# Schema Load User's Guide

**VERSION 2020.0.02**

jade

# Contents

Contents iv

# Before You Begin

The *JADE Schema Load User's Guide* is intended as the main source of information when you are loading schemas into a deployed JADE database.

## Who Should Read this Guide

The main audience of the *JADE Schema Load User's Guide* is expected to be system administrators.

## 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 Initialization File Reference* | Maintaining JADE initialization file parameter values |
| *JADE Installation and Configuration Guide* | Installing and configuring JADE |
| *JADE Report Writer User's Guide* | Using the JADE Report Writer to develop and run reports |
| *JADE Runtime Application Guide* | Administering JADE deployed runtime applications |
| *JADE Web Application Guide* | Implementing, monitoring, and configuring Web applications |

## Conventions

The *JADE Schema Load User's 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.<br><br>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. |

| Convention | Description |
|---|---|
| *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.<br><br>Italic type also signals a new term. An explanation accompanies the italicized type.<br><br>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 "Controlling the Deletion of Missing Elements" cross-reference to display that topic. |
| Bracket symbols ( [ ] ) | Indicate optional items. |
| Vertical bar ( \| ) | Separates alternative items. |
| `Monospaced font` | Syntax, code examples, and error and status message text. |
| ALL CAPITALS | Directory names, commands, and acronyms. |
| Small font | Keyboard shortcut keys. |

Key combinations and key sequences appear as follows.

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

# Chapter 1 Loading Schemas

This document covers the following topics.



# Overview

The GUI and non-GUI versions of the Schema Load utility (**jadload** and **jadloadb**, respectively) and the non-GUI RootSchema **JadeSchemaLoader** application are supplied with the JADE release, to enable you to load runtime-only JADE applications.

A set of Schema Load utility database files that contain no JADE development information is also supplied on the JADE release medium. The **Jade Loader** icon is installed in your JADE program folder when you install JADE.

The standalone Schema Load utility and the **JadeSchemaLoader** application enable you to load JADE applications into a cut-down database, which excludes all JADE development classes and methods. You can load:

- User schemas into a runtime-only JADE database that can then be deployed at user sites
- Patches to your JADE system into a full JADE environment or a runtime-only environment
- Patches to a runtime-only database deployed at a user site

The Schema Load utility and the non-GUI **JadeSchemaLoader** application enable you to:

- Add new classes, interfaces, and methods
- Modify existing methods
- Rename schemas
- Add, delete, or modify properties and inverse references
- Delete existing schemas, classes, interfaces, packages, or methods
- Reorganize user data
- Load form, RPS database map, relational view, and external database definitions

- Load JADE Report Writer data

  As all JADE Report Writer files (that is, files with a suffix of **rwv**, **rwo**, **rwf**, **rwu**, **rwr**, and **rwa**) are loaded only into the current schema version, they should therefore be loaded after any database reorganization transition.

**Note** Exception 5012 is raised if you attempt to load a Unicode schema into an ANSI JADE environment.

You can run the **JadeSchemaLoader** application:

- As a presentation (thin) client
- In single user or multiuser mode as a standard (fat) client
- In single user or multiuser mode from a non-GUI client
- From your application code

# Before You Get Started

**Caution** Before installing patches and performing a reorganization in a runtime-only environment, you *must* take a backup copy of the database as the reorganization can fail under some circumstances (for example, if there is insufficient disk space or a change in the new schema definition is not compatible with runtime data). If this occurs, your JADE application may be left in an unusable state. You should then restore the database from backup, correct the problem that caused the reorganization to fail, and then repeat the load and reorganization.

For details about correcting data after a reorganization, see "Correcting Data if a Reorganization Fails", in Chapter 14 of the *JADE Developer's Reference*.

**Note** If you have mapping method logic on subclassed controls that rely on this logic when executing, you must protect that logic from situations where properties of subclassed controls are accessed or referenced by JADE processes such as the JADE Painter, JADE Translator utility, or the loading of schemas.

If external databases, ActiveX (COM) libraries, relational views, or RPS mappings have been extracted separately by using the appropriate **Extract** command, the order in which you should load extracted files is as follows.

1. Relational view extract file or files.
2. Selective schema extract file.
3. External database, RPS mapping, or ActiveX (COM) library extract file or files.

If a reorganization is required after the .**scm** file load and before the .**ddb** file load, you can load the .**ddbx** file (RPS mapping information) before the reorganization, to keep the RPS mappings up-to-date with the schema changes. After the reorganization, the .**ddb** file is loaded as normal, to complete the changes. When extracting a schema with an RPS mapping and the DDX format style is selected, a separate .**ddbx** file is produced as for a DDB extract but the file contents are in XML format.

For details about loading an encrypted schema file, see "Encrypting Schema Source Files", in Chapter 10 of the *JADE Development Environment User's Guide*.

If you are loading multiple schemas and you want to suppress the display of all warning messages resulting from compiling schema files (for example, class number conflict messages), set the **SuppressWarningDialogs** parameter in the [JadeCompiler] section of the JADE initialization file to **true**. Warning messages are then output only to the message log.

The default installation sets up the **JADE Utilities\JADE Loader** shortcut for the JADE Schema Load utility.

The **Jade Loader** icon is installed in your JADE program folder when you install JADE. The default installation sets up the **JADE Utilities\JADE Loader** shortcut for the JADE Schema Load utility. For details about installing the Schema Load utility, see "Installing Your JADE Software", in Chapter 1 of the *JADE Installation and Configuration Guide*.

The following table lists examples of the properties required to run the JADE Schema Load utility in single user mode.

| Property | Example |
|---|---|
| Command line | jadload.exe path=c:\jade\system |
| Working directory | c:\jade\bin |

If you want all methods loaded from the schema file to be compiled during the load process, including those whose source is the same as their source in the database:

- Specify the **compileUnchangedMethods** argument in the command line with a value of **true**
- Check the **Compile Unchanged Methods** check box in the Additional Parameters group box on the extended Load Schema dialog

By default, the load process skips methods whose source is the same as their source in the database.

Exception 6429 is raised when you attempt to load a schema with the default map file marked as partitionable (that is, the **DbFile** class **partitionable** property is set to **true**).

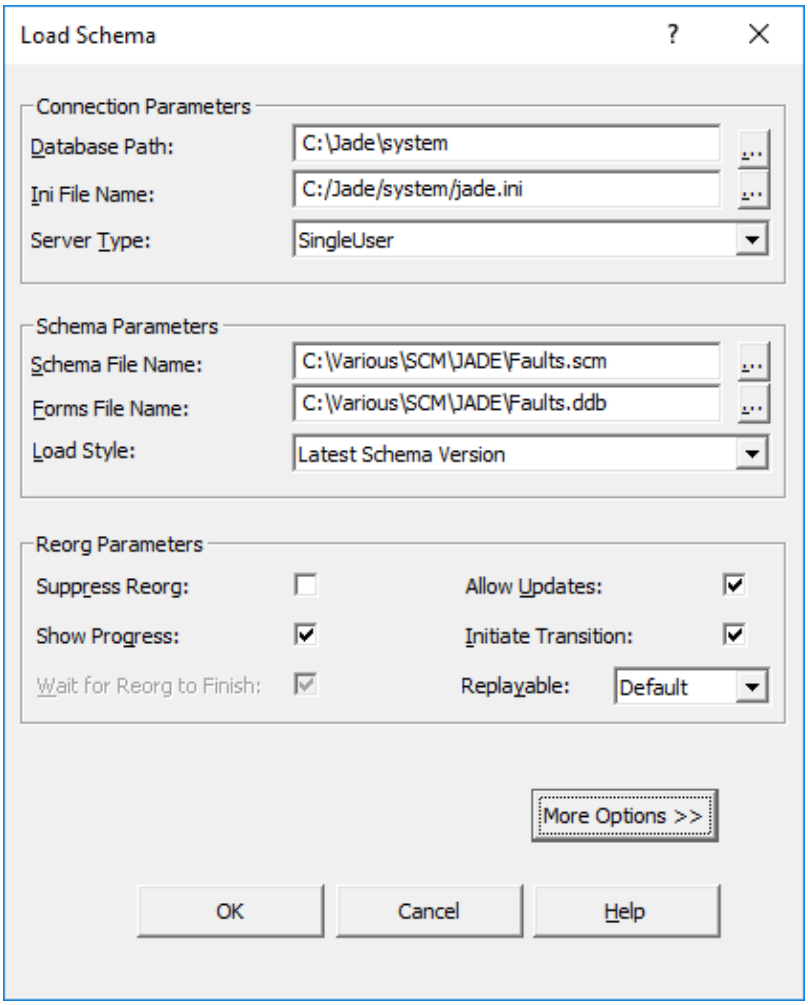
# Loading Schemas using the Schema Load Utility

**To initiate the JADE Schema Load utility**

1. Ensure that you have a full backup of your database.
2. Select the **Jade Loader** program icon from your JADE program folder.

**Note** If a schema load is attempted when a reorganization is in progress (regardless of whether the reorganization progress dialog is displayed), the load fails. For details about reorganizing schemas, see "Reorganizing Your Schema", in Chapter 3 of the *JADE Development Environment User's Guide*.

The Load Schema dialog, shown in the following image in an example of its initial form, is then displayed in the JADE Schema Load Utility window. An extended form of the dialog is displayed when you click the **More Options >>** button. For details, see "Specifying Additional Load Options", later in this document.




The controls are empty when the Schema Load utility is run for the first time. If the utility has been run previously, the values that were specified the last time the utility was run are displayed in the Connection Parameters group box controls and in the **Schema File Name** and **Forms File Name** text boxes. Other values are set to default values in the following steps of this instruction.

To load a JADE Report Writer file, specify the file name and appropriate suffix (for example, **AllReports.rwa** for a single extract file or **MyRptView.rwv** for a reporting view) in the **Schema File Name** text box.

3. In the **Database Path** text box, specify the path in which your JADE database files are located. The database path must exist. If you are unsure of your database directory, click the adjacent browse button (indicated by the … points of ellipsis symbol) to display the common Browse for Folder selection dialog that enables you to select the database path in which your user data is located.

4. In the **Ini File Name** text box, specify the name (and optionally the file path) of your JADE initialization file. The database path is assumed if you do not specify the file path. If you are unsure of your initialization file, click the adjacent browse button (indicated by the … points of ellipsis symbol) to display the common Choose File dialog that enables you to select the initialization file that is used.

