (that is, font, colors, and so on).

16. In the **Popup Menu** combo box, select the drop-down and popup menu skin that you want displayed on your forms, if required. Before you can select a popup menu skin to apply to your forms, you must first have defined the menu skin by using the **Menus** sheet. For details, see "Using the Menus Sheet", in the following subsection.

If a popup menu skin is set for the form, the **Use menu line options for menus** check box is unchecked and disabled.

17. In the **Image Mask** combo box, select a non-rectangular region mask image to be applied to the form skin. To remove an applied mask, select the **<None>** value (the default).
18. In the **Maximize**, **Maximized**, **Minimize**, and **Terminate** combo boxes at the left of the Buttons group box, select the images that you want to apply to your skin for the form's maximize, maximized, minimize, and terminate buttons, respectively.
19. If the form icons have been created with a transparent color surround, click the **Set** button in the Buttons group box. The common Color dialog is then displayed, to enable you to select or define the transparent color that you require. Alternatively, clear an existing button transparent color by clicking the **Clear** button.
20. In the MDI child buttons **Minimize**, **Restore**, and **Terminate** combo boxes at the right of the Buttons group box, select the images that you want to apply to your skin for the minimize, restore, and terminate buttons, respectively, on MDI child forms.
21. Click the **Update** button.

» To update an existing form skin

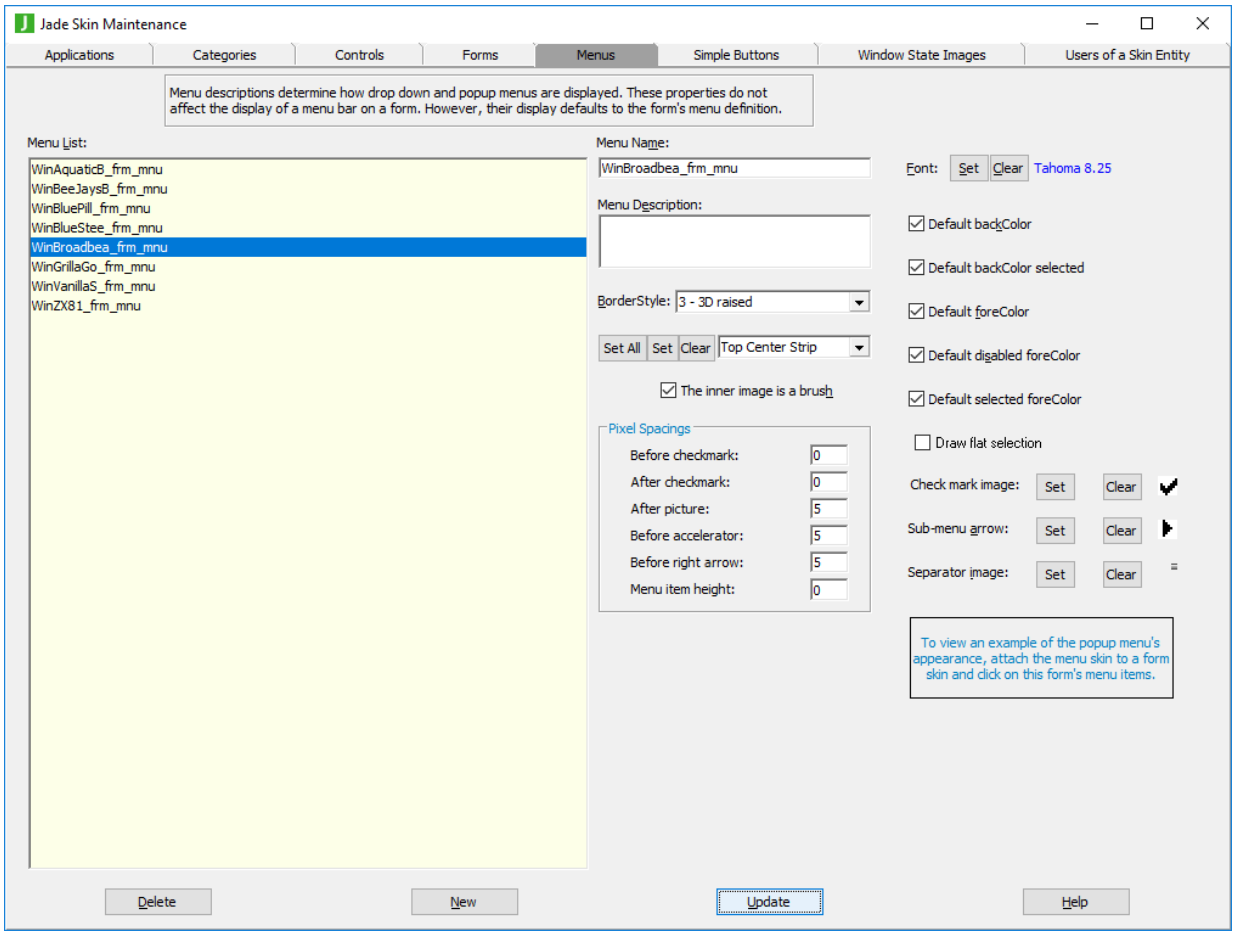
1. In the **Form List** table, select the form skin that you want to change. The name of the selected form skin is then displayed in the **Name** text box.
2. Edit the appropriate data to meet your requirements. (For details, see steps 2 through 16 in the previous instruction for adding a form skin.)
3. Click the **Update** button.

» To delete an existing form skin

1. In the **Form List** table, select the application skin that you want to delete. The name of the selected form skin is then displayed in the **Name** text box.
2. Click the **Delete** button. The **Delete** button is disabled if the form skin is referenced by any other application skin entity.
3. A message box advises you that the skin may be referenced externally and prompts you to confirm that the skin can be safely deleted.
4. Click the **Yes** button to confirm the deletion. Alternatively, click the **No** button to abandon the deletion.

Using the Menus Sheet

The **Menus** sheet, shown in the following image, is displayed when you select the **Menus** tab in the Jade Skin Maintenance dialog.



To add a menu skin

1. Click the **New** button if another menu is selected.
2. Enter the name of the menu skin in the **Menu Name** text box or select the skin from the **Menu List** list box.

   Note Menu names must be unique.

3. Enter a description of the menu in the **Menu Description** text box, if required.
4. In the **BorderStyle** combo box, select the border style that you require for your menus if you do not want the default three-dimensional border style used.
5. If the border style is set to **4 – use border images** in the **BorderStyle** combo box, you can set these images by clicking the **Set All** button that requests the loading of all pictures that describe the skin.

A series of common File dialogs is then displayed, to enable you to locate each of the pictures required in turn. The dialog title displays the required image. If you do not have a file for a specific image or you do not want to specify a file for that image, simply click the **Cancel** button to enable you to select the picture file for the next image. If you cancel a common dialog display for any picture, the default image is used for that property.

When you have specified (or cancelled) the appropriate picture file for each image that you want to define, the common dialog is then closed and focus returns to the **Menus** sheet of the Jade Skin Maintenance dialog. Alternatively, you can perform one of the following actions.

- Set a specific menu skin image by selecting the image type in the combo box (for example, **Bottom Center Strip**) and then clicking the **Set** button. The common File dialog is then displayed, to enable you to locate the picture file that you want to define or change.
- Clear an existing image by selecting the image type in the combo box and then clicking the **Clear** button.

6. Use the **The inner image is a brush?** check box to specify whether the inner image is treated a brush to be repeatedly drawn over the whole area of the menu or whether it is an image that is drawn centered in the inner area of the menu. (By default, the inner image is a brush.)
7. Specify the required number of pixels that you require for spacing in the following text boxes in the Pixel Spacings group box.
   - Before checkmark (the default value is **0**)
   - After checkmark (the default value is **0**)
   - After picture (the default value is **5**)
   - After accelerator (the default value is **5**)
   - Before right arrow (the default value is **5**)
   - Menu item height (the default value is **0**)

   If none of the displayed menu items has a specific entity (for example, a check mark or a picture), the appropriate column for that entity and the specified number of pixels are ignored.
8. If you do not want the default Windows font to be used for your menu skin, click the font **Set** button. The common Font dialog is then displayed, to enable you to select the required font and its attributes.

   Alternatively, click the font **Clear** button to restore the default font of the form.
9. Uncheck the following check boxes at the right of the sheet if you do not want your menu skin to use the:
   - Default backColor
   - Default backColor selected
   - Default foreColor
   - Default disabled foreColor
   - Default selected foreColor
10. Check the **Draw flat selection** check box if you want a selected menu item always drawn in a flat manner using the defined menu line selection colors (with no surrounding border). In addition, the menu line item is not drawn as though it is part of the popup menu.

This check box is unchecked by default; that is, the selected menu line item is drawn using the effective menu item selection background and foreground colors with a border that is the same color as the text of the menu item. In addition, when a menu is dropped down directly below menu item (the menu does not need to be moved to fit on the current display), the selected menu item is drawn as though it is part of the dropped-down menu using the same background color and text color defined for the popup menu.

11. If you want to select the image that is displayed for the check mark, the arrow that indicates a submenu, or the menu separator, click the appropriate **Set** button.

    The common File dialog is then displayed, to enable you to select the existing image that you want displayed for the check mark, submenu arrow, or separator. To apply the image, click the **Open** button in the common dialog.

    When you have selected the appropriate picture file for the image, the common dialog is then closed and focus returns to the **Menus** sheet of the Jade Skin Maintenance dialog. In addition, your selected image is displayed at the right of the **Clear** button for that entity.

12. If you want to remove the current check mark, submenu arrow, or separator image, click the **Clear** button at the right of the appropriate entity. The image is then removed from the right of the **Clear** button on the **Menus** form.

13. Click the **Update** button.

Notes To try out a menu skin, define the menu skin as one that is used by a form skin and then click on the menu item displayed for this form when the **Forms** sheet has focus.

If the menu does not fit on the menu line, the menu is extended to include additional lines as required. Each line is drawn with the same skin images as the first menu line.

To update an existing menu skin

1. In the **Menu List** list box, select the menu skin that you want to change. The name of the selected skin is then displayed in the **Menu Name** text box.
2. Edit the appropriate data to meet your requirements. (For details, see steps b through d in the previous action.)
3. Click the **Update** button.

