JACK Intelligent Agents® Development Environment Manual

Copyright

US Government Restricted Rights

Trademarks

Publisher Information

Agent Oriented Software Pty. Ltd.
P.O. Box 639,
Carlton South, Victoria, 3053
AUSTRALIA

Phone: +61 3 9349 5055
Fax: +61 3 9349 5088
Web: http://www.agent-software.com

If you find any errors in this document or would like to suggest improvements, please let us know.

The JACK™ documentation set includes the following manuals and practicals:

Document	Description
Agent Manual	Describes the JACK programming language and infrastructure. JACK can be used to develop applications involving BDI agents.
Teams Manual	Describes the JACK Teams programming language extensions. JACK Teams can be used to develop applications that involve coordinated activity among teams of agents.
Development Environment Manual	Describes how to use the JACK Development Environment (JDE). The JDE is a graphical development environment that can be used to develop JACK agent and team-based applications.
JACOB Manual	Describes how to use JACOB. JACOB is an object modelling language that can be used for inter-process transport and object initialisation.
WebBot Manual	Describes how to use the JACK WebBot to develop JACK enabled web applications.
Design Tool Manual	Describes how to use the Design Tool to design and build an application within the JACK Development Environment.
Graphical Plan Editor Manual	Describes how to use the Graphical Plan Editor to develop graphical plans within the JACK Development Environment.
JACK Sim Manual	Describes how to use the JACK Sim framework for building and running repeatable agent simulations.
Tracing and Logging Manual	Describes the tracing and logging tools available with JACK.
Agent Practicals	A set of practicals designed to introduce the basic concepts involved in JACK programming.
Teams Practicals	A set of practicals designed to introduce the basic concepts involved in Teams programming.

Table of Contents

1 Introduction

1.1 JACK Development Environment (JDE)

The *JACK™ Development Environment (JDE)* is a cross-platform graphical editor suite written entirely in Java for developing JACK™ agent and team based applications. Extensive use of drag-and-drop and context-sensitive menus assist the construction of agents. The JDE allows the definition of projects, aggregate agents and teams, and their component parts under these projects.

The JDE is a purpose-built toolkit that facilitates the construction of agent/team models. In many situations an application will consist of a single model. However, the JDE also supports the co-operative development of the models required for an application. It also supports the reuse of model components and in the case of co-operative development, the sharing of components.

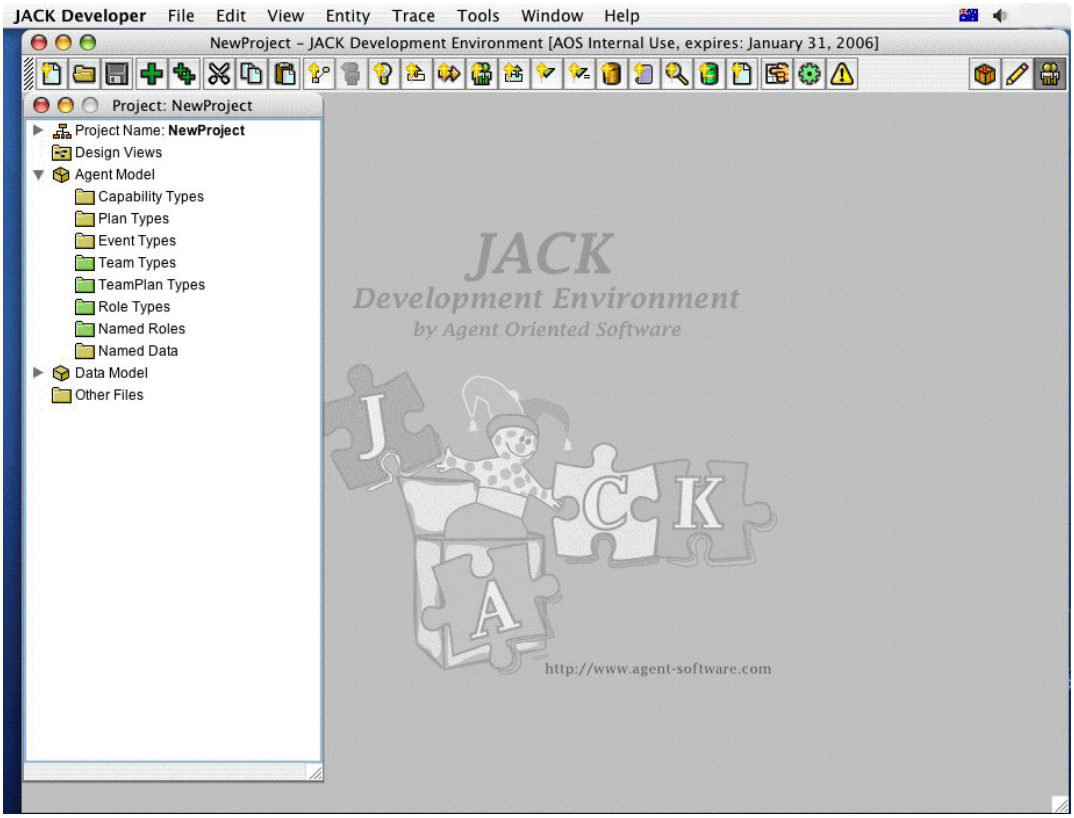

Figure 1-1: JDE

1.2 Readership

This manual is designed to complement the *Agent Manual*. It is assumed that the reader has an understanding of agent programming using the JACK framework.

Integral features of the JDE include the *Design Tool*, the *Graphical Plan Editor* and the *Plan Tracing Tool*, all accessible from within the JDE browser and tool bar. These features each have their own individual documentation in which they are more fully detailed. This documentation is available as part of the complete *JACK™ Intelligent Agents* (JACK) package.

1.3 Opening the JDE

There are two common ways to invoke the JDE:

- running the default configuration
- running a custom configuration.

On Unix platforms, running the default configuration is achieved by running the script, `JACK_Developer`. This is found in the directory in which the JDE has been installed. On Unix, the script can also be run with the name of a project file e.g. `JACK_Developer NewProject.prj`

On Windows platforms, selecting JACK Developer on the Start menu invokes the default configuration.

On Macintosh platforms, double-click on the JACK Developer icon. This is found in the directory in which the JDE is installed.