5. In the **Server Type** combo box, select the mode in which you want to perform the schema load process. Select the **multiUser** value if you want to connect to a running server or the **singleUser** value if you want to connect in single user mode.

6. In the **Schema File Name** text box, specify the full path of the location of your schema (.**scm**) file. If you are unsure of your schema file directory or name, click the adjacent browse button (indicated by the … points of ellipsis symbol) to display the common Choose File dialog that enables you to select the schema file that you want to load.

   **Note** If you are loading a partial file, this may be a .**scm** file or a .**cls** file for a class or interface extract. This can also be a multiple schema file (.**mul**) for loading multiple schemas.

7. In the **Forms File Name** text box, specify the full path of the location of your form and data definition (.**ddb** or .**ddx**) file if you are also loading a forms file. If you are unsure of your forms definition file directory or name, click the adjacent browse button (indicated by the … points of ellipsis symbol) to display the common Choose File dialog that enables you to select the forms file that you want to load.

8. In the **Load Style** list box, select the load style, as follows.

   - The default **Latest Schema Version** value loads a new schema or loads the schema as the latest schema version (a new version will be created if required), and allows structural changes.
   - The **Current Schema Version** value loads the schema into the existing current schema version, which may potentially affect current runtime behavior. As structural changes are not allowed, the load will not proceed if structural changes are to be introduced.

     If you are loading a schema file containing only method changes into a multiuser system, select this option, to ensure that no structural changes are attempted that would require a reorganization of the database and impact existing users. If a structural change is detected, the schema load returns an error.

     **Note** As all JADE Report Writer files (that is, files with a suffix of **rwv**, **rwo**, **rwf**, **rwu**, **rwr**, and **rwa**) are loaded only into the current schema version, they should therefore be loaded *after* any database reorganization transition.

   - The **Only Structural Versioning** value loads only structural changes into the latest version and does not version methods or other non-structural entities. This may potentially affect current runtime behavior.

9. Check the **Suppress Reorg** check box if you want to prevent the reorganization of any loaded schemas that require reorganization on completion of the load process.

   By default, any loaded schemas that require reorganization on completion of the load process are reorganized, regardless of whether they required reorganization prior to the load.

   **Notes** If control classes require reorganization or new **Control** subclasses are defined before loading a forms definition file, a reorganization will be initiated before the forms definition file is loaded. In this case, you must load the Relational Population Service (RPS) .**ddbx** file, if any, before the reorganization, to keep the RPS mappings up-to-date with the schema changes. (When extracting a schema with an RPS mapping and the DDX format style is selected, a separate .**ddbx** file is produced as for a DDB extract but the file contents are in XML format.)

   After the reorganization, the .**ddb** forms definition file is loaded as normal, to complete the changes.

   If such a reorganization is required and you check this control, the forms load process will raise an exception.

10. If you are running in multiuser mode and you do not want to allow other users to continue development and update the database while the reorganization is in progress, uncheck the **Allow Updates** check box. (The **Initiate Transition** check box is then checked if it was unchecked and it is disabled.)

    If updates are disabled while the reorganization is in progress, the reorganization must initiate the transition. Any error occurring during the reorganization results in the reorganization being aborted; that is, the reorganization cannot be restarted. For details, see "Allowing Updates" under "Reorganization Options", in Chapter 14 of the *JADE Developer's Reference*.

    As it is not possible for other applications to update the database in single user mode, this control is unchecked and disabled if **SingleUser** is selected in the **Server Type** combo box.

    In multiuser mode, the **Allow updates** check box is checked by default, to allow updates to proceed before the reorganization transition is initiated; for example, other users can still modify (that is, edit and compile) methods in the development environment.

11. If you do not want the reorganization progress dialog displayed during the reorganization, uncheck the **Show Progress** check box.

    When this is checked (the default value), you can monitor the progress of the reorganization and you can cancel the reorganization at any time.

12. If you do not want the transition to be initiated for any reorganization, uncheck the **Initiate Transition** check box, to suspend the reorganization transition. This check box is checked and disabled if you unchecked the **Allow Updates** check box in step 10 of this instruction.

    For details, see "Initiating Transition" under "Reorganization Options", in Chapter 14 of the *JADE Developer's Reference*.

13. If you want to wait until the reorganization is complete, check the **Wait for Reorg to Finish** check box. As the load process in single user mode must wait for any reorganization to complete, this control is checked and enabled if **SingleUser** is selected in the **Server Type** combo box.

    By default, this check box is not checked. In multiuser mode, you can exit from the JADE Schema Load utility and the reorganization continues.

14. If archival recovery is enabled (that is, the **EnableArchivalRecovery** parameter in the [PersistentDb] section of the JADE initialization file is set to **true**), a replayable reorganization of your JADE database can be performed when the schema is loaded. For details, see "Replayable Reorganizations" under "Reorganization Options", in Chapter 14 of the *JADE Developer's Reference*.

    Select **Default** in the Replayable combo box if you want to perform a replayable reorganization only if the schemas are loaded into a primary database.

    Select **True** if you want to perform a replayable reorganization of your JADE database when the schema is loaded. If archival recovery is not enabled, the reorganization cannot be replayed.

    Select **False** if you do not want to perform a replayable reorganization of your JADE database when the schema is loaded.

15. To display and select additional, delta, and system parameters for your schema load, click the **More Options >>** button.

The extended Load Schema dialog is then displayed. For details, see "Specifying Additional Load Options", later in this document.

## Controlling the Deletion of Missing Elements

By default, the Schema Load utility prompts you to confirm the deletion of any schema elements (such as classes, properties, or methods) that exist in the JADE database but are not defined in the incoming schema file.

**Note** An application in the database that is used in a package is not deleted (over-written) when a complete schema definition containing that application is loaded.

Use the **jadload** executable program **deleteIfAbsent** command line argument to specify element deletion control different from the default behavior when loading your runtime-only application by using the Schema Load utility.

The settings of the **deleteIfAbsent** command line argument are listed in the following table.

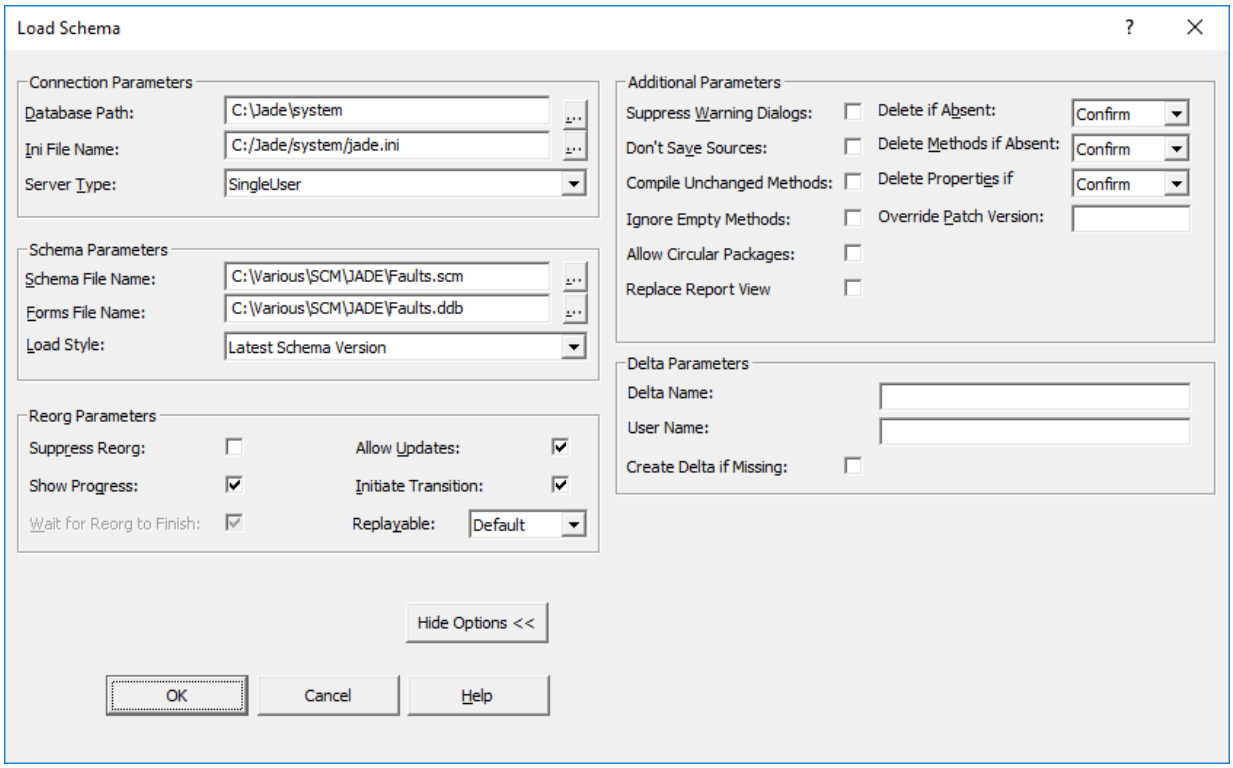
| deleteIfAbsent Argument | Action |
|---|---|
| true | Always delete absent elements (no confirmation) |
| false | Never delete absent elements (no confirmation) |

If you want to deploy patches at a user site without requiring any intervention at the site, set up the **jadload** executable program icon with the **deleteIfAbsent** command line argument set to **true**.

## Specifying Additional Load Options

Use the extended Load Schema dialog, displayed when you click the **More Options >>** button on the initial Load Schema dialog, to specify additional, delta, and system parameters for your schema load. (See also "Loading Schemas using the Schema Load Utility", earlier in this document.)

The following image shows an example of the extended Load Schema dialog.



### To specify more load options

1. In the **Delete If Absent** combo box, select when you want to confirm the deletion of any schema elements (such as classes, interfaces, properties, or methods) that exist in the JADE database but are not defined in the incoming schema file when you want to release JADE applications without method source code; for example, when you release a schema containing JADE applications. (For details about the deletion of missing elements, see "Confirmation of Deletion", in Chapter 10 of the *JADE Development Environment User's Guide*.)