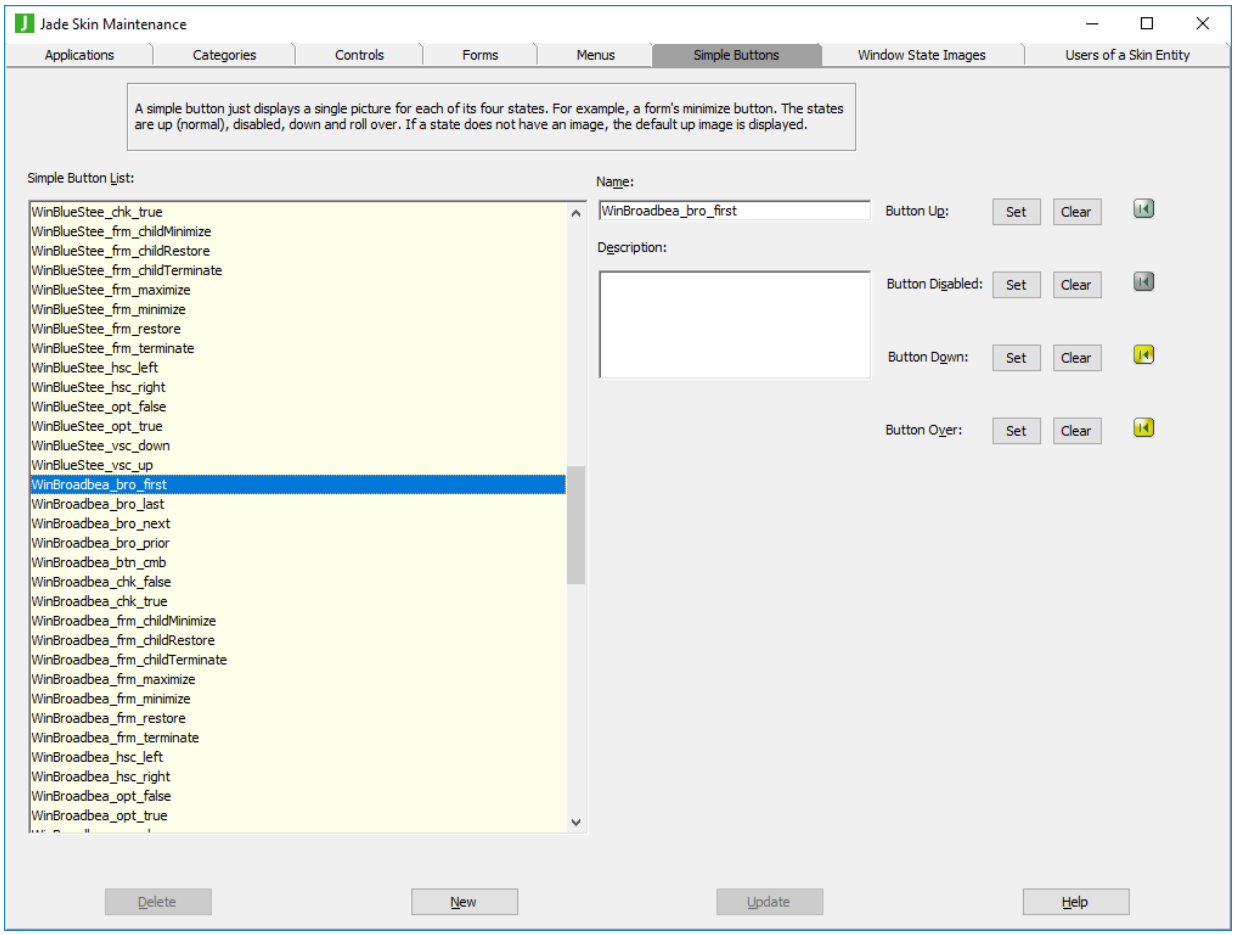
To delete an existing menu skin that is not currently referenced

1. In the **Menu List** list box, select the menu skin that you want to delete. The name of the selected skin is then displayed in the **Menu Name** text box.
2. Click the **Delete** button.

The **Delete** button is disabled if the menu skin is referenced by any form skin entry.

Using the Simple Buttons Sheet

The **Simple Buttons** sheet, shown in the following image, is displayed when you select the **Simple Buttons** tab in the Jade Skin Maintenance dialog.



Note Before you can specify the first, prior, next, and last buttons for your browse buttons skins or the **true** and **false** buttons for your check box and option button skins, you must first have defined those buttons by using the **Simple Buttons** sheet.

To add or maintain a simple button for browse buttons, options buttons, or check box skins

- To add a new simple button skin:

  a. Click the **New** button if another simple button is selected.

  b. Enter the name of the simple button skin in the **Name** text box or select the skin from the **Simple Button List** list box.

  Note Simple button names must be unique.

  c. Enter a description of the simple button in the **Description** text box, if required.

  d. If you want to select the image that is displayed for the up, disabled, down, or over state of simple

buttons, click the appropriate **Set** button at the right of the required caption.

The common File dialog is then displayed, to enable you to select the existing image that you want displayed for the simple button state.

To apply the image, click the **Open** button in the common dialog. When you have selected the appropriate picture file for the image, the common dialog is then closed and focus returns to the **Controls** sheet of the Jade Skin Maintenance dialog.

In addition, your selected image is displayed in the text box at the right of the appropriate caption.

e. If you want to remove the current simple button state picture image, click the **Clear** button at the right of the appropriate caption.

The displayed image is then removed from the text box at the right of the caption on the **Simple Buttons** form.

f. Click the **Update** button.

- To update an existing simple button skin:

  a. In the **Simple Button List** list box, select the simple button skin that you want to change. The name of the selected skin is then displayed in the **Name** text box.

  b. Edit the appropriate data to meet your requirements.

  For details, see steps b through e in the previous action.

  c. Click the **Update** button.

- To delete an existing simple button skin that is not currently referenced:

  a. In the **Simple Button List** list box, select the simple button skin that you want to delete.

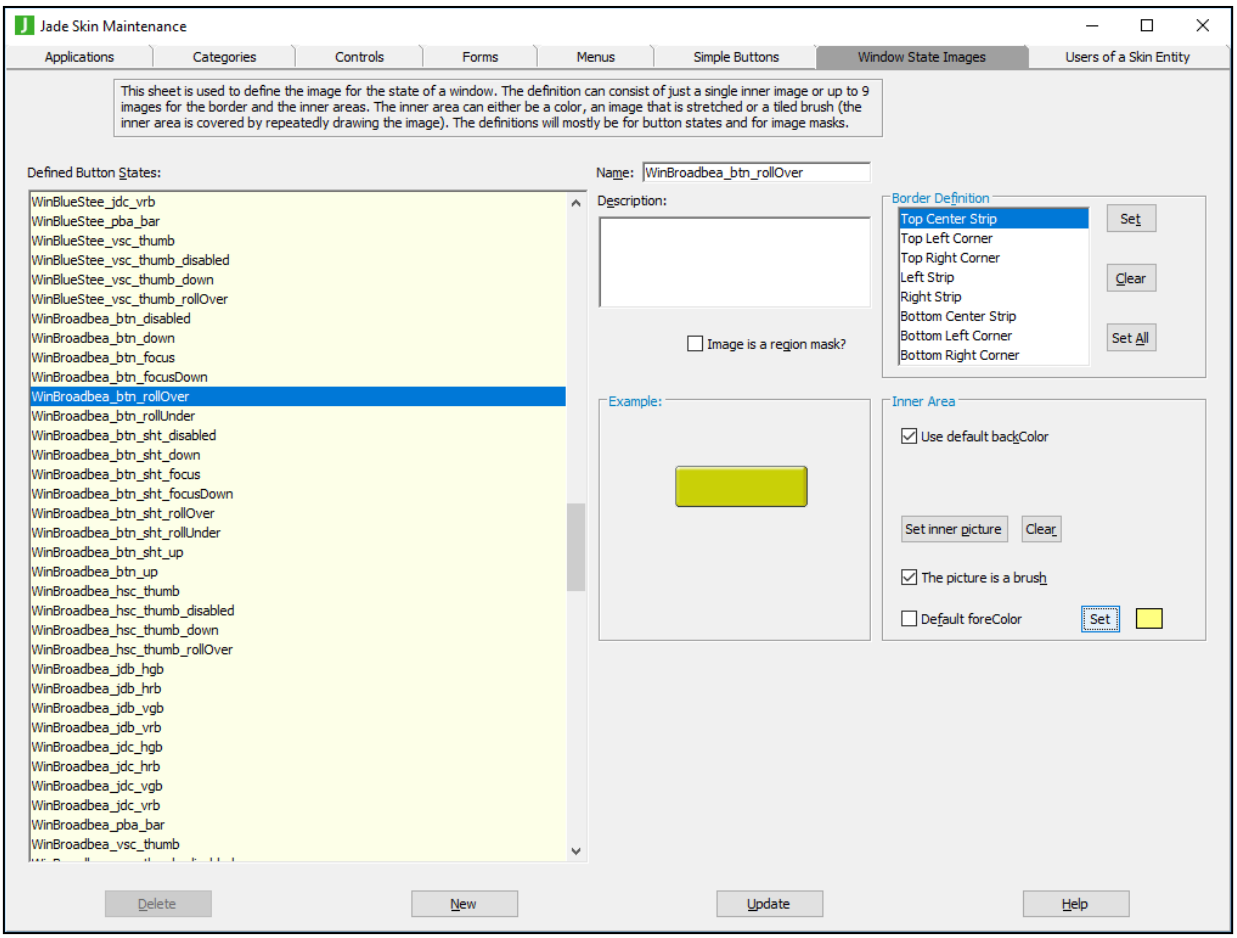
  The name of the selected skin is then displayed in the **Name** text box.

  b. Click the **Delete** button.

  The **Delete** button is disabled if the simple button is referenced by any skin entity.

Using the Window State Images Sheet

The **Window State Image** sheet, shown in the following image, is displayed when you select the **Window State Image** tab in the Jade Skin Maintenance dialog.



Note Before you can specify the image states for button or scroll bar controls thumb tracks, you must first have defined those image states by using the **Window State Images** sheet.

To add or maintain a window state image for button skins

- To add a new window state image:

  a. Click the **New** button if another button state image is selected.

  b. Enter the name of the state image in the **Name** text box or select the state image from the **Defined Button States** list box.

  Note Window state image names must be unique.

  c. Enter a description of the button state image in the **Description** text box, if required.

  d. Check the **Image is a region mask?** check box if you want to specify that the button state image is a region mask. When this check box is checked, the full rectangular image is not drawn using the skin and the image mask applies only to any mouse actions. For example, if a button is an unusual shaped

image on a background, the button then displays only the rollover and click images when the mouse is over that special area.

e. Click the **Set All** button in the Border Definition group box if you want to select the border state image for each area of a skin, to request the loading of all pictures that describe the skin border area.

A series of common File dialogs is then displayed, to enable you to locate each of the pictures required in turn. The title of the common dialog displays the required image. If you do not have a file for a specific image or you do not want to specify a file for that image, simply click the **Cancel** button to enable you to select the picture file for the next image. If you cancel a common dialog display for any picture, the default image is used for that property.

When you have specified (or cancelled) the appropriate picture file for each border image that you want to define, the common dialog is then closed and focus returns to the **Window State Images** sheet of the Jade Skin Maintenance dialog. Alternatively, you can perform one of the following actions.