To start the JDE from the command line, type the following (where `path_to_jack.jar` is the path to the `jack.jar` file): `java -classpath path_to_jack.jar -Xmx85m aos.main.Jack`

If an out of memory error appears, the upper limit of memory available to the Java Virtual Machine may need to be increased. For example, to increase the amount of memory available to 120 megabytes, replace `-Xmx85m` with `-Xmx120m` in the above command line.

1.4 Layout of this manual

After this introductory section, an overview provides a brief sketch of the JDE, covering its functions, the interface elements, and the standard JDE interaction protocol. Subsequent sections delve into the JDE's features in greater depth.

1.5 Typographical conventions

Certain typographical conventions have been adopted in this manual:

- In the narrative sections of this manual, *italics* are used to denote text that would benefit from special emphasis (for example, new terms, mandatory instructions, and critical concepts).
- Samples of code and program output appear in `this typeface`.
- Items that appear in the JDE graphical user interface (GUI) appear in this typeface, for example: Select the Open Project option in the File menu. GUI items are capitalised in the text in an identical manner to their on-screen counterparts.
- Keys that the user must press on their computer keyboard are represented in THIS TYPEFACE, for example: Press the ESCAPE key.

1.6 Platform-specific GUI differences

In keeping with current GUI design practice, the JDE adopts the look and feel of the host operating system. Within this constraint, the JDE has been designed to be as cross-platform compatible as possible. For example, if the user is familiar with the JDE on a Windows platform, they will find that it is much the same on a Unix platform.

The screen snapshots in this manual were created on a Macintosh running Apple's Mac OS X operating system. On another platform, the JDE GUI will look slightly different, but these differences are only cosmetic, and will not affect the performance of the JDE in any way. In some cases, menu options may be found in different locations.

There are also some differences in operating system-specific conventions regarding mouse buttons and modifier keys. These are listed and explained below. Windows is the most common platforms for JDE users, so this document adopts Windows conventions as the default when describing actions within the JDE framework.

In this document, textual content refers to the JDE on a Windows platform. Where there are variations in menus, options and so on across platforms, a note will be provided explaining those differences.

1.6.1 Mouse buttons

Unix systems typically have a three button mouse, Windows systems have a two button mouse, and Macintosh systems have a single button mouse.

In the JDE, the left and right mouse buttons function in the same way on Unix and Windows systems. The middle button is unassigned, and not used for any purpose.

To access the functions of the right mouse button on a Macintosh system, hold down the COMMAND key while pressing the mouse button (unless a third-party two button mouse is available allowing the use of the right button as required).

1.6.2 Modifier keys

Generally, where a CTRL key is used in Windows and Unix, the COMMAND key is used on the Macintosh. For example, on Windows you use the CTRL key to make multiple selections. On the Macintosh you would use the COMMAND key. The SHIFT key performs the same function on all three platforms.

1.7 Further information

Useful information and updates relevant to the JDE are available at the Agent Oriented Software web site, `http://www.agent-software.com`.

2 Overview

JACK™ Intelligent Agents supports two agent reasoning models:

- a basic agent model
- a teams agent model.

These models are implemented as extensions of the Java language. These extensions:

- provide new base classes, interfaces and methods
- extend Java syntax to support the new classes, definitions and statements
- provide semantic extensions that change the execution engine to support a BDI execution model.

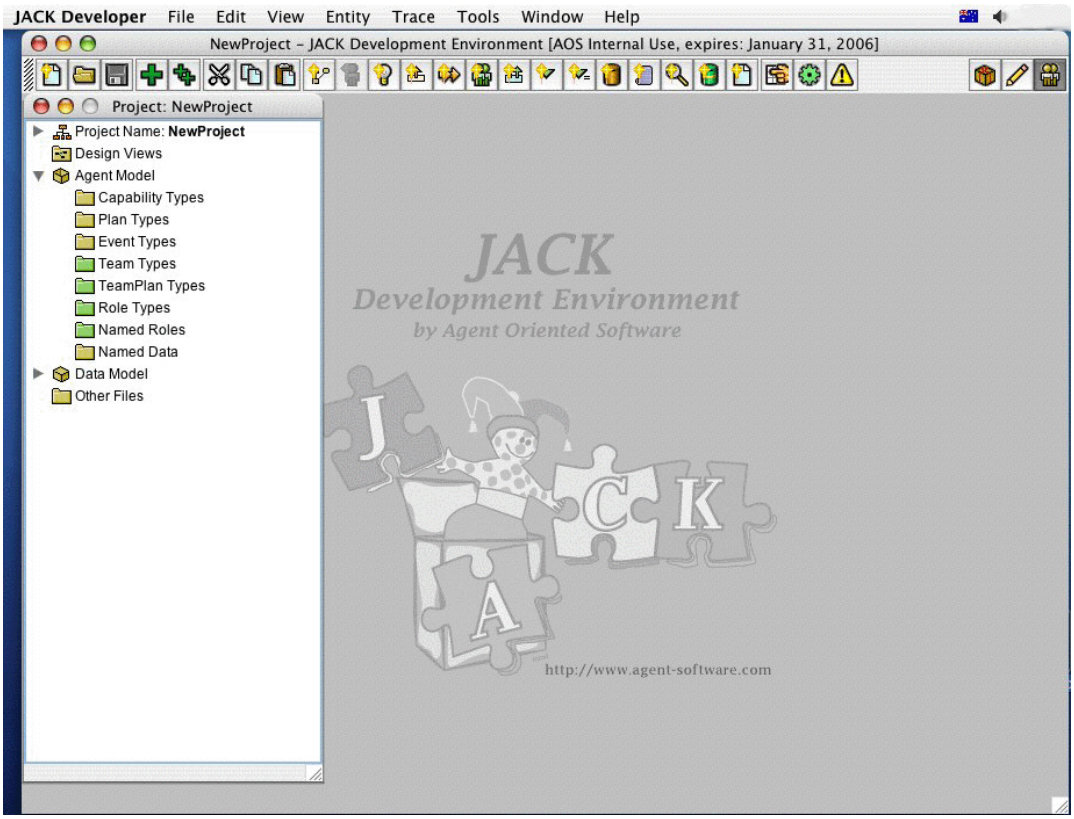

Figure 2-1: JDE menu bar and work area

The class level constructs provided by the two models are discussed in detail in the *Agent Manual* and *Teams Manual*. These constructs are the primary building blocks of every JACK application and can be defined in any text editor. For some, this is the preferred method of program development. However, others prefer to use graphical environments for code

development. The JDE provides support for this mode of development through conventional drag-and-drop functionality and context sensitive menus.