   **Notes** The value that you specify or select in this combo box populates the **Delete Properties If Absent** and **Delete Methods If Absent** combo boxes. The **Delete Properties If Absent** and **Delete Methods If Absent** in steps 4 and 6 of this instruction provide you with more-detailed control over the deletion of missing methods and properties, by selecting a different value for these elements, if required.

   An application in the database that is used in a package is not deleted (over-written) when a complete schema definition containing that application is loaded.

   The values from which you can choose are listed in the following table.

| Option | Action |
|---|---|
| Confirm | Prompt to confirm the deletion of any element that exists in the JADE database but is not defined in the incoming schema file (the default value) |
| Never | Never delete absent elements (no confirmation) |
| Always | Always delete absent elements (no confirmation) |

   If you want to deploy patches at a user site without requiring any intervention at the site, select the **Always** value.

2. Check the **Suppress Warning Dialogs** check box if you want to suppress warning dialogs resulting from compiling schema files (for example, class number conflict messages). Warning messages are then output only to the message log.

   Suppressing warning dialogs is useful when loading multiple schemas. (You can also set the **SuppressWarningDialogs** parameter in the [JadeCompiler] section of the JADE initialization file to **true**.)

3. If you want to specify a value for the deletion of methods that differs from the value that you selected in the **Delete If Absent** combo box in step 2 of this instruction, select the required value in the **Delete Methods If Absent** combo box, using one of the values listed in the table in step 2 of this instruction.

4. Check the **Don't Save Sources** check box if you want to delete the source code from all JADE methods when the schema is loaded. By default, source code is retained when the schema is loaded.

   If you want to release a JADE environment that does not contain source code, you must perform a full schema load for all schema loads into that environment, check the **Don't Save Sources** check box, and specify the **compileUnchangedMethods** argument with a value of **true** on the **jadload** command line.

   When the methods that are being loaded already exist and the source of those methods has not changed, the methods are not compiled. The source is not updated so it is therefore not deleted.

   For details about including recompiled methods in patch versioning when a patch is to be applied to a schema that does not have source available, see "Enabling or Disabling Patch Versioning" and "Setting Up a Patch Number", in Chapter 3 of the *JADE Development Environment Administration Guide*.

5. If you want to specify a value for the deletion of properties that differs from the value that you selected in the **Delete If Absent** combo box in step 2 of this instruction, select the required value in the **Delete Properties If Absent** combo box, using one of the values listed in the table in step 2 of this instruction.

6. Check the **Compile Unchanged Methods** check box if you want all methods loaded from the schema file to be compiled during the load process, including those whose source is the same as their source in the database.

   By default, the load process skips methods whose source is the same as their source in the database (that is, this check box is unchecked).

   If you want to release a JADE environment that does not contain source code, you must perform a full schema load for all schema loads into that environment and check both the **Don't Save Sources** check box in step 5 of this instructions and this **Compile Unchanged Methods** check box.

7. In the **Override Patch Version** text box, specify a value in the range **1** through **Max_Integer - 1** (that is, 2,147,483,646) if you want to override the patch version number in schema entities.

   All entities loaded in the schema file then have their patch version number set to the specified value, regardless of whether patch versioning is enabled for the schema. The specified value does not have to match an existing defined patch number. If the value is the same as an existing patch number, the state of the existing patch version is ignored (that is, the existing patch number in the schema file may be closed).

   A value of zero (**0**) or blank retains the patch numbers specified in the schema file.

8. Check the **Ignore Empty Methods** check box if you want to override the default behavior when an empty method implementation is found while loading a schema file. The default behavior removes the existing source for the method and marks it as needing compilation (that is, if you do not check this check box, the default behavior is performed).

   Check this check box when you want to leave the existing source and code of a method with an empty implementation untouched during a schema load.

   The expected use of this check box occurs when an extracted schema that includes methods that had their source stripped is loaded into an environment where those methods had previously been loaded from an encrypted schema.

   Overriding the default behavior by checking this check box is considered potentially unsafe.

9. Check the **Allow Circular Packages** check box if you want to allow a circular dependency between packages in the schema hierarchy; that is, to permit the loading of an incomplete package (for example, **Schema1** exports **Package1** and imports **Package2**, while **Schema2** exports **Package2** and imports **Package1**).

   When you subsequently create a package that would result in circularities, you are prompted to confirm that you want to continue and allow a circular dependency between packages in the schema hierarchy.

   By default, this check box is unchecked; that is, the packages that are available for import are those that would not result in circular dependencies in the schema hierarchy.

   If you load two schemas that are not circular and you load an importer before the exporter:

   - An incomplete schema is loaded.
   - The form and data definition file (.**ddb** or .**ddx**) is not loaded.
   - Exception 8527 (*Load results in one or more incomplete schemas*) is raised.

     When the **StandardExitValues** parameter in the [FaultHandling] section of the JADE initialization file is set to **true**, you can map this exception to generic exit value **8**. (For details about generic exit values, see "Enabling the Use of Generic Exit Values for Windows", in Appendix A of the *JADE Installation and Administration Guide*.)

Incomplete schema definitions and their usages are resolved by loading the export definition and then reloading the importing files. (To completely resolve all of the interdependencies, more than two iterations may be required.)

An incomplete schema is displayed with the background color of a versioned schema (which defaults to red) in the Schema Browser. The incomplete schema, its subschemas, Class Browser, and any wizards for that schema and subschemas, are not available. To make the Class Browser available, it is your responsibility to make the entire schema complete (by loading the exporting schema and then reloading the incomplete schema).

**Note** An incomplete schema can be versioned, changing the background color, and not allowing the opening of a Class Browser on either version.

10. Check the **Replace Report View** check box if you want a view file being loaded to replace any existing view with the same name. This removes view items (types, features, root collections, joins, and script methods) that are not in the incoming JADE Report Writer view extract file and are not used in any existing report definitions if a reporting view with the same name already exists. Those items that are used in existing reports are retained.

    By default, if a reporting view with the same name already exists, the views are merged.

11. In the **Delta Name** text box, specify the delta into which all loaded methods will be checked out.

    When you specify a value in this text box, you must also specify a value in the **User Name** text box, in step 12 of this instruction. When you specify the delta and user names and the specified delta does not exist, the load process fails if you have not checked the **Create Delta if Missing** check box in step 13 of this instruction.

12. In the **User Name** text box, specify the user name to associate with the delta into which the schema is loaded.

    When you specify a value in this text box, you must also specify a value in the Delta Name text box in step 11 of this instruction. When you specify the delta and user names and the specified delta does not exist, the load process fails if you have not checked the **Create Delta if Missing** check box in step 13 of this instruction.

13. Check the **Create Delta if Missing** check box if you want a delta created if you specify the name of a delta that does not exist in the **Delta Name** text box in step 11 of this instruction.

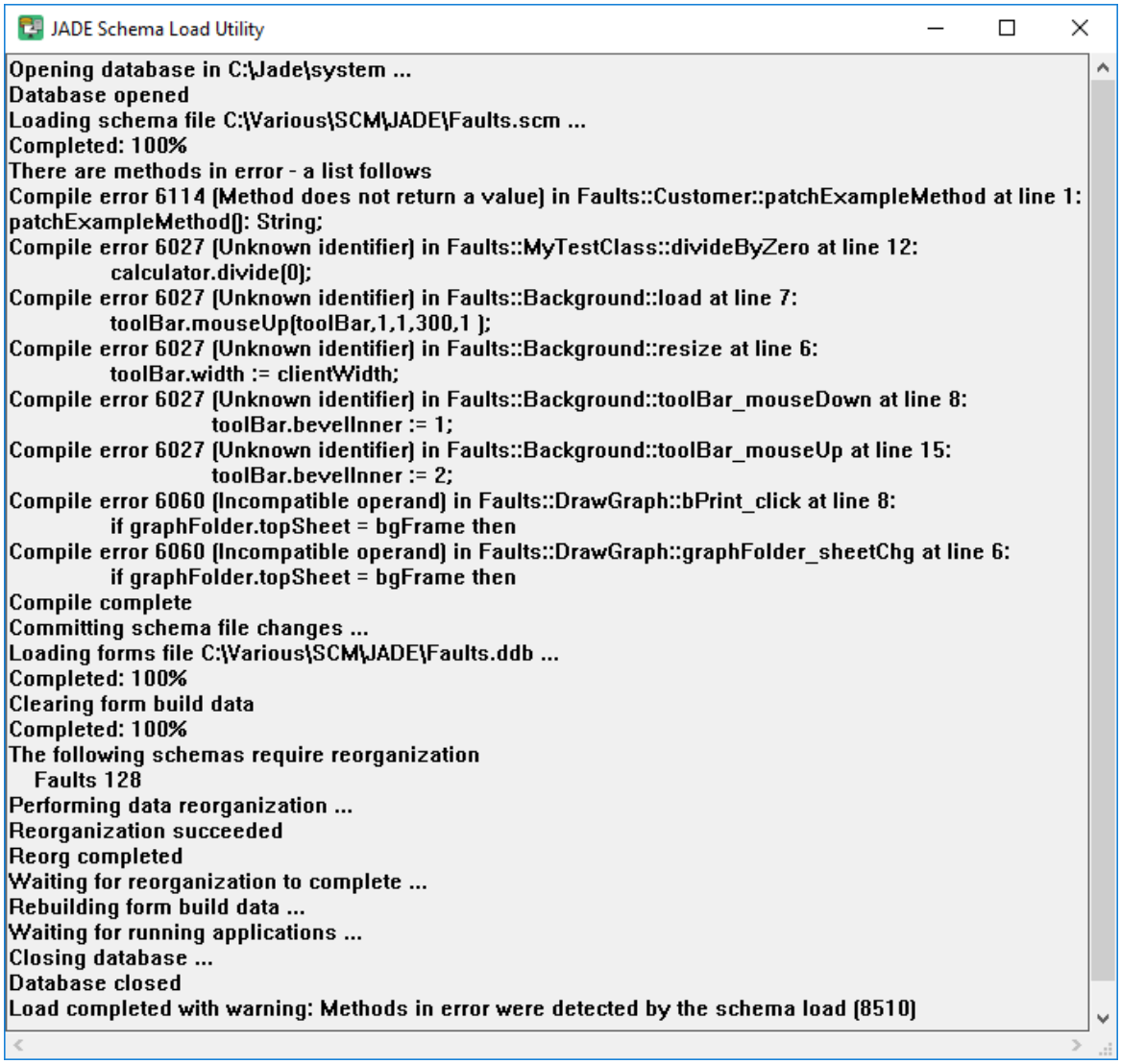
    By default, this check box is unchecked; that is, the delta is not created and the load process fails if you no do not check this check box and you specify values in the **Delta Name** and **User Name** text boxes in steps 11 and 12 of this instruction.