- Set a specific border state image by selecting the border area in the **Border Definition** list box (for example, **Right Strip**) and then clicking the **Set** button. The common File dialog is then displayed, to enable you to locate the picture file that you want to define or change.
- Clear an existing state image by selecting the border area in the **Border Definition** list box and then clicking the **Clear** button.

f. Uncheck the **Use Default backColor?** check box in the Inner Area group box if you do not want the default background color used when drawing the state image.

The common Color dialog is then displayed, to enable you to select the background color that you require. The background color is used only when an inner image is not set or it is not a brush.

If you want to restore the default background color of the inner area of the state image when you have selected a custom color, check the **Use Default backColor?** check box so that a check mark symbol (✓) is displayed and the default color is restored.

g. Click the **Set Inner Picture** button if you want to set the inner picture image. The common File dialog is then displayed, to enable you to select the existing image that you want displayed for the inner window state image.

To apply the image, click the **Open** button in the common dialog. When you have selected the appropriate picture file for the inner image, the common dialog is then closed and focus returns to the **Window State Images** sheet of the Jade Skin Maintenance dialog.

In addition, the **The picture is a brush?** check box is then displayed in the Inner Area group box. This check box is checked by default, indicating that your selected picture is treated a brush to be repeatedly drawn over the whole inner area of the skin.

h. Uncheck the **The picture is a brush?** check box if you want your selected inner image drawn centered in the inner area of the skin.

i. If you want to remove the selected image for the inner area, click the **Clear** button in the Inner Area group box.

The **The picture is a brush?** check box is then no longer displayed.

j. If you want to select or define the foreground color you require for the **rollOver** state of a button skin, uncheck the **Default foreColor** check box.

The common Color dialog is then displayed, to enable you to select or define the foreground color you require for the **rollOver** state of the skin. When you have selected or defined the appropriate foreground color for the rollover state of the button skin, click the **OK** button on the common Color dialog.

The common dialog is then closed and focus returns to the **Window State Images** sheet of the Jade Skin Maintenance dialog, with the **Set** button and the selected foreground color displayed at the right of the **Default foreColor** check box.

k. Click the **Update** button.

- To update an existing window state image:

  a. In the **Defined Button States** list box, select the state that you want to change. The name of the selected state is then displayed in the **Name** text box.

  b. Edit the appropriate data to meet your requirements. For details, see steps b through j in the previous action.

  c. Click the **Update** button.

- To delete an existing window state image that is not currently referenced:

  a. In the **Defined Button States** list box, select the state that you want to delete.

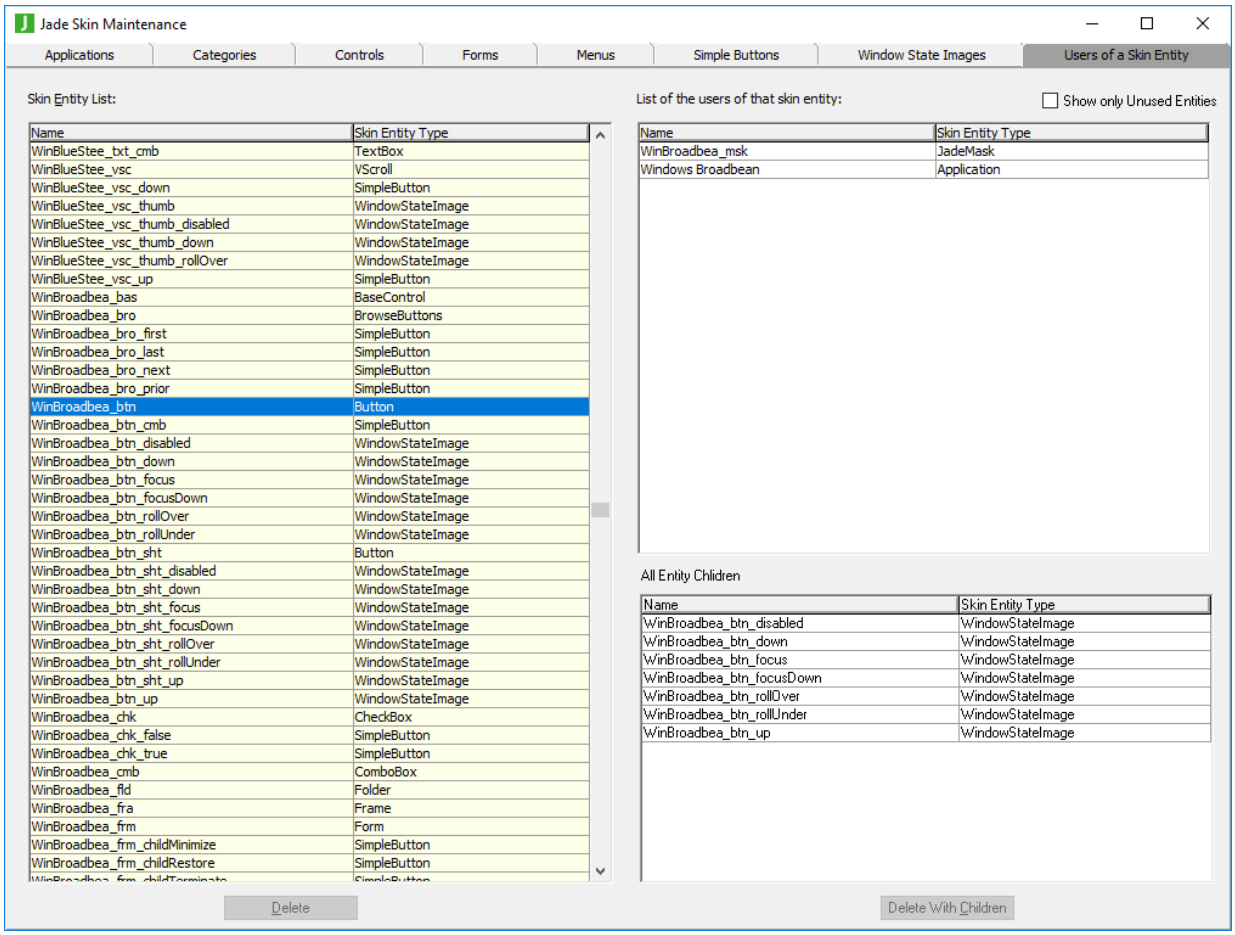
  The name of the selected state is then displayed in the **Name** text box.

  b. Click the **Delete** button.

  The **Delete** button is disabled if the state image is referenced by any skin entity.

Using the Users of a Skin Entity Sheet

The **Users of a Skin Entity** sheet, shown in the following image, is displayed when you select the **Users of a Skin Entity** tab in the Jade Skin Maintenance dialog.



The **Users of a Skin Entity** sheet has a:

- **Skin Entity List** table that lists all defined skin entities. When you select an entry in this table, the **List of the users of that skin entity** table is populated with all skin entities that reference the entity you selected in the **Skin Entity List** table.
- **All Entity Children** table that lists all children that are referenced by the selected skin entity.
- **Show only Unused Entities** check box, which when unchecked, displays all defined skin entities in the **Skin Entity List**.

  When checked, the **Skin Entity List** table contains only skin entities that are not referenced by any other skin entity and the **All Entity Children** table contains only children that are referenced only by the selected skin entity.

  Note The unused **Skin Entity List** also includes application skins, as these are have no skin entity references.

- **Delete** button at the lower left, which is enabled only if the selected skin entity is not referenced by any other

skin element. (You cannot delete a skin entity until there are no skin entities that reference it.)

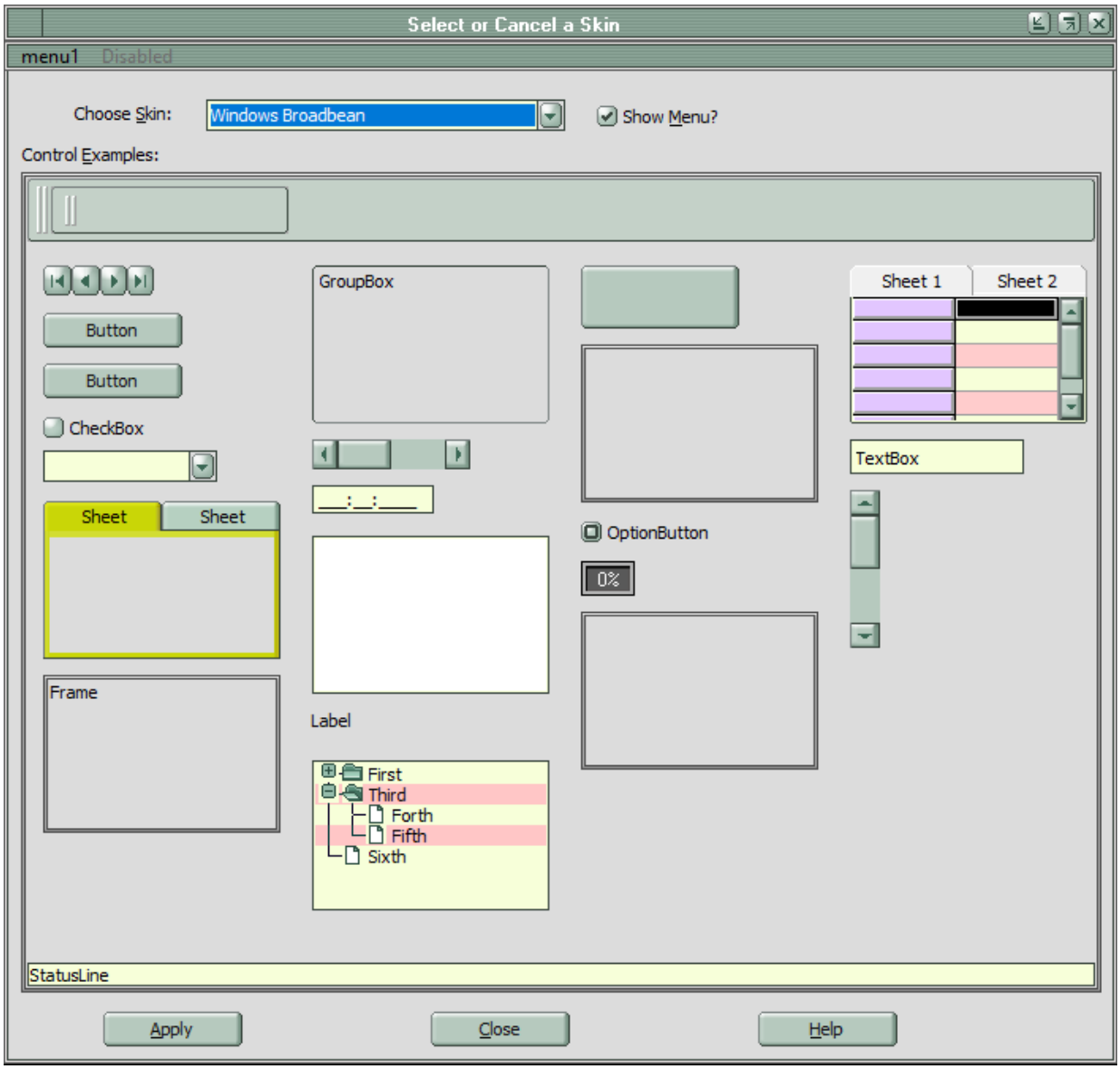
Clicking the enabled button shows a confirmation message box. If you click the **Yes** button, the selected skin entity is deleted (but no referenced children).