The JDE also supports aspects of the software development life cycle other than code generation and execution. In particular, tools are provided to support the design process and the tracing of design and plan execution. These tools are fully integrated with the basic code generation/compilation/execution functionality of the JDE. The Design Tool, the Graphical Plan Editor and the Plan Tracing Tool are each explained separately within their own documents.

When the JDE is invoked, a window is created which initially consists of a menu bar and a work area.

Within the JDE a project consists of a collection of programming elements. Through the JDE menu bar, one can create a new project or open an existing project. When this occurs, a special instance of a JDE browser window is opened. This is known as the *project window*.

Directly below the JDE menu bar is the JDE Tool Bar. The JDE Tool Bar provides a series of shortcut buttons that can be clicked to bypass the drop-down menus of the menu bar for many of the basic functions of the JDE. The buttons available to the left perform various actions in the JDE, and are referred to as *action buttons*.

There are also three buttons to the right end of the JDE Tool Bar. These buttons toggle various modes on and off, and are referred to as *toggle buttons*.

Further details about these buttons and their functions are provided in the *JDE Tool Bar* chapter of this document.

The project window provides the primary interface for code development including access to the Graphical Plan Editor. It also enables the user to create and access design views for the JACK Agent Model or JACK Teams Model.

The project window is configured differently depending on whether one is developing an agent model or a teams model – the appropriate mode is selected via the View menu from the JDE menu bar. The JDE menu bar also provides access to the Compiler Utility which enables the user to compile and execute agent and teams models.

2.1 JDE menu bar

The JDE menu bar contains the following drop-down menus:

- JACK Developer menu (Macintosh only)
- File menu
- Edit menu

- View menu
- Entity menu
- Trace menu
- Tools menu
- Window menu
- Help menu.

These menus provide access to the high level functions of the JDE, such as loading and saving projects, and editing projects in various ways. These menus are outlined briefly below, and are explained in greater detail elsewhere in the *Menu Bar* chapter of this document.

2.1.1 JACK Developer menu (Macintosh only)

The JACK Developer menu exists only in the Macintosh environment. The JACK Developer menu contains the following options:

- About JACK Developer
- Preferences
- Services
- Hide JACK Developer
- Hide Others
- Show All
- Quit JACK Developer.

2.1.2 File menu

The File menu contains the following options:

- New Project
- Open Project
- Save Project
- Load External Edits
- Copy/Move Project
- Convert Old Project
- Close Project
- Update JACK Files
- Generate All JACK Files

- Remove JACK Files
- Generate Project Report...
- Page Setup...
- Print Project Report...
- Exit (non-Macintosh platforms only).

These options are discussed in detail in the *JDE menu bar* chapter of this document.

Note: To the left of the File menu on the Macintosh appears the JACK Developer menu. The Quit function is listed within this menu on the Macintosh rather than in the File menu.

2.1.3 Edit menu

The Edit menu contains the following options:

- Cut Text
- Copy Text
- Paste Text.

2.1.4 View menu

The View menu contains the following options:

- Full Package Path
- Documentation Nodes
- Teams Mode
- Show Tool Bar.

These options can all be toggled on or off providing alternatives affecting the various functions of the JDE. Checks indicate which of the options is currently toggled on.

2.1.5 Entity menu

The Entity menu contains the following options:

- Add Design...
- Add Agent...
- Add Capability...
- Add Plan...
- Add Event...

- Add Team...
- Add TeamPlan...
- Add Role...
- Add Named Role...
- Add Named Data...
- Add Beliefset...
- Add View...
- Add TeamData...
- Add External Class....

Note: Some of the above menu options are only enabled in Teams mode.

2.1.6 Trace menu

The Trace menu contains the following options:

- Connect To Nameserver...
- Connect To Portal...
- Configure Design Tracing...
- Design Tracing Controller
- Ping Agent....

2.1.7 Tools menu

The Tools menu contains the following options:

- Compiler Utility
- Design Palette
- Plan Editor Palette
- Error Log
- Preferences... (non-Macintosh platforms only).

Note: On the Macintosh, the Preferences option appears in the JACK Developer menu.

2.1.8 Window menu

The Window menu maintains a list of titles of all currently opened (not minimised) JDE browser windows. Initially, the Project window will be listed here. Selecting a title form the list will bring a particular window to the front of the JDE.

2.1.9 Help menu

The Help menu contains the following options:

- JACK Documentation
- JACK Frequently Asked Questions
- Agent Oriented Software Homepage
- Create Project from Example
- About JACK Intelligent Agents(non-Macintosh platforms only).

Note: On the Macintosh, the About JACK Intelligent Agents option is named About JACK Developer and is located in the JACK Developer menu.

2.2 Project window

Within a JDE session only one project can be active at a time. When a project is first opened, a special JDE browser window is opened which is known as the *project window.*

Additional JDE browser windows can be opened by selecting items from the project window, and dragging them onto the JDE work area. For example, if a subtree of the project window's folder hierarchy is dragged and dropped into the JDE work area, a new JDE browser window is opened for browsing the selected subtree.

Note: Additional JDE browser windows can be opened, but there can only be one project window.

2.2.1 Opening the project window

The project window is opened when an existing project is loaded or a new project is created. This is achieved via the File menu on the JDE menu bar. The project window displays the name of the current project in the title bar. It also displays a hierarchical tree structure of folders which contain the project components. The window may be moved around the JDE window and resized according to the needs of the user.

2.2.2 Closing the project window

The project can be closed via the File menu on the JDE menu bar. Note that closing the project window also closes the project. If a project has unsaved changes, a prompt will appear asking whether the changes should or should not be saved. A project can be explicitly saved via the File menu at any time during a JDE session.

Saving a project via the File menu causes any entities that have been changed (highlighted in red in the project window) to be saved. The project file is updated to include any references to newly created or loaded files and any new or updated entity files are saved. A project is stored as a file in JACOB syntax with a `.prj` extension. To learn more about JACOB, refer to the *JACOB Manual*. When saving a project, a prompt may appear asking the user to create all required folders necessary for that project.