14. To hide the display of additional, delta, and system parameters and display the initial Load Schema dialog, click the **Hide Options <<** button. (For details, see "Loading a Schema and Forms into a JADE Database", in the previous section.)

15. Click the **OK** button. (Alternatively, click the **Cancel** button to abandon your selections.)

The load of your JADE schema is then started.

The background window, shown in the following image, displays the progress of the load operation.




By default, the background window output buffer can contain up to 64K characters. New messages are always appended to this buffer and the window is refreshed from the buffer when a message is output. The **DisplayBufferSize** parameter in the [JadeLoader] section of the JADE initialization file enables you to configure this value in the metric format (for example, **K**, **M**, and so on), if required.

## What Happens Next

Your schema and optional forms file are then loaded. If an error is detected in the schema file, an editor window is opened to enable you to edit the schema file. (You cannot edit a forms definition file.)

When the schema and forms files have been loaded, the following actions are performed.

- If you specified that you did not want source code loaded, a message is displayed informing you that the method sources will be removed. Click the **OK** button in the message box to confirm that the source code is to be removed from all methods.

- The Schema Load utility checks whether any classes require reorganizing. If so, a reorganization of all required classes is automatically initiated.

**Caution** You should then run the JADE Database utility to certify and compact the files before you release them to a third-party. (For details, see Chapter 3, "Administering the JADE Database", in the *JADE Database Administration Guide*.)

For details about reorganization, see Chapter 14 of the *JADE Developer's Reference*.

# Loading Schemas in Batch Mode using jadloadb

The **jadloadb** program enables you to automate the Schema Load utility by running it in batch mode. Run the batch Schema Load utility (for example, from a command script), specifying the following.

```
jadloadb path=database-path
         ini=initialization-file-name
         [server=singleUser|multiUser]
         schema-loader-arguments
         [help]
```

The schema loader arguments are described in "Loader Arguments", later in this document.

The following is an example of the command for loading the **Test** schema.

```
jadloadb path=d:\jade\system schemaFile=d:\jade\scm\test.scm
ddbFile=d:\jade\scm\test.ddb dontSaveSources=true ini=d:\jade\myjade.ini
deletePropertiesIfAbsent=true loadStyle=onlyStructuralVersioning
replayableReorg=true compileUnchangedMethods=true
```

**Notes** As the batch Schema Load utility does not read parameter values (for example, the **IniFile** and **Path** parameters) from the [JadeLoader] section of the JADE initialization file, you must specify all argument values on the command line if you do not want to use default values.

If a schema load is attempted when a reorganization is in progress, the load fails. For details about reorganizing schemas, see "Reorganizing Your Schema", in Chapter 3 of the *JADE Development Environment User's Guide*.

The batch Schema Load utility program displays a progress report of each phase of the load operation. Standard load 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*.)

An error is returned if a file listed in a multiple (**.mul**) file does not exist.

At the end of the load operation, the running display provides a completion report. If the **jadloadb** executable program fails, a non-zero exit code is returned and an error message is displayed; for example, if the database directory was invalid.

Detailed results of the loading of files extracted from the JADE Report Writer, including the success or otherwise of the load, are logged to the **jadereportwriter*n*.log** file in the log directory.

If an exception occurs during the load process, the exception text is output to the console window and the exit code from the process is non-zero.

**Note** An exception (that is, *8510 – Methods in error were detected by the schema load*) is raised if the batch schema load process detected methods that are in error. Although this is a warning only and the loading of the schema has completed, you should correct the methods that are in error and then run the batch schema load again.

The **jadloadb** program arguments are described in the following subsections.

## path

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

The database path must exist; for example:

```
path=d:\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 **jade.ini**.

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

## server

The optional **server** argument enables you to specify the mode in which the load operation is actioned. By default, this argument is set to **singleUser**.

If you want to connect to a running server by loading schemas and forms in multiple user mode, set this argument to **multiUser**.

## help

The optional **help** argument displays the required arguments and their values.

# JadeSchemaLoader Application

The **JadeSchemaLoader** application enables you to deploy schema changes directly to an application server running in single user mode without having to stop any applications that are running.

**Caution** For details about **JadeSchemaLoader** application security, see "Security Considerations", later in this document

You can run this non-GUI application:

- As a presentation (thin) client

  The loader parameters, or arguments, are specified between the **startAppParameters** and **endAppParameters** arguments.

  ```
  jade.exe schema=RootSchema app=JadeSchemaLoader
  ini=JADE-initialization-file-path
  AppServer=remote-TCP/IP-address-or-name-of-application-server
  ```

```
AppServerPort=TCP/IP-port-number-of-application-server startAppParameters
load-command-line-arguments endAppParameters
```

The following is an example of the command line for loading a **Test** schema from a presentation client.

```
jade.exe schema=RootSchema app=JadeSchemaLoader path=d:\jade\system
ini=d:\jade\myjade.ini AppServer=MyAppServer AppServerPort=1234
startAppParameters schemaFile=d:\jade\scm\test.scm ddbFile=d:\jade\scm\test.ddx
dontSaveSources=true deletePropertiesIfAbsent=true
loadStyle=onlyStructuralVersioning replayableReorg=true
compileUnchangedMethods=true endAppParameters
```

- From **jade.exe** in single user or multiuser mode standard (fat) client

  The loader arguments are specified after the **startAppParameters** argument.

  ```
  jade.exe schema=RootSchema app=JadeSchemaLoader
  ini=JADE-initialization-file-path path=database-path
  server=multiUser|singleUser startAppParameters load-command-line-arguments
  ```

  The following is an example of the command line for loading a **Test** schema from a standard (fat) client.

  ```
  jade.exe schema=RootSchema app=JadeSchemaLoader path=d:\jade\system
  ini=d:\jade\myjade.ini startAppParameters schemaFile=d:\jade\scm\test.scm
  ddbFile=d:\jade\scm\test.ddb dontSaveSources=true deletePropertiesIfAbsent=true
  loadStyle=onlyStructuralVersioning replayableReorg=true
  compileUnchangedMethods=true
  ```

- From **jadclient.exe** in single user or multiuser mode from a non-GUI client

  The loader arguments are specified after the **startAppParameters** argument.

  ```
  jadclient.exe schema=RootSchema app=JadeSchemaLoader
  ini=JADE-initialization-file-path path=database-path
  server=multiUser|singleUser startAppParameters load-command-line-arguments
  ```

  The following is an example of the command line for loading a **Test** schema from a non-GUI client.

  ```
  jadclient.exe schema=RootSchema app=JadeSchemaLoader path=d:\jade\system
  ini=d:\jade\myjade.ini startAppParameters schemaFile=d:\jade\scm\test.scm
  ddbFile=d:\jade\scm\test.ddx dontSaveSources=true deletePropertiesIfAbsent=true
  loadStyle=onlyStructuralVersioning replayableReorg=true
  compileUnchangedMethods=true
  ```

- From your application code, by calling the **Application** class **startApplicationWithParameter** method, passing the command line arguments in a shared transient HugeStringArray; that is:

  ```
  app.startApplicationWithParameter("RootSchema", "JadeSchemaLoader",
  parameters);
  ```

  The result of the load action is returned as the last parameter in the array. A zero (**0**) value indicates no error.

  ```
  "result=error-code"
  ```

  The following is an example of the JADE code for loading the **Test** schema.

  ```
  vars
      parameters : HugeStringArray;
      proc : Process;
      pos : Integer;
      result : Integer;
  ```

```
begin
    beginTransientTransaction;
    create parameters sharedTransient;
    parameters.add("schemaFile=d:\jade\scm\test.scm");
    parameters.add("ddbFile=d:\jade\scm\test.ddx");
    commitTransientTransaction;
    proc := app.startApplicationWithParameter(rootSchema.name,
            "JadeSchemaLoader", parameters);
    // wait for the process to complete
    while app.isValidObject(proc) do
        process.sleep(1000);
    endwhile;
    // extract the result of the load from the last parameter in the list
    pos := parameters.last().pos("result=", 1);
    if pos > 0 then
        result := parameters.last()[ pos + 7 : end ].Integer;
    endif;
end;
```

For details about:

- The arguments for the **JadeSchemaLoader** application and batch **jadloadb** program, see "Loader Arguments", later in this document.
- The **Application** class **startApplicationWithParameter** method, see Chapter 1 of the *JADE Encyclopaedia of Classes*.
- Running a batch **jade** or **jadclient** program to load schemas and the **startAppParameters** and **endAppParameters** arguments, see "Running a JADE Client Application using jade.exe" or "Running a Non-GUI Client Application using jadclient", in Chapter 1 of the *JADE Runtime Application Guide*.
- Displaying and redirecting the output from JADE batch utilities, see the **DisplayApplicationMessages**, **LogServer**, and **UseLogServer** parameters under "JADE Log Section [JadeLog]", in the *JADE Initialization File Reference*.

The non-GUI **JadeSchemaLoader** application run from the **jade** or **jadclient** program outputs standard load information to **stdout** and error information to **stderr**.

Each phase of the load operation and error messages are output to the **JadeSchemaLoader.log** file. A load failure is indicated by a non-zero exit code from the process.

**Applies to Version:** 2018.0.01 and higher

## Security Considerations

The **JadeSchemaLoader** application poses a potential security risk to JADE databases.

If a JADE environment includes a web-facing application server, the **JadeSchemaLoader** application can be run by any presentation (thin) client connected to the internet. Such a client could be used to load an arbitrary schema that could include a malicious JADE application that the user could subsequently run to gain access to the JADE database.

You should enable application restrictions to prevent the unauthorized running of the **JadeSchemaLoader** application. For details, see "Controlling the JADE Thin Client Application Execution", in Chapter 3 of the *JADE Thin Client Guide*; for example:

```
[JadeAppServer]
EnableAppRestrictions=true
```

```
AllowSchemaAndApp1=MySchema,MyApp
AllowSchemaAndApp2=MySchema,MyOtherApp
```

In this example, no RootSchema applications can be run using a presentation client. Only the applications specified in the JADE initialization file **AllowSchemaAndApp<*n*>** parameters in the specified **MySchema** schema can be run by any presentation clients connecting to all application servers.

**Applies to Version:** 2018.0.01 and higher

# Loader Arguments

This section describes the **jadloadb** batch Schema Load utility and non-GUI **JadeSchemaLoader** application arguments that you can specify when running from a command script, for example.