- **Delete with Children** button on the lower right, which is enabled only if the selected skin entity is not referenced by any other skin element.

  Clicking the enabled button shows a confirmation message box. If you click the **Yes** button, the selected skin entity is deleted together with all child skin entities that are were referenced only by the selected skin entity or its children.

Selecting a Skin to Use in Runtime Applications

You can enable users of runtime applications to select the skin that they want to use during the current runtime work session, test an application skin, or cancel the use of a skin, by using the Select or Cancel a Skin dialog, shown in the following example.



For details about defining and maintaining skins, see "Maintaining Skins Using Extended Functionality".

Your user application needs only to create and modally show the **JadeSkinSelection** form. The following example shows the creation and use of the **JadeSkinSelection** form from a **click** event method in user application logic.

```
selectSkin_click(menuItem: MenuItem input) updating;
vars
    form : JadeSkinSelection;
begin
    create form;
```

```
    form.showModal;
epilog
    delete form;
end;
```

Alternatively, you can define a **JadeScript** class **selectSkin** method like that shown in the following example.

```
selectSkin() updating;
/*
Date created: 14 November 2016
Created by:   Wilbur
*/
vars
    form : JadeSkinSelection;
begin
    create form;
    form.showModal;
epilog
    delete form;
end;
```

To select a skin for use in the current work session

1. Select the skin from the list that is displayed in the **Choose Skin** combo box.

   As the Select or Cancel a Skin dialog itself is displayed using the skin selected in the **Choose Skin** combo box, the presentation offered by the selected skin shows examples of the controls that are skinned in the Control Examples group box, as shown in the previous image.

2. Click the **Show Menu?** check box to view the effects of a skin without a menu. By default, a check mark symbol (✓) is displayed in this control, indicating that the menu line is displayed on the Select or Cancel a Skin dialog.

3. To apply the selected skin to all forms in the current application during the current work session, click the **Apply** button.

   Alternatively, perform one of the following actions.

   - To cancel the use of a skin in the current work session, select the **<None>** value in **Choose Skin** combo box and then click the **Apply** button.
   - Click the **Close** button to abandon your selections and leave the skin usage unchanged.

Programmatically, if the user:

- Selects a skin, **modalResult = JadeSkinSelection.Skin_Changed(1)** and the **userObject** property contains the selected skin.
- Clicks the **Close** button, **modalResult = JadeSkinSelection.Skin_Cancel(0)**.
- Selects the **<None>** value in **Choose Skin** combo box and then clicks the **Apply** button, **modalResult = JadeSkinSelection.Skin_Changed(1)**.

Use the **app.getApplicationSkin** method to retrieve the currently selected skin.

Chapter 3 Using Rich Text Controls on Runtime Forms

This chapter covers the following topics.

Overview

Runtime forms that contain **JadeRichText** controls enable users to input and display rich text. Rich text controls support a wide variety of formatting (for example, bulleting, fonts, and tables) and the insertion of other objects (for example, bitmaps or Word documents).

Notes Like all third-party ActiveX controls for rich text, extensive use of the **JadeRichText** control to store large rich text documents in the JADE database may significantly increase a user's disk requirements.

If the application developer set up text that is initially displayed in a rich text control when a specific button is clicked, it is displayed in rich text format in the control.

A context (popup) menu that provides access to edit operations, character formatting, and basic paragraph formatting (for example, setting bulleting, fonts, and indents) may also be available in the control. The actions provided on this menu, determined by the application developer, are described in the following subsections. (Use the Esc key to cancel the display of popup menus.) Context menu options that are not available are automatically disabled (grayed out). Menu items that are followed by the points of ellipsis symbol (…) access a dialog relevant to that option.

Rich Text Control Shortcut Keys

You can use the shortcut keys to quickly perform actions in a **JadeRichText** control at run time.

Copying, Moving, or Deleting Text or Graphics

The shortcut keys listed in the following table enable you to copy, move, or delete text and graphics by using the keyboard.

Key	Action
Ctrl+C	Copies text or graphics to the clipboard
Ctrl+X	Cuts selected text to the clipboard
Ctrl+V	Pastes the clipboard contents, starting at the insertion point
Backspace	Deletes one character to the left of the insertion point
Ctrl+Backspace	Deletes one word to the left of the insertion point
DEL	Deletes one character to the right of the insertion point
Ctrl+DEL	Deletes one word to the right of the insertion point
Ctrl+Y	Redoes, or reapplies, the last editor action
Ctrl+Z	Undoes the last action

Moving the Insertion Point

The shortcut keys listed in the following table enable you to move the insertion point in a rich text control by using the keyboard. (You can use the Shift key in conjunction with the shortcuts listed in the table to select text or graphics in the rich text control; for example, Ctrl+Shift+→ selects the text from the insertion point to the beginning of the next word.)

Key	Moves the insertion point …
←	One character to the left
→	One character to the right
Ctrl+←	To the beginning of the word (delimited by white space and an alphanumeric or a non-alphanumeric character) on the left
Ctrl+→	To the beginning of the word (delimited by white space and an alphanumeric or a non-alphanumeric character) on the right
↑	Up one line
↓	Down one line
Ctrl+↑	Up one paragraph
Ctrl+↓	Down one paragraph
Tab	One cell to the right of a table
Shift+Tab	One cell to the left of a table
Page Up	Up one screen (scrolling)
Page Down	Down one screen (scrolling)
Ctrl+Page Up	To the top of the next page
Ctrl+Page Down	To the bottom of the next page

Key	Moves the insertion point …
Alt+Ctrl+Page Up	To the top of the window
Alt+Ctrl+Page Down	To the bottom of the window
Home	To the start of the current line
End	To the end of the current line
Ctrl+Home	To the start of the document
Ctrl+End	To the end of the document

Formatting Text

The shortcut keys listed in the following table enable you to format text by using the keyboard.

Key	Toggles …
Ctrl+B	Bold formatting on or off
Ctrl+I	Italics formatting on or off
Ctrl+U	Underline formatting on or off

Formatting a Paragraph

The shortcut keys listed in the following table enable you to perform formatting in paragraphs by using the keyboard.

Key	Action
Ctrl+E	Centers a paragraph
Ctrl+J	Justifies a paragraph
Ctrl+L	Left-aligns a paragraph
Ctrl+R	Right-aligns a paragraph
Shift+Ctrl+L	Cycles through bullet options

Moving around a Table

The shortcut keys listed in the following table enable you to move around a table by using the keyboard.

Key	Action
Tab	Moves the insertion point to the next cell in the current row
Shift+Tab	Moves the insertion point to the previous cell in the current row
↑	Moves the insertion point to the previous row
↓	Moves the insertion point to the next row
Enter	Inserts a new row
Delete	Deletes the currently selected row

Formatting Text for a Specific Language

The shortcut keys listed in the following table enable you to format text for a specific language by using the keyboard. After pressing one of these shortcut keys, press the appropriate letter (for example, **a**, **e**, or **u**).

Key	Displays the…
Ctrl+' (apostrophe)	Accent acute
Ctrl+` (grave)	Accent grave
Ctrl+: (colon)	Accent umlaut
Ctrl+Shift+6	Accent caret (circumflex)
Ctrl+, (comma)	Accent cedilla
Ctrl+Alt+E	Euro symbol (€)

These shortcut keys apply only to English, French, German, Italian, and Spanish keyboards.

Formatting and Selecting Text

Unlike other controls whose font and its attributes apply to the whole control, each character in a rich text control can have its own font and attributes because the font is an attribute of the actual text.

Although the font or fonts that are initially displayed are specified when the form is designed during the application development, users can change the font and attributes of selected text if the context menu enables them to do so. For details, see "Formatting Selected Characters" and "Formatting Paragraphs".

Users can apply formatting attributes to both characters and paragraphs, by selecting text using the mouse or the keyboard. The *current selection* is the range of selected characters or the position of the insertion point if no characters are selected.

You can perform the mouse actions listed in the following table in rich text controls.

Action	Result
Double-click	Selects a whole word
Triple-click	Selects a whole paragraph
Click in the selection bar (the narrow strip down the left side)	Selects the whole line

In addition, you can select text or graphics by clicking the mouse at the start of the selection and then dragging the mouse to the end of the text or graphics required for selection while the key is depressed.

Tip Double-click on a word to select a single word, and then right-click to access the context menu that provides you with editing and formatting operations (for example, you can apply font attributes, align a paragraph, or apply a bullet to a paragraph). Alternatively, select several words or a paragraph of text and apply your formatting to your selection.

Formatting Selected Characters

The character formatting of the insertion point is applied to newly inserted text if the current selection is empty. When the selection changes, the default formatting changes to match the first character in the new selection.

» To change the font of selected text or from the insertion point

1. Right-click in the control. The context menu is then displayed.
2. Select the **Font** command in the context menu (if it is visible or enabled). The common Font dialog is then displayed.
3. Set the appropriate font and its attributes for the selected text or from the insertion point.
4. Click the **OK** button when you have selected the required font and attributes. Alternatively, click the **Cancel** button to abandon your selections.

Formatting Paragraphs

Users can apply alignment, tabs, indents, and numbering formatting attributes to paragraphs if the context menu enables them to do so.

The *current* paragraph is the paragraph that contains the insertion point.

» To change the formatting of the current paragraph or the selected paragraphs

1. Right-click in the control. The context menu is then displayed.
2. Select the **Paragraph** command in the context menu (if it is visible or enabled). The Paragraph dialog is then displayed.
3. In the **Left** text box, specify the distance (in pixels) between the left edge of the control and the left edge of the current paragraph. The default value of zero (**0**) indicates that the paragraph is not indented.
4. In the **Right** text box, specify the distance (in pixels) between the right edge of the control and the right edge of the current paragraph.

   The default value of zero (**0**) indicates that the paragraph is not indented.
5. In the **First line** text box, specify the distance (in pixels) between the left edge of the first line of text in the current paragraph and the left edge of subsequent lines in the same paragraph. (This value can be negative.)

   The default value of zero (**0**) indicates that the first line of text is not indented.
6. In the **Alignment** combo box specify or select the alignment of the current paragraph.

   The alignment can be one of the following values.