2.2.3 Features of the project window

The project window contains project details (such as Project Name and Documentation) and the Design Views folder. It also holds the Agent Model container and the Data Model container, both of which contain folders for storing the various programming elements of the project. Finally, the project window also contains the Other Files folder, enumerating non-JACK files of interest.

These folders allow grouping of entities in a project. For large projects, some folders will contain many entities. In such situations, it is useful to group the entities within a folder into sub-folders. This is achieved through the use of nested containers. Containers can be nested within containers, without limit to the number of nesting levels. Entities of the same type can be moved between nested containers.

The elements in the project window's folder hierarchy can be expanded or contracted by clicking on the '+' symbol or arrow to the left of the element or by double-clicking on the element's folder or name. When an element is expanded, the '+' symbol becomes a '-' symbol, or the right arrow becomes a down-arrow on the Macintosh. Clicking on the '-' symbol or down-arrow (or, equivalently, double-clicking on the element next to the '-' symbol or down-arrow) collapses the expanded tree.

Context-sensitive menus are provided for performing operations on folders and folder elements. The menus are accessed by placing the mouse cursor over an item and pressing the right mouse button. A menu then appears providing operations that can be applied to that item.

Drag-and-drop is used to associate one entity with another (such as a plan with an agent). The element to be dragged is first selected by clicking and holding the left mouse button down while over the icon. The left button is then kept down while the mouse is moved until the cursor is over the destination item. Releasing the left mouse button will then associate the dragged element with the destination element. Multiple drag-and-drop can also be performed as described in the *JDE Browser* chapter.

Note that if a drop operation is permitted, the Drop Allowed cursor is displayed.

Figure 2-2: Drop Allowed cursor

If the operation is not permitted, the No Drop Allowed cursor is displayed.

Figure 2-3: No Drop Allowed cursor

2.2.3.1 Project details

A project as a whole encapsulates all of the resources related to an agent model and any extra (non-JACK) files such as regular Java files and data files required.

The project details are the first item in the project window. It has the following seven attributes:

Attribute	Description
`Project Name`	The name of the project e.g. NewProject.
`Documentation`	Documentation associated with the project.
`Root Package`	The top-level Java package to which all project components belong.
`Save-folder`	This is where the G-Code is stored. G-Code is the graphical file format used by the JDE. The folder is called `gcode_ProjectName by default.`
`Generated JACK Sub-folder`	This is the sub-folder (of the folder where the project file is located) where the generated JACK code will be stored.
`Generated Java Sub-folder`	This is where the JACK compiler stores the Java files generated while compiling the JACK files.
`Generated Class Sub-folder`	This is where the java compiler stores the class files generated from the Java files.

Table 2-1: Project attributes

The type definition files created within the JDE are stored as G-Code files in the `Save` folder. Compiling an application results firstly in JACK files being generated from the G-Code files. The JACK files are written to the Generated JACK Sub-folder. The files in the Generated JACK Sub-folder are then translated to Java code; these files are stored in the Generated Java Sub-folder. Finally, the generated Java code is compiled along with any other Java source files needed. The resulting class files are stored in the Generated Class Sub-folder. Note that the G-Code to JACK code step can be performed independent of the other steps if desired.

Applications that have been developed outside of the JDE can be imported into the JDE. This can be done through the Compiler Utility. Alternately, the G-Code and a project file can be generated by entering the following command in the directory which is the root of the application source tree:

```
java aos.main.JackBuild -r -x -E myproject.prj -dj backup
```

`myproject` can then be opened in the normal way from within the JDE.

The `-dj` option results in the application files being copied to a directory called `backup`. If this option is omitted, and the resulting G-Code was compiled with the default setting for the Generated JACK Sub-folder, the original JACK files would be overwritten.

The user can also import components from other projects into the current project. This is discussed in the *JDE Browser* chapter of this document.

2.2.3.2 Design Views folder

It is possible to design and build an application using the Design Tool. The Design Tool can be accessed by right-clicking on the Design Views folder. Existing designs can be accessed from within the Design Views folder. Entities that exist in the JDE browser can be incorporated into a design by dragging them onto the design canvas. Any new entities or links between entities that are created while building a design will result in corresponding changes to the contents of the JDE browser window.

The Design Tool is discussed in greater detail in the *Design Tool Manual*.

2.2.3.3 Agent Model container

The following folders are contained within the Agent Model container:

- Agent Types folder

  Agents are added to the Agent Types folder in the Agent Model container. They are further elaborated by dragging plans, events, named data and capabilities from the other folders and dropping them onto the appropriate target folders within the Agent Types folder.

- Capability Types folder

Capabilities are added to the Capability Types folder of the current project and are further elaborated by dragging plans, events, named data and other capabilities from the folders below and dropping them onto the appropriate target folders within the Capability Types folder.

- Plan Types folder

  Plans are added to the Plan Types folder in the Agent Model container. They are further elaborated by dragging events and named data from the folders below and dropping them onto the appropriate target folders within the Plan Types folder.

  Plan reasoning methods can be edited using a text editor or the Graphical Plan Editor. The Graphical Plan Editor is discussed in more detail in the *Graphical Plan Editor Manual*.

  If the plan reasoning methods were created using the Graphical Plan Editor, the Plan Tracing Tool can be used to display and trace the execution of Graphical Plans in JACK applications.

  For further information on the Plan Tracing Tool, refer to the *Plan Tracing Tool Technical Brief*.

- Event Types folder

  Events are added to the Event Types folder in the Agent Model container. They are further elaborated by dragging named data from the other folders and dropping them onto the appropriate target folders within the Event Types folder.

- Named Data folder

  Named Data are added to the Named Data folder in the Agent Model container. The Named Data folder contains named data that can be used by JACK components. The type of the named data is any data model element.

- Team Types folder

  Teams are added to the Team Types folder in the Agent Model container. They are further elaborated by dragging plans, events, capabilities, team plans and named data from the other folders and dropping them onto the appropriate target folders within the Team Types folder. The Team Types folder is only visible within the JDE browser when the JDE is in Teams Mode (when the Teams Mode option in the View menu is checked).