```
[allowCircularPackages=boolean-value]
[commandFile=command-file-name]
[compileUnchangedMethods=boolean-value]
[createDeltaIfMissing=boolean-value]
[ddbFile=forms-file-name.ddb|ddx]
[deleteIfAbsent=boolean-value]
[deleteMethodsIfAbsent=boolean-value]
[deletePropertiesIfAbsent=boolean-value]
[deleteSchema=schema-name]
[delta=delta-name]
[dontSaveSources=boolean-value]
[ignoreEmptyMethods=boolean-value]
[initiateTransition=boolean-value]
[loadStyle=currentSchemaVersion|latestSchemaVersion|onlyStructuralVersioning]
[noReorgRecovery=boolean-value]
[overridePatchVersion=patch-version-number]
[reorgAllowUpdates=boolean-value]
[replayableReorg=boolean-value or default]
[reportDefnFile=report-definitions-extract-file-name.rwr]
[reportFoldersFile=report-folders-extract-file-name.rwf]
[reportFormatsFile=report-formats-extract-file-name.rwo]
[reportLoadAllFile=report-all-extract-file-name.rwa]
[reportReplaceView=boolean-value]
[reportUsersFile=report-users-extract-file-name.rwu]
[reportViewFile=report-view-extract-file-name.rwv]
[schemaFile=schema-file-name.scm]
[showProgress=boolean-value]
[suppressReorg=boolean-value]
[targetSchema=schema-name
[subschemaOf=schema-name]]
[unversionAllSchemas=boolean-value]
[unversionSchema=boolean-value]
[userName=user-name]
[waitForReorg=boolean-value]
```

**Applies to Version:** 2018.0.01 and higher

For details, see the following subsections.

## allowCircularPackages

Set the optional **allowCircularPackages** argument to **true** if you want to allow a circular dependency between packages in the schema hierarchy; that is, to permit the loading of an incomplete package (for example, **Schema1** exports **Package1** and imports **Package2**, while **Schema2** exports **Package2** and imports **Package1**). When you subsequently create a package that would result in circularities, you are prompted to confirm that you want to continue and allow a circular dependency between packages in the schema hierarchy.

The default value is **false**; that is, the packages that are available for import are those that would not result in circular dependencies in the schema hierarchy.

If you load two schemas that are not circular and you load an importer before the exporter:

- An incomplete schema is loaded.
- The form and data definition file (.**ddb** or .**ddx**) is not loaded.
- Exception 8527 (*Load results in one or more incomplete schemas*) is raised.

  When the **StandardExitValues** parameter in the [FaultHandling] section of the JADE initialization file is set to **true**, you can map this exception to generic exit value **8**. (For details about generic exit values, see "Enabling the Use of Generic Exit Values for Windows", in Appendix A of the *JADE Installation and Administration Guide*.)

Incomplete schema definitions and their usages are resolved by loading the export definition and then reloading the importing files. (To completely resolve all of the interdependencies, more than two iterations may be required.) If all of the related files are specified in a single multiple schema file (*.**mul**) load, the reprocessing of all files is done automatically.

An incomplete schema is displayed with the background color of a versioned schema (which defaults to red) in the Schema Browser. The incomplete schema, its subschemas, Class Browser, and any wizards for that schema and subschemas, are not available. To make the Class Browser available, it is your responsibility to make the entire schema complete (by loading the exporting schema and then reloading the incomplete schema).

**Note** An incomplete schema can be versioned, changing the background color, and not allowing the opening of a Class Browser on either version.

## appServer

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

## appServerPort

The **appServerPort** argument specifies the TCP/IP communications port number of the application server. This argument must be specified when you run the **JadeSchemaLoader** application from **jade.exe** as a presentation (thin) client and it must have a valid TCP/IP port value (for example, **1500**).

## commandFile

The optional **commandFile** argument enables you to specify the fully qualified name of a text file (for example, a JADE command **.jcf** file) that contains commands to create, rename, move, or delete entities, as shown in the following example.

```
commandFile=d:\temp\RenameClasses.jcf loadStyle=currentSchemaVersion
```

The command file, which has the following syntax, contains a header section, followed by the commands to be processed.

```
JadeCommandFile
JadeVersionNumber <version-number>
Commands
<commands>
<options>
```

If you require a non-default name of the JADE initialization file, you must specify this name in the **ini** argument on the command line. If you require a non-default database path, you must specify the valid absolute path in the **path** argument on the command line or in the **path** parameter in the [JadeCommandLine] section of the JADE initialization file.

The *<version-number>* value is the JADE version number (for example, **7.1.03**).

The *<commands>* value specifies a list of commands, with each command on a separate line. The following commands are supported.

```
AbortOnError [True|False]
Delete Instances schema-name::class-name
Delete Class schema-name::class-name
Rename Class schema-name::existing-class-name new-class-name
Move Class schema-name::class-name new-superclass-name
Create DbFile schema-name::dbfile-name [partitionable]
Delete DbFile schema-name::dbfile-name
Remap Class schema-name::class-name dbfile-name
Delete Interface schema-name::interface-name
Rename Interface schema-name::existing-interface-name new-interface-name
Delete Property schema-name::class-name::property-name
Rename Property schema-name::existing-class-
name::existing-property-namenew-property-name
Delete Method schema-name::class-name::method-name
Rename Constant schema-name::class-name::existing-constant-namenew-constant-name
Delete Constant schema-name::class-name::constant-name
Rename GlobalConstant
schema-name::existing-global-constant-namenew-global-constant-name
Delete GlobalConstant schema-name::global-constant-name
Delete GlobalConstantCategory schema-name::category-name
Delete ExternalFunction schema-name::external-function-name
Delete WebServiceConsumer schema-name::consumer-name
Create Locale schema-name::locale-number CopyFrom base-locale-number
Create Locale schema-name::locale-number CloneOf base-locale-number
Modify Schema schema-name DefaultLocale locale-number
Modify Schema schema-name FormsManagementStyle style-number
Rename Schema existing-schema-name new-schema-name
Delete Schema schema-name
Delete Locale schema-name::locale-number
Delete LocaleFormat schema-name::format-name
```

```
Delete Package schema-name::package-name
Delete TranslatableString schema-name::translatable-string-name
Exclude Table schema-name::rps-mapping-name::table-name
Exclude Column schema-name::rps-mapping-name::table-name::column-name
MoveInstances [Workers number]
```

For details, see the following subsections.

The *<options>* value, which is not yet supported, specifies optional arguments for processing the command file, which is not schema-specific.

If you set the **AbortOnError** command to **True**, processing of the command file is aborted when the first error is encountered when processing subsequent commands in the file.

If you set the **AbortOnError** command to **False** (the default value), subsequent commands continue to be processed regardless of errors, and a warning exit code (8514) is returned to indicate if one or more commands were not completed successfully. The default **False** value of the **AbortOnError** command applies only to errors that do not matter; for example, attempting to delete an entity such as a class, property, or method that does not exist. All other errors (for example, moving or renaming a class that does not exist) are always fatal errors.

When using a command file to create, rename, move, or delete entities during the batch loading of forms and schemas, all versioned schemas are reorganized by default at the end of a successful command file load (unless the **suppressReorg** argument is set to **true**).

---

**Note** When a transaction is committed (that is, no errors are found) and reorganization is not suppressed, a scan is made for schemas requiring reorganization. If any schema is found to require reorganization, *all* schemas are reorganized.

---

For details about:

- Using the **commandFile** argument to customize an RPS mapping for a specific site if your JADE applications are deployed to multiple sites with different requirements, see "Site-Specific RPS Mapping Customization", in Chapter 2 of the *JADE Synchronized Database Service (SDS) Administration Guide*.
- Creating a command file of property rename, deletion, and move actions when you extract a schema, see "Specifying Your Schema Options", in Chapter 10 of the *JADE Development Environment User's Guide*.
- Remapping classes and moving class instances, see "Using JCF Commands to Remap Classes and Move Class Instances", in Chapter 14 of the *JADE Developer's Reference*.

The following is an example of a JADE command file that moves class instances.

```
JadeCommandFile
JadeVersionNumber 7.1.00
Commands
Create DbFile Bats::NewFile1
Create DbFile Bats::NewFile2
Remap Class Bats::License NewFile1
Remap Class Bats::Test NewFile2
Remap Class Bats::LicenseDict NewFile1
Remap Class Bats::Company NewFile2
Remap Class Bats::TestByNameDict NewFile2
MoveInstances Workers 2
```

The following is an example of a command file that performs various actions.

```
JadeCommandFile
JadeVersionNumber 7.1.03
Commands
```

```
AbortOnError True
Delete Class ErewhonInvestmentsModelSchema::TenderSale
Rename Class ErewhonInvestmentsModelSchema::Region Area
Rename Property ErewhonInvestmentsModelSchema::Tender::offer bid
Delete Property ErewhonInvestmentsModelSchema::Country::allRegions
Delete Method ErewhonInvestmentsModelSchema::TenderSale:addNewBid
Modify Schema ErewhonInvestmentsModelSchema DefaultLocale 1033
Modify Schema ErewhonInvestmentsModelSchema FormsManagementStyle 2
Create Locale ErewhonInvestmentsModelSchema::2057 CopyFrom 1033
```

## Delete Instances Command

The **Delete Instances** command for the **commandFile** argument enables you to specify that persistent instances of the user class specified in the *class-name* parameter are deleted.

This command has the following syntax.