Value	Paragraph…
Left	Aligns to the left with text on the right (the default)
Right	Aligns to the right with text on the left
Center	Is centered in the control
Justify	Is aligned relative to the left and right margins

7. Click the **OK** button when you have selected the required paragraph attributes. Alternatively, click the **Cancel** button to abandon your selections.

Applying a Bullet to a Paragraph

When the context menu for the control has been designed so that users can apply a bullet to a paragraph of rich text on a run time form, users can select the type of bullet that they want to apply to the current paragraph.

» To apply a bullet

1. Right-click in the paragraph to which you want to apply a bullet. The context menu is then displayed.
2. Select the **Bullet Style** command in the context menu (if it is visible).
3. From the submenu that is then displayed, select the style of bullet that you want to apply. You can apply one of the following bullet styles.
   - **None** (the default value)
   - **Dot** (a solid circle; that is, (•)
   - **Number** (starting at **1)** and automatically incrementing for each paragraph that is inserted after the current paragraph)
   - **Lowercase Letter** (starting at **a)** and automatically incrementing for each paragraph that is inserted after the current paragraph)
   - **Uppercase Letter** (starting at A**)** and automatically incrementing for each paragraph that is inserted after the current paragraph)
   - **Lowercase Roman Numeral** (starting at **i)** and automatically incrementing from **ii)** onwards for each paragraph that is inserted after the current paragraph)
   - **Uppercase Roman Numeral** (starting at **I)** and automatically incrementing from **II)** onwards for each paragraph that is inserted after the current paragraph)

The selected style of bullet is then inserted at the left of the first line of the current paragraph.

Printing a Rich Text Control

Users can specify the margins around a printed page and print the contents of the rich text control or only the selected portion of the control if the context menu enables them to do so, by using the **Page Setup** and **Print** commands.

The common Print Setup dialog or Print dialog that is displayed enables users to specify their requirements for the printed control or selected contents of the control.

URL Detection

When automatic Uniform Resource Locator (URL) detection is enabled for a rich text control, JADE automatically detects a URL when a user enters it into the control (for example, if a user were to enter **www.jadeworld.com**, the color of the text changes to blue and is underlined).

When automatic URL detection is enabled:

- The control scans any modified text to determine whether the text matches the format of a URL.
- The control highlights the URL string by underlining it and setting the text color.
- Double-clicking on a URL activates the link to the appropriate Web page.

Clipboard Operations

JADE rich text controls support standard clipboard operations and multiple levels of undo and redo operations, to a maximum of 100 undo or redo operations.

To perform a clipboard action

1. Right-click in the control. The context menu is then displayed.
2. Select the appropriate command in the context menu (if it is visible or enabled).

   The following clipboard commands may be available and enabled if another action has been performed (for example, you cannot perform a cut or copy action if no text is selected and the **Redo** command is disabled if you have not performed an **Undo** action).

   - Redo
   - Undo
   - Cut
   - Copy
   - Paste

The selected action is then performed.

Finding and Replacing Text

JADE rich text controls enable users to search for specified text and to search for text and replace it with a specified value.

To perform a find or a replace action

1. Right-click in the rich text control. The context menu is then displayed.
2. Select the **Find** command if you want to only to locate specific text or select the **Replace** command if you want to replace located text with another text value.

   The Find dialog or the Find and Replace dialog is then displayed, depending on the menu command that you selected. The Find dialog is a simpler form of the Find and Replace dialog.
3. In the **Find what** text box, specify the text that you want to locate in the control.
4. If you selected the **Replace** command, in the **Replace with** text box specify the text that is to replace the text specified in the **Find what** text box.
5. In the Search group box, select the **Up** option button if you want to search backwards from the current position on the control to the top of the control. The default value of **Down** searches for text from the current position to the end (bottom) of the control.
6. Check the **Match case** check box if you want the exact match by case (where uppercase or lowercase is significant). A search is then performed for text with the same capitalization as the text in the **Find what** text box. By default, searching is case-insensitive; that is, this check box is unchecked.
7. Check the **Whole word only** check box if you want to find only whole words specified in the text in the **Find what** text box. By default, any word or part of a word that contains the specified characters is searched; that is, this check box is unchecked.

8. To find the next occurrence of the specified text, click the **Find Next** button.

   If JADE finds the text string that matches the specified options, the located text is then highlighted. If JADE cannot find the text string that matches your specified options, a message dialog informs you that the search text was not found and waits for you to click the **OK** button in the message dialog before returning focus to the control.

9. If you selected the **Replace** command and you want to confirm that the located text is replaced with the replacement text, click the **Replace** button and then click the **Find Next** button again to find the next occurrence of the specified text from the text selected in the control, if required.

   Alternatively, click the **Replace All** button if you want to replace all occurrences of the specified search text with the specified replacement text. If JADE finds the text string within the selected text that matches your specified options, all occurrences of the located text are replaced with the specified replacement text.

10. Click on the **OK** button. Alternatively, click the **Cancel** button to abandon your selection.

Scrolling

Rich text controls on runtime forms automatically support scrolling when the developer of the form set the appropriate scroll bar functionality and the text to be displayed exceeds the dimensions of the control.

No horizontal scrolling takes place by default, as the text is wrapped horizontally within the bounds of the control.

Inserting Objects

Users can insert any Component Object Model (COM) object such as an Excel spreadsheet, a bitmap, or a Word document into a rich text control, by either embedding or linking to it.

An embedded object is edited within the rich text control itself and a linked object is edited in the source file location. The COM object can be displayed as a view of the object or as an icon representing the program that runs the inserted object (for example, Paintbrush or Word for Windows).

Double-clicking an inserted object activates its server.

If an object is selected and the context menu is displayed, the verb submenu of the selected object is displayed as a submenu of the **Object** menu item.

» To insert an object into a rich text control

1. Right-click in the position at which you want to insert the object in the control. The context menu is then displayed.

2. Select the **Insert Object** command. The OLE Insert Object dialog is then displayed.

3. Select the **Create New** option button if you want to create a new COM object.

   The **Object Type** drop-down list then displays all COM object types so that you can select the type of object. The appropriate program executable for that object type is then invoked so that you can create the object that you want to insert (for example, Paintbrush if you selected **Bitmap Image** or Microsoft Excel if you selected **Microsoft Excel Chart**).

4. If you want to insert an existing object (the default), in the **File** text box specify the name of the existing file containing the object to be inserted. If you do not specify a path, JADE looks in the current directory and raises an exception if the specified file is not in the directory or if it does not exist.

If you are unsure of the directory that contains the object, click the adjacent **Browse** button to display the common Browse for Folder selection dialog that enables you to select the path in which your object file is located.

5. Check the **Link** check box if you are inserting an existing object and you want to link to the object in the appropriate file (for example, when editing the object and so that changes to that object are reflected in your rich text control) rather than embedding it within the control itself.

   As inserted objects are embedded by default, this check box is unchecked and it applies only to existing objects.

6. Check the **Display As Icon** check box if you want the object inserted as an icon so that the contents of the object are displayed in the control only when you double-click on the icon to activate the program that created it.

   As the contents of an inserted object are displayed within the control rather than displaying an icon for the object type, this check box is unchecked.

7. Click the **OK** button. Alternatively, click the **Cancel** button to abandon your selections.

The selected object is then displayed at the current caret position (the insertion point) in the control, as the full contents of the object or as an icon representing the object.

Note To change the view of the object (that is, display it in the control as an icon view or the full contents view), right-click on the object in the rich text control and then select the **Object Properties** command on the context menu.

The OLE Properties dialog for that type of object then enables you to change the way in which the object is inserted.

Inserting Tables

Users can insert a table into a rich text control. When using tables in rich text controls, note the following.

- Users can insert and delete a whole table or table rows by using the keyboard but they cannot insert or delete columns.
- The display of a table inside a rich text control is limited and table dimensions cannot be edited.
- As text does not wrap within a cell, text can overflow into neighboring cells.

» To insert a table at the current position

1. Right-click in the position at which you want to insert the table in the control. The context menu is then displayed.
2. Select the **Insert Table** command. The Insert Table dialog is then displayed.
3. In the **Number of Rows** text box, specify the number of rows in the table.
4. In the **Number of Columns** text box, specify the number of columns in the table.
5. In the **Left Margin** text box, specify the distance (in pixels) between the left edge of the control and the left edge of the table. The default value of zero (**0**) indicates that the table is not indented.
6. In the **Column Width** text box, specify the width of all columns (in pixels).
7. Click the **OK** button. Alternatively, click the **Cancel** button to abandon your selections.

The table that meets your specified values is then displayed in the rich text control and you can enter values into the cells to meet your requirements. (However, remember that as text does not wrap within a cell, text can overflow into neighboring cells.)

Protecting Text

You can mark a selection of content in a **JadeRichText** control as being protected. Protected text cannot be changed by users.

The **setTextProtection** method enables you to mark a content of a **JadeRichText** control from the position specified in the **start** parameter through to the length specified in the **length** parameter as protected or unprotected, by setting the **protected** parameter to **true** or **false**, respectively.

The **getTextProtection** method returns the protection state of text from the position specified in the **start** parameter through to the length specified in the **length** parameter. The return value indicates if the text is protected, unprotected, or contains a mixture of protected and unprotected text. The return values are represented by the **JadeRichText** class constants listed in the following table.

Constant	Integer Value	Description
TextProtection_Set	1	All text in the specified range is protected
TextProtection_NotSet	0	No text in the specified range is protected
TextProtection_Mixed	#80000000	Mixture of protected and unprotected text

The **protected** event method is raised whenever a user attempts to alter in any way (for example, to insert, delete, or change the size of the font) text that is marked as protected. Your logic can determine if changes the rich text control should be allowed. Set the value of the **allowChange** parameter in the **protected** event method to **true** to allow the changes or to **false** if changes cannot be made to text in the control. The values of the **start** and **length** parameters indicate the block of text that the user is attempting to change. If you do not implement the **protected** event method on a specific control, any attempt to alter protected text is ignored.

In JADE, you can set or clear text protection only via logic (that is, by calling the **setTextProtection** method). Protected text in rich text format created externally to JADE and loaded into the **JadeRichText** control is honored by JADE.

When you call the **setTextProtection** method with the **protected** parameter set to **true** on protected text, the **protected** event is not raised.

Chapter 4 Converting a User Database

This chapter covers the following topics.

Overview

You can convert a JADE user database in multiuser mode to copy the data from the database server and write it to a local directory. This feature enables you to:

- Upgrade a user database from an earlier release (for example, from a JADE 2016 release to JADE 2018)
- Convert a database from one operating system or hardware platform to another
- Convert an ANSI databases containing multiple-byte characters to Unicode

You can run multiple copies of the file convert worker application inside the client node if you want multiple database file conversions to occur concurrently.