- TeamPlan Types folder

  Teamplans are added to the TeamPlan Types folder in the Agent Model container. They are further elaborated by dragging events, roles, named data and named roles from the other folders and dropping them onto the appropriate target folders within the TeamPlan Types folder. The TeamPlan Types folder is only visible within the JDE browser when the JDE is in Teams Mode (when the Teams Mode option in the View menu is checked).

- Role Types folder

  Roles are added to the Role Types folder in the Agent Model container. They are further elaborated by dragging events and named data from the other folders and dropping them onto the appropriate target folders within the Role Types folder. The Role Types folder is only visible within the JDE browser when the JDE is in Teams Mode (when the Teams Mode option in the View menu is checked).

- Named Roles folder

  The Named Roles folder contains named roles of a type defined in the Role Types folder.

Of these folders, Team Types, TeamPlan Types, Role Types and Named Roles are available only when the Teams mode option is checked in the View menu.

When the JDE is in Teams Mode, the Agent Types folder may be included or excluded (as per your preference settings).

2.2.3.4 Data Model container

The following folders are located within the Data Model container:

- Beliefset Types folder

  Beliefsets are added to the Beliefset Types folder in the Data Model container. They may be further elaborated by dragging events from the other folders and dropping them onto the appropriate target folders within the Beliefset Types folder.

- View Types folder

  Views are added to the View Types folder in the Data Model container. They may be further elaborated by dragging named data from the other folders and dropping them onto the appropriate target folders within the View Types folder.

- TeamData Types folder

  TeamData are added to the TeamData Types folder in the Data Model container. The TeamData Types folder is only visible in the Data Model container when the JDE is in Teams Mode (when the Teams Mode option in the View menu is checked).

- External Classes folder

  This provides the ability to identify Java classes that are external to the project. The External Classes folder is used to provide a type for named data.

2.2.3.5 Other Files folder

Other files associated with the project are listed here. These files include regular Java files and data files of interest within the project.

2.3 Code Editor

The Code Editor appears whenever a text item is edited, unless an external editor is used. Using an external editor is discussed in the *Preferences* section.

The kinds of items that are edited with the Code Editor include:

- documentation
- Java code (contained in the Other Code folder of an entity)
- reasoning methods (only textual, not graphically edited reasoning methods)
- posting methods
- external files (contained in the Other Files folder).

The Code Editor has a number of features designed to facilitate writing JACK code. It has a tool bar located in the top-left corner of the Code Editor window. The functions of the Code Editor are discussed in further detail in the *Code Editor* chapter of this document.

2.3.1 Opening the Code Editor

To open the Code Editor:

1. Right-click on a text item and choose the appropriate menu item from the context menu. For example, a reasoning method or any documentation of a programming element (pencil icon). Alternatively, use a double left-click on the element.
2. The Code Editor will open, either as the default Code Editor supplied with the JDE package, or the external editor specified in the user's Preferences options.

2.3.2 Closing the Code Editor

2.3.2.1 Saving changes internally

When editing an internal JDE text item (i.e. not an external file), the Code Editor window has two buttons to the right: Save and Close.

- the Save button stores the user's work but leaves the window open so that work can be continued
- the Close button closes the window without storing work (after providing a confirmation prompt).

2.3.2.2 Saving changes to external files

Saving changes to external files is discussed in detail in the *Code Editor* chapter of this document.

2.4 Using an External Editor

The JDE allows the user to specify an alternative to the Code Editor. This is discussed in the *Preferences* section of this document.

2.5 Compiler Utility

JACK-based applications can be compiled and run from within the JDE with the Compiler Utility. It provides a graphical user interface to the standard command-line based `aos.main.JackBuild` utility provided with JACK. Refer to the relevant section of the *Agent Manual* for further details.

2.5.1 Opening the Compiler Utility

Open the Compiler Utility at any time by selecting Compiler Utility from the Tools menu on the JDE menu bar, or by clicking on the Compiler Utility icon in the tool bar. If the Compiler Utility is already open, its window will be brought to the front.

Figure 2-4: Compiler Utility icon

2.5.2 Closing the Compiler Utility

To close the window at any time, click the Close button.

2.5.3 Features of the Compiler Utility

The Compiler Utility window consists of the tabs listed below. Further details are provided in the *Compiler Utility* chapter of this document.

- Options tab

  Use this tab to set options used in compiling JACK code with the Compiler Utility.

- Compile application tab

  Use this tab to convert an application from source code into an executable form.

- Run application tab

  Use this tab to execute a compiled application (or any other Java application).

- Convert Non-JDE JACK tab

  Use this option tab to generate a new JDE project from existing JACK code which was not developed using the JDE.

- Output/Errors tab

  This tab displays the output from compiling or running an application as well as any output associated with the generation of G-Code.

3 JDE menu bar

3.1 Main menus

The JDE menu bar contains the following menus:

- JACK Developer menu (Macintosh only)
- File menu
- Edit menu
- View menu
- Entity menu
- Trace menu
- Tools menu
- Window menu
- Help menu.

The JDE menu bar, located at the top of the JDE work area, provides access to major functions, as outlined below.

3.2 JACK Developer menu (Macintosh only)

The JACK Developer menu is used to perform application specific operations on the Mac OS X platform. The JACK Developer menu contains the following options:

- About JACK Developer
- Preferences...
- Services
- Hide JACK Developer
- Hide Others
- Show All
- Quit JACK Developer

The options from the above list that are specific to the JDE are discussed in detail below. The remaining options are standard for all Macintosh software.

3.3 About JACK Developer

The About JACK Developer option provides information about Agent Oriented Software Pty. Ltd. It also provides information about the particular release version of the JDE and JACK being used, and further copyright information.

3.4 Preferences...

The Preferences option is located in the JACK Developer menu on the Mac OS X platform, whereas in Windows it is located in the Tools menu. More information on Preferences is provided below.

3.5 Quit JACK Developer

This option quits the JACK Developer.

3.6 File menu

The File menu is used to perform project-level operations and to exit the JDE. The File menu contains the following options:

- New Project
- Open Project...
- Save Project
- Load External Edits
- Copy/Move Project...
- Convert Old Project
- Close Project
- Update JACK Files
- Generate All JACK Files
- Remove JACK Files
- Generate Project Report...
- Page Setup...
- Print Project Report...
- Exit (Windows only).