```
Delete Instances schema-name::class-name
```

You must specify a **loadStyle** of **currentSchemaVersion** for this command.

The target class specified in the *class-name* parameter must be local to the target schema specified in the *schema-name* parameter (that is, it cannot be a subschema-copy class), it must not be a system class (that is, a subclass of **Application**, **Global**, or **WebSession**), and it must not be a subclass of **Control**.

The target schema specified in the *schema-name* parameter must have at least one non-GUI application defined.

**Notes** Persistent instances of subclasses of the target class are not deleted.

You can delete the target class (by specifying the **Delete Class** command) in the same command file after its instances have been successfully deleted and it has no other usage; for example, property types or collection memberships.

A separate application deletes the instances in a single transaction, by executing **class.instances().purge()**. This executes destructors and performs inverse maintenance (including child reference deletions).

If a destructor relies on environmental information established by initialization or log-on methods such as transient objects referred to using the **app** system variable, the destructor may raise exceptions. In this case, you must write your own scripts to delete the target class instances.

## Delete Class Command

The **Delete Class** command for the **commandFile** argument enables you to delete the class specified in the *class-name* parameter from the schema specified in the *schema-name* parameter.

This command has the following syntax.

```
Delete Class schema-name::class-name
```

## Rename Class Command

The **Rename Class** command for the **commandFile** argument enables you to rename the class specified in the *existing-class-name* parameter to the name specified in the *new-class-name* parameter.

You must specify a **loadStyle** of **currentSchemaVersion** for this command.

This command has the following syntax.

```
Rename Class schema-name::existing-class-name new-class-name
```

## Move Class Command

The **Move Class** command for the **commandFile** argument enables you to move the class in the class hierarchy specified in the *class-name* parameter, changing its superclass to the class specified in the *new-superclass-name* parameter.

This command has the following syntax.

```
Move Class schema-name::class-name new-superclass-name
```

## Create DbFile Command

The **Create DbFile** command for the **commandFile** argument enables you to create a new **DbFile** with the specified *dbfile-name* in the schema specified in the *schema-name* parameter.

You must specify a **loadStyle** of **currentSchemaVersion** for this command.

This command has the following syntax.

```
Create DbFile schema-name::dbfile-name [partitionable]
```

Specify the optional **partitionable** parameter if the file is partition-capable. By default, this parameter is not specified, indicating that the database file cannot be partitioned when it is created. (For details, see Chapter 20 of the *JADE Developer's Reference*.)

## Delete DbFile Command

The **Delete DbFile** command for the **commandFile** argument enables you to remove the **DbFile** specified in the *dbfile-name* in the schema specified in the *schema-name* parameter, provided that the database file has no classes mapped to it.

You must specify a **loadStyle** of **currentSchemaVersion** for this command.

This command has the following syntax.

```
Delete DbFile schema-name::dbfile-name
```

## Remap Class Command

The **Remap Class** command for the **commandFile** argument enables you to change the database file mapping for the specified class *schema-name*::*class-name* to the specified *dbfile-name*.

You must specify a **loadStyle** of **currentSchemaVersion** for this command.

This command has the following syntax.

```
Remap Class schema-name::class-name dbfile-name
```

The execution of this command is deferred until the final **MoveInstances** command is processed.

For more details, see "Using JCF Commands to Remap Classes and Move Class Instances", in Chapter 14 of the *JADE Developer's Reference*.

## Delete Interface Command

The **Delete Interface** command for the **commandFile** argument enables you to delete the interface specified in the *interface-name* parameter from the schema specified in the *schema-name* parameter.

This command has the following syntax.

```
Delete Interface schema-name::interface-name
```

## Rename Interface Command

The **Rename Interface** command for the **commandFile** argument enables you to rename the interface specified in the *existing-interface-name* parameter to the name specified in the *new-interface-name* parameter.

You must specify a **loadStyle** of **currentSchemaVersion** for this command.

This command has the following syntax.

```
Rename Interface schema-name::existing-interface-name new-interface-name
```

## Delete Property Command

The **Delete Property** command for the **commandFile** argument enables you to delete the property specified in the *property-name* parameter from the class specified in the *class-name* parameter.

This command has the following syntax.

```
Delete Property schema-name::class-name::property-name
```

## Rename Property Command

The **Rename Property** command for the **commandFile** argument enables you to rename the property specified in the *existing-property-name* parameter to the name specified in the *new-property-name* parameter.

This command has the following syntax.

```
Rename Property schema-name::existing-class-name::existing-property-name
new-property-name
```

## Delete Method Command

The **Delete Method** command for the **commandFile** argument enables you to delete the method specified in the *method-name* parameter from the class specified in the *class-name* parameter in the schema specified in the *schema-name* parameter.

This command has the following syntax.

```
Delete Method schema-name::class-name::method-name
```

## Rename Constant Command

The **Rename Constant** command for the **commandFile** argument enables you to rename the class constant specified in the *existing-constant-name* parameter to the name specified in the *new-constant-name* parameter.

The source of any methods and constants that reference a renamed class constant are updated to include the new class constant name.

This command has the following syntax.

```
Rename Constant schema-name::class-name::existing-constant-name new-constant-name
```

**Applies to Version:** 2016.0.02 (Service Pack 1) and higher

## Delete Constant Command

The **Delete Constant** command for the **commandFile** argument enables you to delete the class constant specified in the *constant-name* parameter from the class specified in the *class-name* parameter in the schema specified in the *schema-name* parameter.

This command has the following syntax.

```
Delete Constant schema-name::class-name::constant-name
```

## Rename GlobalConstant Command

The **Rename GlobalConstant** command for the **commandFile** argument enables you to rename the global constant specified in the *existing-global-constant-name* parameter to the name specified in the *new-global-constant-name* parameter.

To programmatically rename one or more global constants, create a command file with the relevant commands and then use the JADE Schema Load utility to process your file.

The source of any methods and constants that reference a renamed global constant are updated to include the new global constant name.

This command has the following syntax.

```
Rename GlobalConstant schema-name::existing-global-constant-name
new-global-constant-name
```

**Applies to Version:** 2016.0.02 (Service Pack 1) and higher

## Delete GlobalConstant Command

The **Delete GlobalConstant** command for the **commandFile** argument enables you to delete the global constant specified in the *global-constant-name* parameter from the schema specified in the *schema-name* parameter.

This command has the following syntax.

```
Delete GlobalConstant schema-name::global-constant-name
```

## Delete GlobalConstantCategory Command

The **Delete GlobalConstantCategory** command for the **commandFile** argument enables you to delete the global constant category specified in the *global-constant-category* parameter for the schema specified in the *schema-name* parameter.

This command has the following syntax.

```
Delete GlobalConstantCategory schema-name::category-name
```

The category cannot be deleted if it has global constants associated with it.

## Delete ExternalFunction Command

The **Delete ExternalFunction** command for the **commandFile** argument enables you to delete the external function specified in the *external-function-name* parameter from the schema specified in the *schema-name* parameter.

This command has the following syntax.

```
Delete ExternalFunction schema-name::external-function-name
```

## Create Locale Command

The **Create Locale** command for the **commandFile** argument enables you to create the locale specified in the *locale-number* parameter in the schema specified in the *schema-name* parameter by using the:

- **CopyFrom** expression, to copy the existing base locale in the same schema specified in the *base-locale-number* parameter as the new base locale. The form and translatable string translations are copied from the existing locale to the new locale. The new locale must not be present in the schema specified in the *schema-name* parameter.

  If you specify zero (**0**) as the existing base locale, the default locale of the schema specified in the *schema-name* parameter is assumed. If you do not specify zero (**0**), you must specify an existing base locale in the **CopyFrom** expression.

- **CloneOf** expression, to create a new clone locale of the existing base locale specified in the *base-locale-number* parameter. The new locale must not be present in the schema specified in the *schema-name* parameter.

  To convert a locale from a clone locale to a base locale (or the reverse), you must first delete the clone and then create it.

The **Create Locale** command has the following syntaxes.

```
Create Locale schema-name::locale-number CopyFrom base-locale-number

Create Locale schema-name::locale-number CloneOf base-locale-number
```

## Modify Schema Command

The **Modify Schema** command for the **commandFile** argument enables you to modify the schema specified in the *schema-name* parameter by using the:

- **DefaultLocale** expression, to change the default locale of the specified schema to the existing base locale specified in the *locale-number* parameter.

- **FormsManagementStyle** expression, to change the **formsManagement** style used by the specified schema and its subschemas to the style specified in the *style-number* parameter.

  The style number can be **0** (default multiple form definition and multiple translation), **1** (single form definition and single translation), or **2** (single form definition and multiple translations using translatable strings). For details, see the **Schema** class **formsManagement** property.

  **Note** You should use the same **FormsManagementStyle** value for all user schemas in a schema branch.

You must specify a **loadStyle** of **currentSchemaVersion** for this command. In addition, the schema cannot be versioned.

The **Modify Schema** command has the following syntaxes.

```
Modify Schema schema-name DefaultLocale locale-number

Modify Schema schema-name FormsManagementStyle style-number
```

## Rename Schema Command

The **Rename Schema** command for the **commandFile** argument enables you to rename the schema specified in the *existing-schema-name* parameter to the name specified in the *new-schema-name* parameter.

This command has the following syntax.

```
Rename Schema existing-schema-name new-schema-name
```

You cannot rename the current version of a schema.

## Delete Schema

The **Delete Schema** command for the **commandFile** argument enables you to delete the schema specified in the *schema-name* parameter.

This command has the following syntax.

```
Delete Schema schema-name
```

The following example contains the fully qualified name of a JADE command **.jcf** text file that deletes a schema.

```
commandFile=d:\temp\DeleteSchema.jcf loadStyle=currentSchemaVersion
```

The **DeleteSchema** command file in the previous example deletes schema **Par9999Importer**, as follows.

```
JadeCommandFile
JadeVersionNumber 7.1.03
Commands
AbortOnError True
Delete Schema Par9999Importer
```

A schema cannot be deleted if it has subschemas, it is versioned, or it exports a package that is imported by another schema.

## Delete Locale Command

The **Delete Locale** command for the **commandFile** argument enables you to delete the locale specified in the *locale-name* parameter and any locales that are clones of it from the schema specified in the *schema-name* parameter.

This command has the following syntax.

```
Delete Locale schema-name::locale-number
```

To avoid potential invalid reference exceptions, delete locales in single user mode.

You cannot delete the schema default locale and there must always be at least one base locale.

You can delete non-present locales without causing an error.

## Delete Package Command

The **Delete Package** command for the **commandFile** argument enables you to delete an imported or exported package specified in the *package-name* parameter from the schema specified in the *schema-name* parameter.

This command has the following syntax.