The user database conversion creates a **copyfile.log** file in the **logs** directory, which reports when processing a database file (by number) starts and when the file is converted.

For details about running the **jadclient** program to execute a non-GUI application within your JADE code, see "Running a Non-GUI Client Application using jadclient", in Chapter 1.

Notes Converting a database requires the JADE **bin** and empty **system** directories be installed on the target platform and be at the same patch level as the source platform.

Converting an ANSI database to an empty Unicode database using the **jadclient** program with the **userData** and **userSchema** arguments set to **true** avoids exceptions that can occur if you use the JADE Load utility (**jadload** or **jadloadb**) to load schemas into a Unicode JADE system.

If you are manually loading user-defined schemas (that is, the value of the **userSchema** argument is **false**), schemas loaded into the target system must just have been extracted from the source database, to ensure that the target database control file has the identical number for each database file. Extract all schemas from the source database, load them into a new target database (that is, they are the first set of schemas loaded into the target database), and then perform the convert operation.

As patch history information is stored in compressed binary format, converting a system from ANSI to Unicode does not convert this information. This information will not be valid when using it in a Unicode system unless you first extract the patch history from the ANSI system and load it into the Unicode system using the **Replace** option (that is, you must perform the patch history extract action *before* converting the ANSI database and then perform the load action in the converted Unicode database). For details, see "Extracting and Loading a Patch History", in Chapter 3 of the *JADE Development Environment Administration Guide*.

The **JadeConvertDb** application converts up to 1,000 files. If more than 1,000 files are being converted, an error message is output and the application will not convert any files.

To convert a user database, specify the following arguments in the **jadclient** program.

```
jadclient path=database-path
          schema=RootSchema
          app=JadeConvertDb
          ini=jade-initialization-file
          startAppParameters
          defaultPath=default-destination-map-file-directory
         [codepage=codepage-identifier]
         [copies=number-of-multiple-parallel-file-copies]
         [copySingleFileJadeBytes=true|false]
         [excludeuserfiles=map-file-list]
         [userPath=map-file-directory
         [mapFile]
         [overwrite=true|false]
         [rebuildDicts=true|false]
         [userData=true|false]
         [userSchema=true|false]
```

For details, see "Database Conversion Arguments" in the following section.

Caution When rebuilding dictionaries (that is, the **rebuildDicts** argument is set to **true**), the **JadeConvertDb** application must be run from a user-defined schema only.

When you specify an argument after the **startAppParameters** argument in the **jadclient** command line for the **JadeConvertDb** application, there cannot be white space on either side of the equals sign (for example, **rebuildDicts = true** will not work).

If an argument value is longer than 2,048 characters, an exception is raised and logged (including the argument text).

Database Conversion Arguments

This section contains the arguments required to run the **JadeConvertDb** application.

codepage Argument

The optional **codepage** argument enables you to specify the appropriate numeric codepage identifier value (for example, **936** for the Chinese PRC codepage) if you want to convert ANSI database containing multiple-byte characters to Unicode.

For more details, see "Converting a Multiple-Byte Character User Database from ANSI to Unicode".

copies Argument

The optional **copies** argument allows multiple parallel map file conversions to occur simultaneously by specifying how many worker applications are started in the **jadclient** application.

This argument, which accepts a small positive number, defaults to **1**.

copySingleFileJadeBytes Argument

The optional **copySIngleFileJadeBytes** argument specifies whether single file **JadeBytes** instances are copied during the database conversion.

The default value is **true**; that is, single file instances are copied.

If you want to copy these files manually, set the argument to **false**; for example, if the total size of these files is large, you may want to copy them manually.

defaultPath Argument

The **defaultPath** argument specifies the default destination directory for the system and user schema map files.

This argument must be specified, and it must define an existing directory. All converted user map files are created in this directory by default, if you do not specify the **userPath** argument.

excludeuserfiles Argument

The optional **excludeuserfiles** argument enables you to exclude specific user map files from the database conversion process.

The *map-file-list* variable is a semicolon-delimited list of the map file names to exclude from the database conversion; for example:

```
excludeuserfiles=map1;map2
```

If the value of the **userData** argument is **true** and there is no specified **excludeuserfiles** argument or arguments, all user map files are converted.

mapFile Argument

The optional **mapFile** argument enables you to specify the name or names of user map files. Map file arguments are space-separated.

The specified map file or map files are converted in the directory specified by the last of the **defaultPath** or **userPath** argument (that is, the one that is specified last immediately prior to this argument in the **jadclient** command).

As all user map files are converted if the value of the **userData** argument is **true**, specify this argument only when the value of the **userData** argument is **false**.

For an example of a single-threaded copy of a command prompt that converts five user database map files, splitting the output between two directories, see "User Database Conversion Examples".

overwrite Argument

The optional **overwrite** argument, which you can specify once only, indicates whether any existing files in the destination directory are overwritten when converting the user database. This is a simple restart feature at the complete file level.

The default value of **false** indicates that the contents of the destination directory are not overwritten. Specify this argument with a value of **true** if you want to overwrite (replace) an existing map files in the destination directory if they are present in the source database that you are converting.

rebuildDicts Argument

The optional **rebuildDicts** argument, which you can specify once only when in single user mode following the conversion of an ANSI database to a Unicode database, rebuilds the key ordering of collections when the argument is specified and set to **true**.

Notes When you specify the **rebuildDicts** argument, there can be no other arguments after the **startAppParameters** argument and you must run the **jadclient** executable in single user mode.

When the **rebuildDicts** argument is set to **true**, the **JadeConvertDb** application must be run from a user-defined schema only.

Collections that had keys that were multiple-byte sequences in the ANSI database may result in objects being returned in a different order when converted to wide (Unicode) characters.

When you specify **rebuildDicts=true**, the Unicode database is scanned for collections that have **Character** and **String** keys. If the collection has members and if the collection has key values that are non-ASCII (that is, that are greater than 127 decimal), the results are output to the **jommsg.log** file and the **Collection**::**rebuild** method (used by the JADE Logical Certifier utility) is invoked.

Use the **rebuildDicts=false** argument to perform only the checks and logging, without rebuilding.

userData Argument

The optional **userData** argument, which you can specify once only, indicates whether user database map files are converted.

The default value of **true** indicates that *all* user database map files are converted. For example, if you were to specify the following, the value of the **userData** argument has precedence, so all user map files are converted, regardless of any user map files specified in the **mapFile** argument.

```
jadclient path=c:\jade\system ini=c:\jade\test\jade.ini schema=RootSchema
app=JadeConvertDb startAppParameters userData=true map1, map2
defaultPath=/jade/system
```

Conversely, the following example converts all user map files *except* for the **map1** and **map2** files specified in the **excludeuserfiles** argument.

```
jadclient path=c:\jade\system ini=c:\jade\test\jade.ini schema=RootSchema
app=JadeConvertDb startAppParameters userData=true excludeuserfiles=map1,map2
defaultPath=/jade/system
```

When the value of this argument is **false**, in the following example, only the **map1** and **map2** files specified in the **mapFile** argument are converted.

```
jadclient path=c:\jade\system ini=c:\jade\test\jade.ini schema=RootSchema
app=JadeConvertDb startAppParameters userData=false map1, map2
defaultPath=/jade/system
```

userPath Argument

The optional **userPath** argument specifies the map files directory in which all converted map files specified after this argument are created. The specified directory must exist.

As you can specify this argument more than once, you can use it to locate your converted map files in more than one directory, if required.

userSchema Argument

The optional **userSchema** argument, which you can specify once only, indicates whether the **_user*** map files are converted.

The default value of **false** indicates that these files are not converted.

User Database Conversion Examples

When specifying command line arguments, separate the argument and its associated values with a space and enclose any command line arguments that contain spaces in double (**" "**) or single (**' '**) quotation marks. (See also "Running the User Database Conversion Application Example".)

Caution When you specify an argument after the **startAppParameters** argument in the command line for the **JadeConvertDb** application, there cannot be white space on either side of the equals sign (for example, **rebuildDicts = true** will not work).

The following command prompt example converts all database files, including user schemas and user data. This example is a single-threaded copy.

```
jadclient path=c:\jade\system ini=c:\jade\test\jade.ini schema=RootSchema
app=JadeConvertDb startAppParameters defaultPath=/jade/system userSchema=true
```

The following command prompt example converts five user database files, splitting the output between two directories. This example is a single-threaded copy.

```
jadclient path=c:\jade\system ini=c:\jade\test\jade.ini schema=RootSchema
app=JadeConvertDb startAppParameters defaultPath=/jade/system mapfile1 mapFile2
userPath=/jade/system1 mapFile3 userPath=/jade/system2 mapFile4 mapFile5
```

The following command prompt example converts user database files in groups of five in parallel. The control file is to be recreated.

```
jadclient path=c:\jade\system ini=c:\jade\test\jade.ini schema=RootSchema
app=JadeConvertDb startAppParameters defaultPath=/jade/system copies=5
```

Running the User Database Conversion Application Example

The steps listed in this section are examples for the conversion of an ANSI user database to a Unicode user database.

To convert a database

1. At the JADE developer site, perform the following actions.

   a. Install the ANSI or Unicode binaries in the target system at the same patch level as the source platform.

   b. Start a database server on the source platform. Run the JADE Remote Node Access utility, as shown in the following example.

   ```
   jadrap.exe ini=c:\jade\jade.ini path=c:\jade\system
   ```

   c. Start a **jadclient** application on the target (destination) platform, which is where the converted database is to be located, executing the **JadeConvertDb** application in a user-defined schema with the appropriate arguments; for example:

   ```
   jadclient path='c:\jade\system'
             ini=/jade/jade.ini
             schema=user-defined-schema
             app=JadeConvertDb
             startAppParameters
             command-line-arguments
   ```

   The *command-line-arguments* value, which is specified after the **startAppParameters** argument, represents the arguments described under "Database Conversion Arguments" that control the **JadeConvertDb** application.

   For details about running **jadclient** to execute a non-GUI application within your JADE code, see "Running a Non-GUI Client Application using jadclient".

   d. There may be untranslatable characters in the ANSI database, caused by characters having been entered in a different code page. If this occurs, the **copyfile.log** file, created in the **logs** directory, indicates which objects had string conversion problems.

   Note It is your responsibility to check this file, investigate which strings in the object were not translatable, and fix them to meet your requirements.

   e. Re-run the **JadeConvertDb** application to rebuild the key ordering of collections for collections that have key values that are non-ASCII (that is, greater than 127 decimal). This guarantees that referential integrity is maintained for such collections.

   To rebuild the key ordering of collections in the Unicode database following the conversion of an ANSI database to a Unicode database, specify the **rebuildDicts** argument with a value of **true** after the **startAppParameters** argument when running the **jadclient** executable in single user mode, as shown in the following example.