These options are discussed in detail below.

3.6.1 New Project

The New Project option creates a new project. By default, the location of this project is the directory in which the editor is invoked.

To create a new project:

1. Click on the File menu in the JDE menu bar.
2. Click on the New Project option on the File menu. A file chooser window will appear.
3. Navigate through the files and directories until the appropriate place for the new project is located.
4. Enter a name for the new project in the File name text box and click New, or click Cancel to cancel the operation.

3.6.2 Open Project...

The Open Project option opens an existing JDE project file (`.prj`).

To open an existing project:

1. Click on the File menu in the JDE menu bar. The File menu will open, providing a series of options.
2. Click on the Open Project option in the File menu.
3. A user directory window will pop up. Navigate through the home directory structure until the desired project file is located.
4. Click on the desired project file. It will become highlighted and will be visible in the File name text box.
5. Click the Load button to load the project, or click Cancel to cancel the operation.

3.6.3 Save Project

The Save Project option saves all project components that have been modified or are unsaved (highlighted in red). If the component being saved already exists on the file system, the existing version is copied to a backup file with the extension `.bak` and the new version is saved in its place.

To save a project:

1. Click on the File menu in the JDE menu bar.
2. Select Save Project from the various options within the File menu.
3. Any unsaved changes will be saved, changing from being highlighted text in the JDE browser to black default colour text (not highlighted).

Note: When saving a project, a prompt may appear asking the user whether to create all required folders necessary for the project.

3.6.4 Load External Edits

The Load External Edits option will cause any text which has been changed by an external editor and saved to its temporary file, to be loaded back into the appropriate JDE programming element.

The Load External Edits option will be inactive unless an external editor has been selected in the preferences. It will also be inactive if the preference to Load changes from an external editor as soon as the file is saved/written is set since it makes this option redundant.

3.6.5 Copy/Move Project...

The Copy/Move Project option moves the project to a different directory, or makes a copy of the current project.

To copy/move a project:

1. Select Copy/Move Project from the File menu.
2. The following options will appear: Cancel, Copy or Move.
   - To cancel the operation, click Cancel.
   - To copy, select Copy and then select a new directory for the copy of the project.
   - To move, select Move and select a new directory.

3.6.6 Convert Old Project

If a project is being opened that was saved with an old version of the JDE, this option will load everything and update it to the new internal format. This is useful if you know ahead of time that the old project needs to be re-formatted.

3.6.7 Close Project

The Close Project option closes the current project. If the current project has any unsaved components, an option is provided in a pop-up dialog box to save them before closing.

To close a project:

1. Click on the File menu in the JDE menu bar.
2. Click on the Close Project option in the File menu.
3. If no unsaved changes exist, the project will close.

4. If unsaved changes exist within the current (open) project, a warning pop-up menu will appear, providing the options to either Save, Discard Changes or Cancel.
5. Select the appropriate option to save new changes, discard unsaved changes or cancel the operation altogether and keep the current project open.

Note: When closing a project, a prompt may appear asking the user whether to create all required folders necessary for the project.

3.6.8 Update JACK Files

This will generate JACK code for any elements that require it; for example, JACK files for new elements or elements that have been modified (highlighted in red).

To update JACK files:

1. Click on the Update JACK Files option in the File menu.
2. The necessary JACK code will be generated, and a progress dialog will inform the user that the process is occurring.
3. Any errors or warnings will be displayed in the Error Log.

3.6.9 Generate All JACK Files

The Generate All JACK Files option generates JACK code for all elements in the current project, regardless of whether they are up-to-date or not. In other words, the Generate All JACK Files option is a complete generation and updating option.

To generate all JACK files:

1. Click on the File menu in the JDE menu bar.
2. Click on the Generate All JACK Files option in the File menu.
3. A progress dialog will inform the user that code is being generated.
4. Any errors or warnings will be displayed in the Error Log.

3.6.10 Remove JACK Files

The Remove JACK Files option in the File menu removes any generated JACK files from the currently active project. The Remove JACK Files option actually renames old generated files to become `*.bak` files.

3.6.11 Generate Project Report

The Generate Project Report option in the File menu generates a project report containing the documentation and details of the selected project components.

3.6.12 Page Setup

The Page Setup option in the File menu provides various options relating to printing JDE projects. Depending on the platform, it provides options on paper size and source, orientation (landscape or portrait), and margin widths.

3.6.13 Print Project Report

The Print Project Report option in the File menu opens the Print Project Report window. This window allows the user to select the print format (text or XML). The various parts of the report that the user wishes to print can be selected individually with a variety of checkboxes, or all parts of the report can be printed at once.

3.6.14 Exit (Windows only)

The Exit option exits the JDE and provides a prompt to save any unsaved changes before exiting.

To exit the JDE:

1. Click on the File menu in the JDE menu bar.
2. Click the Exit option in the File menu.
3. If there are no unsaved changes in the current project, the project will close and the JDE will close down.
4. If unsaved changes exist, a pop-up prompt will appear, offering the options to either Save, Discard Changes or Cancel. Select the required option and unsaved changes will be saved or discarded and the JDE will shut down, or the operation will be cancelled accordingly.

3.7 Edit menu

This menu provides the following options for text in the editor windows:

- Cut Text
- Copy Text
- Paste Text.

3.7.1 Cut Text

The Cut Text option cuts highlighted text from a text editor window.

To cut text:

1. Highlight the required text in the text editor window with the mouse.
2. Click on the Cut Text option in the Edit menu.

3. Alternatively, cut text and transfer it onto the clipboard using the keyboard shortcut CTRL+X, or COMMAND+X on the Macintosh.

3.7.2 Copy Text

This option copies highlighted text from a text editor window to the clipboard.

To copy text:

1. Highlight the relevant text in the text editor window with the mouse.
2. Click on the Copy Text option in the Edit menu.
3. Alternatively, copy the text using the keyboard shortcut CTRL+C, or COMMAND+C on the Macintosh.

3.7.3 Paste Text

This option pastes text from the clipboard into the current text editor window at the current insertion point. Text that has been copied or cut can be pasted from the clipboard. Text must be copied or cut before the Paste Text option can be used.