```
Delete Package schema-name::package-name
```

The following example contains the fully qualified name of a JADE command **.jcf** text file that deletes a package.

```
commandFile=d:\temp\DeletePackages.jcf loadStyle=currentSchemaVersion
```

The **DeletePackages** command file in the previous example deletes imported package **P99** from the **Par9999Importer** schema, as follows.

```
JadeCommandFile
JadeVersionNumber 7.1.03
Commands
AbortOnError True
Delete Package Par9999Importer::P99
```

An exported package cannot be deleted if it is imported by another schema. An imported package cannot be deleted if it imports a class or interface that is used as the type of a property.

## Delete LocaleFormat Command

The **Delete LocaleFormat** command for the **commandFile** argument enables you to delete the locale format specified in the *format-name* parameter from the schema specified in the *schema-name* parameter.

This command has the following syntax.

```
Delete LocaleFormat schema-name::format-name
```

## Delete TranslatableString Command

The **Delete TranslatableString** command for the **commandFile** argument enables you to delete the translatable string specified in the *translatable-string-name* parameter for all locales from the schema specified in the *schema-name* parameter. This command has the following syntax.

```
Delete TranslatableString schema-name::translatable-string-name
```

## Delete WebServiceConsumer Command

The **Delete WebServiceConsumer** command for the **commandFile** argument enables you to delete the Web service consumer specified in the *consumer-name* parameter from the schema specified in the *schema-name* parameter.

This command has the following syntax.

```
Delete WebServiceConsumer schema-name::consumer-name
```

The *schema-name* value is the name of the schema in which the Web service consumer you want to delete is defined and the *consumer-name* value is the name of the consumer (that is, the entity displayed in the Web Service Consumer Browser). For example, if the schema is **S1** and the Web service consumer is **WebServiceOverHttpApp**, the command file would look like the following.

```
JadeCommandFile
JadeVersionNumber 7.1.03
Commands
AbortOnError False
Delete WebServiceConsumer S1::WebServiceOverHttpApp
```

## Exclude Table Command

The **Exclude Table** command for the **commandFile** argument enables you to exclude the table specified in the *table-name* parameter from the RPS mapping specified in the *rps-mapping-name* parameter in the schema specified in the *schema-name* parameter.

This command has the following syntax.

```
Exclude Table schema-name::rps-mapping-name::table-name
```

## Exclude Column Command

The **Exclude Column** command for the **commandFile** argument enables you to exclude the column specified in the *column-name* parameter from the table specified in the *table-name* parameter from the RPS mapping specified in the *rps-mapping-name* parameter in the schema specified in the *schema-name* parameter.

This command has the following syntax.

```
Exclude Column schema-name::rps-mapping-name::table-name::column-name
```

## MoveInstances Command

The **MoveInstances** command for the **commandFile** argument enables you to trigger the analysis of pending class remaps, the generation of an execution plan to move instances of classes to reflect the changed mappings, followed by the execution of the move instances by the database engine. You must specify a **loadStyle** of **currentSchemaVersion** for this command.

This command has the following syntax.

```
MoveInstances [Workers number]
```

The optional **Workers** parameter specifies the maximum number of worker threads that can execute file move operations concurrently. The default value is one worker.

---

**Notes** The **MoveInstances** command should always be the last command in a remap and move command file specification.

When changing the database file to which a class is mapped, the database file can be defined anywhere within the scope of the current schema or superschema hierarchy; that is, in the schema that defines the class or in any user superschema.

---

For more details, see "Using JCF Commands to Remap Classes and Move Class Instances", in Chapter 14 of the *JADE Developer's Reference*.

# compileUnchangedMethods

The optional **compileUnchangedMethods** argument enables you to specify whether you want all methods loaded from the schema file to be compiled during the load process, including those whose source is the same as their source in the database.

By default, the load process skips methods whose source is the same as their source in the database (that is, the default value is **false**). If you want to release a JADE environment that does *not* contain source code, you must perform a full schema load for all schema loads into that environment and specify the **dontSaveSources** and **compileUnchangedMethods** arguments with values of **true** on the command line.

# createDeltaIfMissing

The optional **createDeltaIfMissing** argument enables you to specify whether you want a delta created if you specify in the **delta** argument the name of a delta that does not exist.

The default value of this argument is **false**; that is, the delta is not created and the load process fails if you do not set this parameter to **true** and you specify values in the **delta** and **userName** arguments.

## ddbFile

Use the **ddbFile** argument to specify the fully qualified name of the form and data definition (.**ddb** or .**ddx**) file that you want to load, if any.

## deleteIfAbsent

The optional **deleteIfAbsent** argument enables you to specify whether you want elements (for example, classes and properties) that exist in the JADE database but are not defined in the incoming schema file to be deleted.

You can also use the **deleteMethodsIfAbsent** and **deletePropertiesIfAbsent** arguments to provide you with more-detailed control over the deletion of missing methods and properties.

**Note** An application in the database that is used in a package is not deleted (over-written) when a complete schema definition containing that application is loaded.

By default, undefined elements are deleted.

## deleteMethodsIfAbsent

The optional **deleteMethodsIfAbsent** argument enables you to specify whether you want to delete methods that exist in the JADE database but are not defined in the incoming schema file.

The value of this argument defaults to the value of the **deleteIfAbsent** argument. For example, if you want to delete the methods in the JADE database that are not in your schema file but you want to retain existing interfaces, classes, and properties that are not in the incoming file, specify:

```
schemaFile=d:\jade\scm\test.scm ddbFile=d:\jade\scm\test.ddb deleteIfAbsent=false
deleteMethodsIfAbsent=true
```

## deletePropertiesIfAbsent

The optional **deletePropertiesIfAbsent** argument enables you to specify whether you want to delete properties that exist in the JADE database but are not defined in the incoming schema file.

The value of this argument defaults to the value of the **deleteIfAbsent** argument. For example, if you want to delete the classes, interfaces, and methods in the JADE database that are not in your schema file but you want to retain existing properties that are not in the incoming file, specify:

```
schemaFile=d:\jade\scm\test.scm ddbFile=d:\jade\scm\test.ddb
deletePropertiesIfAbsent=false deleteIfAbsent=true
```

## deleteSchema

The optional **deleteSchema** argument enables you to specify the name of a user-defined schema or the **JadeReportWriterSchema** that you want to delete from a runtime database, as shown in the following example.

```
deleteSchema=DbAdminTest
```

When you delete a schema, all of its class instances are also deleted.

You can remove user-defined schemas and the **JadeReportWriterSchema** only. You cannot remove a system schema or a user-defined schema that has one or more subschemas.

To remove a schema that has subschemas, you must first remove any subschemas of that schema.

**Notes** As this argument is case-sensitive, you must specify the schema name exactly. For example, **deleteSchema=ErewhonInvestmentsViewSchema** is valid but if you were to specify **deleteSchema=Erewhoninvestmentviewschema**, an error would be output.

You can specify only one user-defined schema for deletion. If you want to remove more than one schema (for example, subschemas and then their superschema), you must run separate copies of the **jadloadb** program. In addition, you cannot delete a schema in a **jadloadb** execution that loads a schema (that is, you must specify either the **deleteSchema** argument or the **schemaFile** argument, but not both), a user-defined schema that is in reorganization state, or a schema that has exported packages that are in use by other schemas.

When loading schemas, you cannot:

- Load a schema in a **jadloadb** execution and remove a schema (that is, you must specify either **schemaFile** argument or the **deleteSchema** argument, but not both).
- Delete a user-defined schema if the schema is in reorganization state or if the schema has exported packages that are in use by other schemas.

If classes in the schema have destructors that delete dependent objects, errors may be raised if the schema deletion has already deleted those objects.

For details about deleting a schema from the JADE development environment, see "Removing a User-Defined Schema", in Chapter 3 of your *JADE Development Environment User's Guide*.

## delta

The optional **delta** argument enables you to specify the delta into which loaded methods will be checked out. You must also specify a value in the **userName** argument.

When you specify the **delta** and **userName** arguments and the specified delta does not exist, the load process fails if you have not set the **createDeltaIfMissing** arguments to **true**.

## dontSaveSources

The optional **dontSaveSources** argument enables you to specify that you want to delete the source code from all JADE methods when the schema is loaded. By default, source code is retained when the schema is loaded.

If you want to release a JADE environment that does *not* contain source code, you must perform a full schema load for all schema loads into that environment and specify the **dontSaveSources** and the **compileUnchangedMethods** arguments with values of **true** on the command line.

When the methods that are being loaded already exist and the source of those methods has not changed, the methods are not compiled. The source is not updated so it is therefore not deleted.

For details about including recompiled methods in patch versioning when a patch is to be applied to a schema that does not have source available, see "Enabling or Disabling Patch Versioning", in Chapter 3 of your *JADE Development Environment Administration Guide*.

## ignoreEmptyMethods

The optional **ignoreEmptyMethods** argument enables you to override the default behavior when an empty method implementation is found while loading a schema file. The default behavior removes the existing source for the method and marks it as needing compilation (that is, if you do not specify the **ignoreEmptyMethods** argument, it is assumed to be **false**).

Set this argument to **true** when you want to leave the existing source and code of a method with an empty implementation untouched during a schema load.

The expected use of this argument occurs when an extracted schema that includes methods that had their source stripped is loaded into an environment where those methods had previously been loaded from an encrypted schema.

**Caution** Overriding the default behavior by using the **ignoreEmptyMethods** argument is considered potentially unsafe.

## initiateTransition

The optional **initiateTransition** argument enables you to override the default behavior when the transition phase of a reorganization is reached.

By default, the transition phase of the reorganization is attempted as soon as the object conversion phase is complete. For details, see "Initiating Transition" under "Reorganization Options", in Chapter 14 of your *JADE Developer's Reference*.

Set this argument to **false** if you want to suspend the reorganization and explicitly initiate the transition at a later stage.

## loadStyle

The optional **loadStyle** argument enables you to specify the load style, as follows.

- The default **latestSchemaVersion** value loads a new schema or loads the schema as the latest schema version (a new version will be created if required), and allows structural changes.
- The **currentSchemaVersion** value loads the schema into the existing current schema version, which may potentially affect current runtime behavior. As structural changes are not allowed, the load will not proceed if structural changes are to be introduced.

  If you are loading a schema file containing only method changes into a multiuser system, select this option, to ensure that no structural changes are attempted that would require a reorganization of the database and impact existing users. If a structural change is detected, the schema load returns an error.

  The commands that can be performed only in the current schema version are:

  - Create DbFile
  - Delete DbFile
  - Modify Schema
  - Delete Schema
  - MoveInstances
  - Remap Class
  - Delete Instances
  - Rename Class
  - Rename Interface

  For details about these commands, see "commandFile", earlier in this chapter.

- The **onlyStructuralVersioning** value loads only structural changes into the latest version and does not version methods or other non-structural entities. This may potentially affect current runtime behavior.

**Note** As all JADE Report Writer files (that is, files with a suffix of **rwv**, **rwo**, **rwf**, **rwu**, **rwr**, and **rwa**) are loaded only into the current schema version, they should therefore be loaded after any database reorganization transition.

## noReorgRecovery

The optional **noReorgRecovery** argument enables you to disallow the creation of temporary backups (**.bak** files) of the original database files (**.dat** files) 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 **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 parameter 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.

## overridePatchVersion

The optional **overridePatchVersion** argument enables you to override the patch version number in schema entities by specifying that all loaded entities have their patch version number set to the specified value, which can be in the range **1** through **Max_Integer - 1** (that is, 2,147,483,646).

When you specify a valid value, it is effective regardless of whether patch versioning is enabled for the schema. A value of zero (**0**) causes the **overridePatchVersion** argument to be ignored. The **overridePatchVersion** argument is not required if you want to retain the patch number specified in the schema file.

The specified value does not have to match an existing defined patch number. If the value is the same as an existing patch number, the state of the existing patch version is ignored (that is, the existing patch number in the schema file may be closed).

## reorgAllowUpdates

The optional **reorgAllowUpdates** argument enables you to specify whether you want to allow other users to continue development and update the database while the reorganization is in progress when you are loading schemas in multiuser mode. For details, see "Allowing Updates" under "Reorganization Options", in Chapter 14 of your *JADE Developer's Reference*.

In multiuser mode, the **reorgAllowUpdates** argument is set to **true** by default, to allow updates to proceed before the transition is initiated; for example, other users can still modify (that is, edit and compile) methods in the development environment.

## replayableReorg

The optional **replayableReorg** argument defaults to the value of the **EnableArchivalRecovery** parameter in the [PersistentDb] section of the JADE initialization file, and is constrained by the setting of the **EnableArchivalRecovery** parameter for the database.

Set the value of the **replayableReorg** argument to **false** if you do not want to perform a replayable reorganization of your JADE database when the schema is loaded. If you set the argument to **true** or you let it use the default value, the argument is set to the value of the database **EnableArchivalRecovery** parameter.

For details, see "Replayable Reorganizations" under "Reorganization Options", in Chapter 14 of your *JADE Developer's Reference*.

## reportDefnFile

The optional **reportDefnFile** argument enables you to specify the fully qualified name of a JADE Report Writer report definitions unload (extract) file that you want to load, as shown in the following example.