   ```
   unicode/bin/jadclient path=unicode/system
                         ini=unicode/jade.ini
                         schema=user-defined-schema
                         server=singleUser
                         app=JadeConvertDb
   ```

```
startAppParameters
rebuildDicts=true
```

f. In the Unicode system, recompile everything, by extracting all schemas and then reloading them.

g. Run the certify database operation in the JADE Database utility (for details, see "Database Certification", in Chapter 1 of the *JADE Database Administration Guide*), the JADE Logical Certify utility (for details, see Chapter 5, "JADE Logical Certifier Diagnostic Utility", of the *JADE Object Manager Guide*), and then extensively test your JADE system.

2. At your customer site or sites, have the customer perform the following actions.

   a. Take a copy of their production database for conversion testing (referred to as verification database).

   b. Run the certify database operation in the JADE Database utility (for details, see "Database Certification", in Chapter 1 of the *JADE Database Administration Guide*) and the JADE Logical Certify utility (for details, see Chapter 5, "JADE Logical Certifier Diagnostic Utility", of the *JADE Object Manager Guide*) on their ANSI verification database, to ensure a good source database.

   c. Test the conversion on the verification database and send all **certifydb.log**, logical certify log, and **copyfile.log** files back to you at your JADE developer site.

3. At the JADE developer site, perform the following actions.

   a. Review the customer log files, paying special attention to translation problems of user objects in the **copyfile.log** file.

   b. Produce the fix-up scripts (which could be part of the application schemas) that are appropriate for that customer.

   c. Extract all schemas from the Unicode system.

   d. Send all schemas and fix-up scripts to the customer.

4. At your customer site or sites, have the customer perform the following actions.

   a. Re-run the conversion of the ANSI verification database.

   b. Review the **copyfile.log** file.

   c. Load all Unicode schemas and scripts into the customer Unicode verification database.

   d. Run any fix-up scripts, to deal with user objects that had translations problems.

   e. Perform solid testing and verification of the customer Unicode verification database.

   f. Repeat steps 3.a through 4.e of this instruction until you are happy with newly converted Unicode database.

   g. Stop the production ANSI system.

   h. Backup the customer binaries and database.

   i. Convert the customer production ANSI database.

   j. Review the **copyfile.log** file for any differences from prior test conversions.

   k. Load all Unicode schemas and fix-up scripts into new customer Unicode production database.

   l. Run any fix-up scripts, to deal with user objects that had translations problems.

m. Perform sanity testing of the converted customer Unicode database.

n. Restart the new Unicode production system.

Converting a Multiple-Byte Character User Database from ANSI to Unicode

You can convert ANSI databases containing multiple-byte characters to Unicode.

Caution When converting ANSI databases to Unicode, the shared memory or Hybrid Pipe Shared Memory (HPSM) transport type cannot be used; TCP/IP must be used.

Use the **codepage** command line argument of the **JadeConvertDb** application to specify the numeric value representing the appropriate codepage (for example, the Chinese PRC codepage is numerically **936**).

All fixed-length strings or String Large Objects (slobs) are converted, based on the selected codepage. For more details, see "Running the User Database Conversion Application Example".

Converting a multiple-byte character user database from ANSI to Unicode has the following implications.

- A slob can get smaller; that is, a pair of JADE ANSI characters can be a single-language character (that is, a multiple-byte), which converts to a single wide Unicode character.
- A string or a slob can have characters that cannot be translated. For example, the single-byte © copyright character can be entered in a codepage "English (New Zealand)", which is invalid when the conversion specifies **codepage=936**.
- The conversion handles the fixing of slob lengths in parent objects and the reporting of any translation failures.

When converting a multiple-byte ANSI database to Unicode, consider the following.

- Both the ANSI and Unicode binaries and databases *must* have the same level of binaries and hot fixes applied. This includes the system database ***.bin** files.
- To convert an ANSI database to Unicode:

  a. Start a database server for the ANSI database; for example:

  ```
  F:\Jade\a_bin\jadrap.exe ini=F:\Jade\a_jade.ini path=F:\Jade\a_system
  ```

  b. Start a Unicode **jadclient** program; for example:

  ```
  F:\Jade\u_bin\jadclient.exe ini=F:\Jade\u_jade.ini path=F:\Jade\a_system
  server=multiUser schema=TestUserSchema app=JadeConvertDb startAppParameters
  defaultPath=F:\Jade\u_system existing=true codepage=936 userSchema=true
  userData=true
  ```

  In these examples, **a_bin/a_system/a_jade.ini** represents the ANSI side and **u_bin/u_system/u_jade.ini** represents the Unicode side.

- If you want to extract your application schemas from ANSI and load them into Unicode before you run the **JadeConvertDb** application from the **jadclient** program, set the **userSchema** argument to **false**.

After you have converted your multiple-byte ANSI database to Unicode, check for reported errors. Your **jommsg*n*.log** file can contain the following message (the class number may vary). Although this is a known warning message that can be ignored, you should investigate the cause of any other messages carefully.

```
2014/10/10 15:07:25 00a8c-1420 JomLog:
jdiConvertCollBlockFromRemoteFormat: invalid class: 322
2014/10/10 15:07:25 00a8c-1420 JomLog: getCollBlockBufferLength: invalid collection
class: 322
```

If any translation errors occur, the **copyfile.log** in the **logs** directory can have output like that shown in the following example.

```
2014/03/25 22:16:12.828 00e88-1248 CvtDb: copyFile file=24 processing.
2014/03/25 22:16:12.843 00e88-1248 CvtDb: copyFile file=24 finished.
2014/03/25 22:16:12.859 00e88-1248 CvtDb: copyFile file=25 processing.
2014/03/25 22:16:12.859 00e88-1248 CvtDb: copyFile file=25 finished.
2014/03/25 22:16:12.875 00e88-1248 CvtDb: copyFile file=26 processing.
2014/03/25 22:16:13.187 00e88-1248 CvtDb: oid=[0.22.929.2.1:1] : 5706
                        (Untranslatable character)
2014/03/25 22:16:13.187 00e88-1248 CvtDb: oid=[0.36.929.2.1:1] : 5706
                        (Untranslatable character)
2014/03/25 22:16:13.187 00e88-1248 CvtDb: oid=[0.58.929.2.1:1] : 5706
                        (Untranslatable character)
2014/03/25 22:16:13.187 00e88-1248 CvtDb: oid=[0.97.929.2.1:1] : 5706
                        (Untranslatable character)
2014/03/25 22:16:13.203 00e88-1248 CvtDb: oid=[0.103.929.2.1:1] : 5706
                        (Untranslatable character)
2014/03/25 22:16:13.203 00e88-1248 CvtDb: oid=[0.169.929.2.1:1] : 5706
                        (Untranslatable character)
2014/03/25 22:16:13.234 00e88-1248 CvtDb: copyFile file=26 finished.
2014/03/25 22:16:13.296 00e88-1248 CvtDb: copyFile file=27 processing.
2014/03/25 22:16:13.312 00e88-1248 CvtDb: copyFile file=27 finished.
```

It is your responsibility to check this file and then deal with any conversion issues that are logged. If an untranslatable character is found, the output character is set to a question mark (**?**) symbol, which can happen for both fixed-length strings and slobs. For example, you can create a Workspace in the JADE development environment with **'1280.1'.asOid.inspect;** and check *all* of the string fields in the object for unanticipated question mark symbols. (The oid to use is specified in the **copyfile.log** file.)

Note The character conversion depends on the code page selected for the conversion. The display of accented characters depends on the locale and font used.

Although the data can be converted in the Unicode system, we recommend that you fix the ANSI source database and then run the database conversion again.

Impacts of Data Conversion on Your Current Systems

Data conversion is useful when you want to migrate your current system and convert your database into a format appropriate for the new platform.

When the conversion has been performed, you should perform a number of checks on your new database, including database certifications, logical certifications, and any application-specific checks that may be required.

When converting your JADE system to different hardware or operating system, you may need to address other issues, including:

- Create a version of your schemas that can work on both ANSI and Unicode, with the code that is ANSI/Unicode susceptible protected with appropriate usage of the **Application** class **isUnicode** method.

  Note It is best if the code changes happen in the ANSI system before the conversion takes place.

- If you encrypt passwords into a Binary value, be aware that the results of that encryption may be different in Unicode than it was in ANSI, and that the Binary values will not have been converted.
- If you store any large string values as compressed binaries to save storage space, you must uncompress these to normal strings before you perform the database conversion and then recompress them again when the database has been converted to Unicode.
- Thoroughly test any external system interfaces and any third-party libraries that you use, as some third-party libraries may not have Unicode versions.

  A similar approach may be necessary for TCP/IP interfaces.
- Check for any usage of Windows API calls, and if appropriate, change to the "wide" version if applicable; for example, **ShellExecuteA** in **shell32** needs to become **ShellExecuteW** in **shell32**.
- Locale issues; for example, sorting order.

Caution Your collection orders may change when converting from an ANSI database to a Unicode database or the reverse, as multiple-byte sequences in an ANSI database may result in objects being returned in a different order when converted to wide (Unicode) characters (or the reverse).

Operational Considerations when Converting User Databases

When converting a user database, consider the following.

- The JADE binaries on both operating system and hardware platforms must be on the same JADE level.
- When the database conversion occurs, ensure that there are no active clients and no running JADE applications.
- As the source database must be in a stable state when conversion occurs, no user application should be running.
- The source database cannot have any schemas that require reorganization. This restriction applies even if the files that are to be converted do not require reorganization.
- If a restart is required, entire map files are converted.

Chapter 5 Deploying Database Files

This chapter covers the following topics.

Overview

The standard approach to deploying patches or upgrades to applications in a runtime-only environment is to use the Schema Load utility to load a schema file containing all of the changes to the schema (either a selective or a complete extract of the schema).

However, an alternative approach is possible in cases where only methods or form definitions have been changed. For upgrades of this type, you can directly replace the database files that define the schema of your application.

Note For details about the system files required when deploying a JADE system and using the **jdbutilb** batch JADE Database utility **markOffline** command to mark system file as being offline for a deployed database, see "System Map Files", in Chapter 3 of the *JADE Development Environment User's Guide* and Chapter 1 of the *JADE Database Administration Guide*, respectively.

The database files that define the schema of your application are as follows.

- _userscm.dat
- _userxrf.dat
- _usergui.dat
- _userint.dat
- _userdev.dat

Caution This list does *not* include the **_rootdef.dat** file, which is the default user data file.

To upgrade a runtime-only installation at a site

1. Ensure that you have a backup copy of the entire database.
2. Replace the old **_userscm.dat**, **_userxrf.dat**, **_usergui.dat**, **_userint.dat**, and **_userdev.dat** files with the new files of the same names.
3. Run the reset timestamp operation in the JADE Database utility, to reset timestamps. For details, see "Resetting Timestamps", in Chapter 3 of the *JADE Database Administration Guide*.
4. Run the JADE application, to verify that the upgrade was successful.