To paste text:

1. Click with the mouse at the desired insertion point to create a cursor mark where the copied or cut text is to be pasted.
2. Click on the Paste Text option in the Edit menu.
3. Alternatively, press CTRL+V (or COMMAND+V on the Macintosh) to paste the text using a keyboard shortcut.

3.8 View menu

The View menu on the JDE menu bar provides the following options:

- Full Package Path
- Documentation Nodes
- Teams Mode
- Show Tool Bar.

Check marks to the left of each option indicate whether that option has been selected or not.

3.8.1 Full Package Path

When the Full Package Path option is checked, the entire package path of the current project is displayed on elements in the project.

3.8.2 Documentation Nodes

When the Documentation Nodes option is checked, the user is able to view and edit documentation for types and files, including the current project.

3.8.3 Teams Mode

By clicking on the Teams Mode option in the View menu, the user can toggle between Teams mode and Agent mode. The results of toggling between the two modes are directly visible in both the JDE browser and the Design Palette.

The folders which are displayed within the JDE browser are mode dependent. In Agent mode the following folders can be accessed: Agent Types, Capability Types, Plan Types, Event Types, Named Data, Beliefset Types, View Types and External Classes.

In Teams mode the following folders can be accessed: Capability Types, Plan Types, Event Types, Team Types, TeamPlan Types, Role Types, Named Roles, Named Data, Beliefset Types, View Types, TeamData Types and External Classes.

Likewise, the contents of the Design Palette are mode dependent. Agent mode provides the following icons: Agent, Capability, Event, Plan, Named Data and Note while Teams mode provides all the icons of Agent mode, as well as Team, TeamPlan, Named Role and Role. The Agent icon is greyed out if the Agent Types folder is opted to be excluded in Preferences; otherwise it is functional.

3.8.4 Show Tool Bar

When this option in the View menu is checked, the JDE Tool Bar is visible. To hide the JDE Tool Bar, simply uncheck this option by clicking it.

3.9 Entity menu

The Entity menu provides the following options:

- Add Design...
- Add Agent...
- Add Capability...
- Add Plan...
- Add Event...
- Add Team...
- Add TeamPlan...
- Add Role...

- Add Named Role...
- Add Named Data...
- Add Beliefset...
- Add View...
- Add TeamData...
- Add External Class....

The options listed above all provide shortcuts that add an object of the selected type.

3.10 Trace menu

The Trace menu provides the following options:

- Connect To Nameserver...
- Connect To Portal...
- Configure Design Tracing...
- Design Tracing Controller
- Ping Agent...

3.10.1 Connect To Nameserver...

Use this menu option to connect to a DCI name server. For example, if the application had been started using the DCI arguments described in the previous section, the user would enter `localhost:9999` or `9999` to connect to the application's nameserver. Name server connection is optional; however, without a nameserver, the Design Tracing Tool user needs to know explicit portal addresses in order to connect the Design Tracing Tool to each portal, rather than using portal names.

Note that the nameserver address can also be entered using the Connect To Portal... menu option.

Refer to the *Design Tracing Tool* chapter of the *Tracing and Logging Manual* for further information on this option.

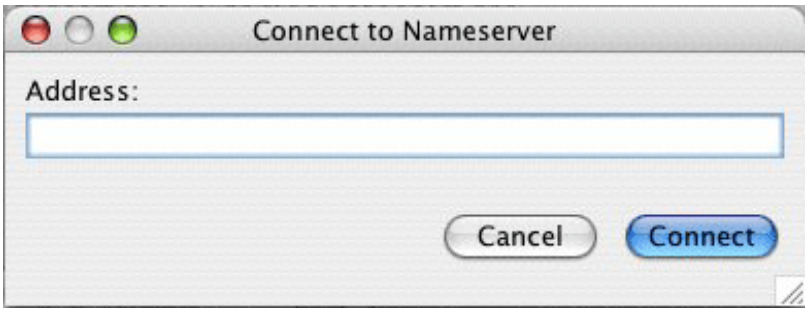

Figure 3-1: Connect to Nameserver dialog

3.10.2 Connect To Portal...

Use this menu option to connect to a portal via the portal name or address. Once connected, the Tracing window will be displayed, showing all agents connected to that portal.

If already connected to a nameserver, you would normally connect by specifying a portal name. Otherwise, use an explicit address such as `localhost:9999` or `9999`.

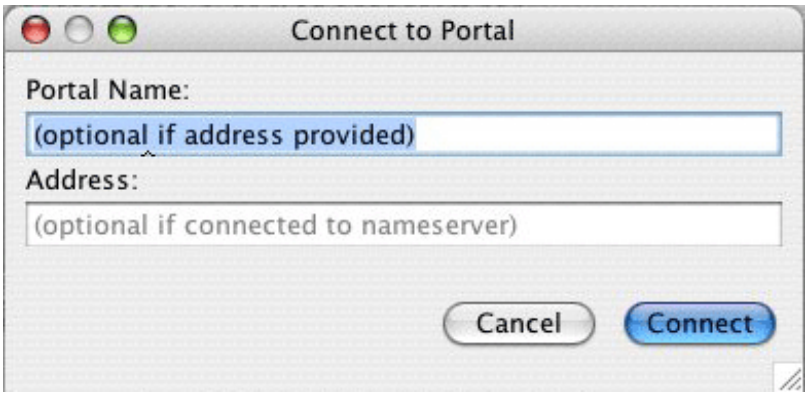

Figure 3-2: Connect to Portal dialog

Once connected to a portal an application can be resumed from its suspension point and traced with the DTT.

3.10.3 Configure Design Tracing...

Use this menu option to configure settings for design tracing. Design tracing should be configured whilst an application is stopped or paused.

Designs can be traced for specific agent types and/or agent names. An asterisk can be used as a wildcard to indicate matching all agent types or agent names.

The Design Tracing Configuration window lists the following:

- Project:Design Name
- Agent Type
- Agent Name
- Trace Group.

Agents can be added or removed in the Design Tracing Configuration window by clicking the Add or Remove options to the right of the window, and the options to ApplyLoad...Save... or Close are available at the bottom of the screen. Refer to the *Tracing and Logging* Manual for more details on design tracing.

3.10.4 Design Tracing Controller

Use this menu option to control tracing of designs. Refer to the *Tracing and Logging* Manual for more details on design tracing.