```
reportDefnFile=d:\reports\SalesReports.rwr
```

For details about extracting report definitions, see the*JADE Report Writer User's Guide*.

## reportFoldersFile

The optional **reportFoldersFile** argument enables you to specify the fully qualified name of a JADE Report Writer report folders unload (extract) file that you want to load, as shown in the following example.

```
reportFoldersFile=d:\reports\DevFolders.rwf
```

For details about extracting report folders, see the *JADE Report Writer User's Guide*.

## reportFormatsFile

The optional **reportFormatsFile** argument enables you to specify the fully qualified name of a JADE Report Writer formats unload (extract) file that you want to load, as shown in the following example.

```
reportFormatsFile=d:\reports\AcctsFormats.rwo
```

For details about extracting report formats, see the *JADE Report Writer User's Guide*.

## reportLoadAllFile

The optional **reportLoadAllFile** argument enables you to specify the fully qualified name of a single unload (extract) file that contains all JADE Report Writer view, folder, system option, user, and report definitions that you want to load, as shown in the following example.

```
reportLoadAllFile=d:\jade\rpts\alldata.rwa reportReplaceView=true
```

Specify the optional **reportReplaceView** argument with a value of **true** if you want the JADE Report Writer view file being loaded to replace any existing view with the same name. When you specify **reportReplaceView=true**, view items that are not in the incoming view extract file and are not used in any existing report definitions are removed. Those items that are used in existing reports are retained.

By default, if a reporting view with the same name already exists, the views are merged.

For details about extracting all report writer data, see the *JADE Report Writer User's Guide*.

## reportReplaceView

The optional **reportReplaceView** argument enables you to specify whether the JADE Report Writer view file being loaded should replace any existing view with the same name, as shown in the following example.

```
reportViewFile=d:\reports\TestViews.rwv reportReplaceView=true
```

If you specify this argument with a value of **true** and a reporting view with the same name already exists, view items (types, features, root collections, joins, and script methods) that are not in the incoming view extract file and are not used in any existing report definitions are removed. Those items that are used in existing reports are retained. By default, if a reporting view with the same name already exists, the views are merged.

You can use the **reportReplaceView** argument with both the **reportViewFile** and **reportLoadAllFile** arguments, as they both contain views. If you specify the **reportReplaceView** argument with one of the other report-related parameters (for example, the **reportFoldersFile** argument), it is ignored as they do not contain view files.

For details about extracting reporting views, see the *JADE Report Writer User's Guide*.

## reportUsersFile

The optional **reportUsersFile** argument enables you to specify the fully qualified name of a JADE Report Writer report users unload (extract) file that you want to load, as shown in the following example.

```
reportUsersFile=d:\reports\MgmtUsers.rwu
```

For details about extracting report users, see the *JADE Report Writer User's Guide*.

## reportViewFile

The optional **reportViewFile** argument enables you to specify the fully qualified name of a JADE Report Writer reporting view unload (extract) file that you want to load, as shown in the following example.

```
reportViewFile=d:\reports\TestViews.rwv reportReplaceView=true
```

Specify the optional **reportReplaceView** argument with a value of **true** if you want the JADE Report Writer view file being loaded to replace any existing view with the same name. When you specify **reportReplaceView=true**, view items that are not in the incoming view extract file and are not used in any existing report definitions are removed. Those items that are used in existing reports are retained.

By default, if a reporting view with the same name already exists, the views are merged.

For details about extracting reporting views, see the *JADE Report Writer User's Guide*.

## schemaFile

Use the **schemaFile** argument to specify the fully qualified name of the schema that you want to load.

If you are loading a partial file, this may be a .**scm** file or a .**cls** file for a class or interface extract. This can also be a multiple schema file (.**mul**) for loading multiple schemas.

---

**Note** When deleting a user-defined schema (by using the optional **deleteSchema** argument) or loading a report view, formats, folders, users, definitions, or all report data file (by using the optional **reportViewFile**, **reportFormatsFile**, **reportFoldersFile**, **reportUsersFile**, **reportDefnFile**, or **reportLoadAllFile** argument), you do not have to specify the otherwise mandatory **schemaFile**. However, you must always specify the **path** argument.

---

## showProgress

The optional **showProgress** argument enables you to specify that you want to suppress the progress (status) message display resulting from compiling schema files (for example, class number conflict messages) and from any reorganization, by setting this argument to **false**.

When using a command file to create, rename, move, or delete entities during the batch loading of forms and schemas, all versioned schemas are reorganized by default at the end of a successful command file load (unless the **suppressReorg** argument is set to **true**).

**Note** When a transaction is committed (that is, no errors are found) and reorganization is not suppressed, a scan is made for schemas requiring reorganization. If any schema is found to require reorganization, *all* schemas are reorganized.

Suppressing progress dialogs is useful when loading from a script or command file.

By default, progress messages are displayed; that is, the default value is **true**.

**Note** Full details of the load operation are still output to the appropriate log file.

## subschemaOf

If you specify the name of a new schema by using the optional **targetSchema** argument, you must specify its superschema by using the **subschemaOf** argument.

This argument is not required if you specify the name of an existing schema for the **targetSchema** argument.

## suppressReorg

The optional **suppressReorg** argument enables you to specify if you want to prevent the reorganization of any loaded schemas that require reorganization on completion of the load process.

By default, any loaded schemas that require reorganization on completion of the load process are reorganized, regardless of whether they required reorganization prior to the load; that is, the default value is **false**.

**Notes** If control classes require reorganization or new **Control** subclasses are defined before loading a forms definition file, a reorganization will be initiated before the forms definition file is loaded. In this case, you must load the Relational Population Service (RPS) .**ddbx** file, if any, before the reorganization, to keep the RPS mappings up-to-date with the schema changes. (When extracting a schema with an RPS mapping and the DDX format style is selected, a separate .**ddbx** file is produced as for a DDB extract but the file contents are in XML format.)

After the reorganization, the .**ddb** or .**ddx** form and data definition file is loaded as normal, to complete the changes.

If such a reorganization is required and you check this control, the forms load process will raise an exception.

## targetSchema

The optional **targetSchema** argument enables you to specify a different target schema name from the names defined in the schema file.

You can specify a new name for your schema or the name of an existing schema if you want to load into a schema other than the one defined in the schema file. This argument has no effect if you are loading multiple schemas.

## unversionAllSchemas

The optional **unversionAllSchemas** argument enables you to specify that you want to remove the latest version of all versioned schemas, by setting this argument to **true**.

## unversionSchema

The optional **unversionSchema** argument enables you to specify that you want to remove the latest version of a versioned schema (and any related schema) specified in the **schemaFile** argument when a new version is loaded, by setting this argument to **true**.

## userName

The optional **userName** argument enables you to specify the user name to associate with the delta. You must also specify a value in the **delta** argument.

When you specify the **delta** and **userName** arguments and the specified delta does not exist, the load process fails if you have not set the **createDeltaIfMissing** argument to **true**.

## waitForReorg

The optional **waitForReorg** argument enables you to specify that you do not want to wait for a reorganization to complete, by setting the value to **false**.

As the load process in single user mode must wait for any reorganization to complete, this argument is set to **true** by default if the value of the **server** argument is **singleUser**.

The **waitForReorg** argument is set to **true** by default.