Note You can upgrade an application in this way if methods or form definitions only have been changed between the previous release and the current release. You cannot upgrade in this way if any changes have been made to the structures of persistent classes, including adding, deleting, or modifying properties, changing dictionary keys, or changing collection memberships. If any of these changes have been made, the database files that define your schema will be incompatible with the runtime data and errors will result when you attempt to access the data.

Class, interface, final method numbers, and property subId numbers must also be identical between the previous and the current releases of the database files that define your schema. For details, see "Restrictions" under "Preserving Runtime Data".

Preserving Runtime Data

In some situations, it is possible to preserve user runtime data when installing a fresh JADE database or when copying runtime data directly from one database to another.

This is achieved by simply saving and restoring (or copying) the specific database files that contain your runtime data. These files are the **_rootdef.dat** file and all of the database files whose names do not begin with an underscore character (_). You specify these names when you define class maps for the classes in your schema.

Copying runtime database files is useful when you are performing any of the following actions.

- Transferring your complete schema to a fresh JADE database, with associated test data that you want to retain
- Passing a copy of the schema to another developer, with associated runtime data
- Installing a copy of your schema at a user site, with site-specific runtime data that you want to preserve

Caution When the runtime data contains direct references to schema objects such as classes, interfaces, properties, or methods, these references will no longer be valid when the schema is installed in a new database, as the object ids of these objects will almost certainly change during the load process.

If you want to retain direct references to schema objects, look up these objects dynamically rather than storing persistent references to them or write your own "fix up" code to correct the invalid references after you have restored the runtime data files.

To preserve your runtime data

1. Extract your schema or schemas from your existing JADE database.
2. Install the new database in a different directory.
3. Load the schema or schemas into the new database.
4. Copy the runtime data files from the old database to the new database.

   As a developer of the application, you should be aware of which files constitute the runtime data; that is, the **_rootdef.dat** file and all of the non-system files that you defined as map files in the schemas.
5. Run the JADE Database utility (**jdbutil**) to override the timestamp mismatch for the runtime data files. For details, see "Resetting Timestamps", in Chapter 3 of the *JADE Database Administration Guide*.

The new database is now ready for use.

Note The preservation of runtime data will fail unpredictably if you change class structures after reloading the schema but before copying the runtime data.

Restrictions

Copying runtime database files from one database to another is possible only if the following conditions apply.

- The defined structures of all classes must be identical in both databases.

  If changes have been made in either database that cause a class to require reorganization, the runtime data files are no longer compatible and cannot be copied.

- Class numbers, interface numbers, final method numbers, and property subId numbers must be identical in both databases. This is normally the case, as these numbers are included in extracted schema files, allowing the same numbers to be allocated when the schema is loaded into another database.

  However, problems may arise when loading schema files from different third-party sources, in which the schemas have number conflicts. The compiler reports a warning if clashes of this sort are detected when loading a schema file.

Chapter 6 Upgrading to a New Application Release

This chapter covers the following topics.

Overview

The following sections describe some of the situations in which JADE or a JADE user application might be patched or upgraded, and the ways in which such releases would be deployed.

No Changes to JADE

Caution If you overwrite the **_userscm.dat**, **_userxrf.dat**, _**usergui.dat**, **_userint.dat**, and **_userdev.dat** files, you must run the JADE Database utility (**jdbutil**) to reset the timestamp on these files.

If you do not do so, the database reports an error on start up. (For details, "Resetting Timestamps", in Chapter 3 of the *JADE Database Administration Guide*.)

User Schema Changes, No Reorganization

When there are changes to the user schema but none to the JADE product, perform one of the following actions if no reorganization is required.

- Load the schema file containing the patches, by using the Schema Load utility.
- Overwrite the old **_userscm.dat**, **_userxrf.dat**, _**usergui.dat**, **_userint.dat**, and **_userdev.dat** files with the new **_userscm.dat**, **_userxrf.dat**, _**usergui.dat**, **_userint.dat**, and **_userdev.dat** files.

User Schema Changes, and Reorganization

When there are changes to the user schema but none to the JADE product, perform the following action if reorganization is required.

- Load the schema file containing patches, and then reorganize the user data by using the JADE Schema Load utility or the batch **JadeSchemaLoader** application.

Patch Release of JADE

The following subsections describe upgrading a deployed user application to a patch release of JADE.

JADE Patch Release, User Schema Changes, No Reorganization

When there is a JADE patch release and changes to the user schema, perform the following actions if no reorganization is required.

1. Install the new library and executable files in your JADE working directory (for example, **c:\jade\bin**).
2. Install the JADE patches, by using the Schema Load utility.
3. Perform one of the following actions.
   - Load the schema file containing the patches, by using the Schema Load utility.
   - Overwrite the old **_userscm.dat**, **_userxrf.dat**, **_usergui.dat**, **_userint.dat**, and **_userdev.dat** files with the new **_userscm.dat**, **_userxrf.dat**, **_usergui.dat**, **_userint.dat**, and **_userdev.dat** files.

JADE Patch Release, User Schema Changes, and Reorganization

When there is a JADE patch release and changes to the user schema, perform the following actions if reorganization is required.

1. Install the new library and executable files in your JADE working directory (for example, **c:\jade\bin**).
2. Install the JADE patches, by using the Schema Load utility.
3. Load the schema file containing patches, and then reorganize the user data by using the Schema Load utility.

Recommended Practices when Upgrading a Production Database

When upgrading or applying patches to a production runtime database, we strongly recommend that you perform both of the following actions.

1. Conduct a "trial run" on a copy of the production database.

   Take a copy of the production database and then apply the upgrade to this copy. Ensure that the patches can be applied without errors and that the reorganization (if required) can be completed successfully.

   Run the application to ensure that it functions correctly. If possible, run both the original and the upgraded (copy) application in parallel for a period.

2. Take a backup copy of the production database before doing the final upgrade.

   It is essential that you take a backup copy of the database before performing the upgrade. This will allow you to revert to the original database should the upgrade be unsuccessful for any reason.

Appendix A Customizing the Deployment Upgrade Process

The Customizable Deployment Upgrade (CDU) utility provides simple customization for the upgrading of deployed systems.

The Customizable Deployment Upgrade enables you to:

- Simplify the upgrade for deployed systems
- Prefill installation parameters for a specific application or customer site
- Brand the Customizable Deployment Upgrade with your product name and product graphics for your customers

The Customizable Deployment Upgrade utility is distributed on the release medium in the **CDU** directory.

Caution As there is no recovery of the upgrade process, backup your database before starting the Customizable Deployment Upgrade process.

To customize the deployment upgrade process

1. Copy the **CDU** directory from the release medium and unset the **Read Only** attribute for the **setup.ini** file on **disk1**.
2. Run the **setup.exe** program on **disk1**, and perform a trial installation.

   Installation parameters are written to the **setup.ini** file.
3. To create the configurations that you require, simply edit this **setup.ini** file, as follows.

   a. To change the default **JADE *version*** product name (where the ***version*** value is the JADE release; for example, **JADE 2018**) to your own product name, change both the **Product** parameter value in the [Startup] section and the name of the [JADE *version*] section to your product name.

   Note Both the [Startup] section **Product** parameter and the product name section must have the same value.

   The parameters in the [Startup] section are utilized by the InstallShield program.

   b. To hide the display of dialogs during the deployment upgrade process, set the appropriate **Show*XXXX*** parameters to **No**, or set the appropriate parameters to **Yes** to display the installation dialogs.

   These parameters and their default settings are listed in the following table.

Parameter	Installation Dialog	Default Value
ShowWelcome	Welcome to the InstallShield Wizard	Yes
ShowLicense	Software License Agreement (display **LicenseFile**)	No
ShowInfo	Information (display **InfoFile**)	No
ShowUpgradeDir	Select Installation Folders	Yes

Parameter	Installation Dialog	Default Value
ShowLicenseKey	User Information	**No** if no license key is required for an upgrade; **Yes** if a license key is required
ShowSummary	Start Copying Files	Yes
ShowReadme	Setup Completed! (display **readme.txt** file)	No
ShowFinish	Setup Completed!	Yes

c. You can use the parameters listed in the following table to define file locations and to supply pre-defined values.

Parameter	Enables you to specify …
BackColor	The color of the upgrade process background (defaults to **BLACK**), in the RGB format *nnn,nnn,nnn*; for example, **128,0,0** specifies light red.
BackgroundBitmap	The name of the file (defaults to **title.bmp**) that contains the background bitmap displayed during the upgrade process.
LicenseFile	The name of the file (defaults to **license.txt**) in which your licensing agreement is located, if the value of the **ShowLicense** parameter is **Yes**.
InfoFile	The name of the text file (defaults to **Info.txt**) that contains upgrade-specific information or instructions during the upgrade process, if the value of the **ShowInfo** parameter is **Yes**.
UpgradeInstallDir	The base installation directory (displayed in the **Install Directory** text box of the Select Installation Folders dialog).
UpgradeDbDir	The database directory that you want to upgrade (displayed in the **Database Directory** text box of the Select Installation Folders dialog).
UpgradeBinDir	The binaries directory that you want to upgrade (displayed in the **Executable Directory** text box of the Select Installation Folders dialog).
UpgradeINIFile	The name of the JADE initialization file that you want to use during the upgrade process (displayed in the **JADE INI File** text box of the Select Installation Folders dialog).
LicenseName	The license company name (displayed in the **License Name** text box on the User Information dialog).
LicenseKey	The license key (displayed in the **License Key** text box on the User Information dialog). Do not include dash symbols (-).
ReadmeFile	The name of the text file (defaults to **readme.txt**) that you want to display at the end of the upgrade process if the value of the **ShowReadme** parameter is **Yes**.

The **title.bmp**, **readme.txt**, and **license.txt** files on **disk1** are default files. If you do not use any of the parameters in the above, you can overwrite or remove the files. File names for the parameters in the table are relative to the directory in which the **setup.ini** file is located (that is, to **disk1**).

The directory paths specified in the **UpgradeInstallDir**, **UpgradeDbDir**, **UpgradeBinDir**, and **UpgradeINIFile** parameter values can include a Windows environment variable; for example, **%USERPROFILE%\jade**.

4. Set the **Read Only** attribute for the **setup.ini** file to preserve the parameter values and then test your installation.

Caution The upgrade setup initialization file must be called **setup.ini**. You must have a separate configured set of files on **disk1**, each containing a **setup.ini** file (and supporting files) for each product or customer whose JADE deployment upgrade you want to customize.

Following the successful upgrade, you will need to perform the following actions.

1. Copy any additional files (for example, configuration files) that need to reside in the binaries directory
2. Ship recompiled versions of any external method Dynamic Link Libraries (DLLs)