3.10.5 Ping Agent...

Use this menu option to check if an agent is still responding. The number of attempts and the delay in seconds can be set between ping attempts. The agent name and portal name need to be entered, e.g. `robot23@Server`.

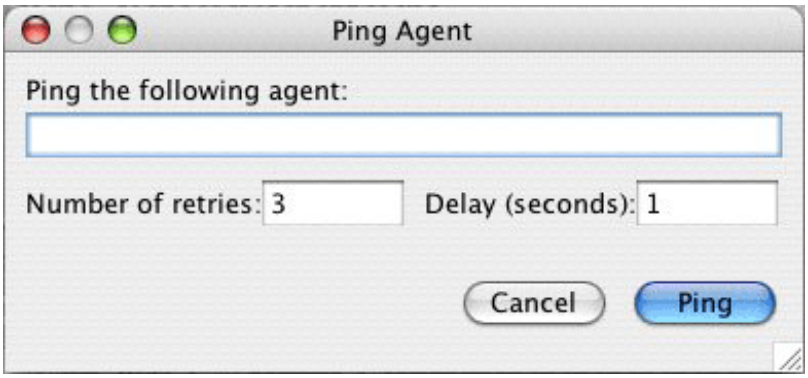

Figure 3-3: Ping Agent dialog

3.11 Tools menu

The Tools menu provides the following options:

- Compiler Utility
- Design Palette
- Plan Editor Palette

- Error Log
- Preferences...

3.11.1 Compiler Utility

The Compiler Utility and its various features and functionalities are discussed separately in the *Compiler Utility* chapter of this document.

3.11.2 Design Palette

The Design Palette provides access to the various elements of a design. It has two modes – Agent mode and Teams mode, and the user can toggle between these modes by clicking on the Teams Mode option in the View menu.

Figure 3-4: Design Palette – Agent mode

Figure 3-5: Design Palette – Teams mode

The icons on the Design Palette can be dragged and dropped from the palette onto a Design View canvas by clicking, dragging and releasing the icon with the mouse. In this way, designs can be built quickly and easily with the Design Tool, the JDE's graphical user interface (GUI), for creating project designs.

The Design Palette, its modes and its uses are described in greater detail in the *Design Tool Manual*.

3.11.3 Plan Editor Palette

The Plan Editor Palette provides access to the various reasoning method statements and components used within the Graphical Plan Editor environment. Icons can be dragged from the palette and dropped onto the Plan Graph Editor canvas, from which point they are used to build and develop graphical reasoning methods.

Note: The Plan Editor Palette has more options when running in Teams Mode.

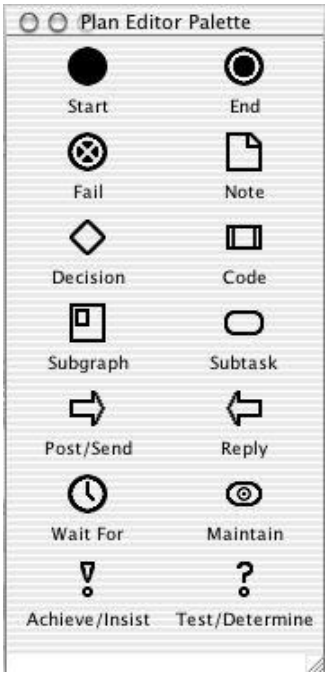

Figure 3-6: Plan Editor Palette

The Plan Editor Palette and its functions are described in greater detail in the *Graphical Plan Editor Manual*.

3.11.4 Error Log

The Error Log shows errors and warnings related to JACK components, e.g. a plan does not handle any event type. It shows errors and warnings generated during JACK file generation.

The Clear Log button at the base of the Error Log window clears the text within the Error Log window.

Figure 3-7: Error Log

3.11.5 Preferences

The Preferences option allows the specification of JDE preferences. Various JDE configuration parameters can be set with the Preferences option in the Edit menu.

Clicking on the Preferences option in the Edit menu opens a window with a series of tabs providing various preference options.

The Preferences options window can also be opened using the Preferences shortcut in the JDE Tool Bar.

Figure 3-8: Preferences icon

3.11.5.1 Preferences tabs

The Preferences tabs are:

- Project View tab
- Text Editor tab
- Design Tool tab
- Graphical Plans tab
- Fonts tab
- Advanced tab.

The various options within each tab window are discussed in detail below.

Tab windows in the Preferences option are opened by clicking on the relevant window tab at the top of the Preferences window. Only one tab window is accessible at a time, but all tabs are visible to the user at any given time.

To select a preference in any of the tab windows:

1. Click on the check box to the left of the desired option.
2. Click the Apply Changes button at the base of the window, followed by the Close button to close the Preferences window.
3. The relevant preference is now selected.

To save preferences in the configuration file `.jack.properties` in the user's home area (system dependent), click on Apply and Save Preferences. They will be used next time the

JDE is started up. After Apply and Save Preferences has been clicked, the Close button must be clicked to close the Preferences window.

If the Close button is clicked instead of Apply and Save Preferences, default preferences will be used until the JDE is exited.

3.11.5.2 Project View tab

The Project View tab provides the various options listed and explained below. To select an option, click on the appropriate checkbox. The options available on the Project View tab are as follows:

- Prefix all top-level components by their Java package name

  If this option is checked, each top-level component will be displayed with its package prefix.

- Pop up dialog when creating new objects

  If this option is checked, the user will be presented with a dialog box when defining a new element. If it is not checked, the new element is added to the JDE browser directly.

- Use external windows for popups and dialogs

  If this option is checked, all popups and dialogs will appear in their own external windows, separate from the JDE work area.

- Automatically update references when a component is changed or deleted

  This option automatically updates references when a component is changed or deleted, instead of providing a pop-up prompting the user.

- Remove restrictions on top-level structure in project browser

  Restrictions are introduced to avoid accidental changes to the overall structure. This option exists in order to be able to turn off these restrictions. If this option is checked, the user can remove top-level structures in the project browser. Top-level structures are the Agent Model container and the Data Model container. Selecting this preference also adds an extra option to the context-sensitive menus for these structures. For example, Remove "Agent Model" Container.

- Sorting:

  The Sorting: option includes several drop-down selections that can be chosen. These are:

  – When sorting a container, keep sub-containers at the top

  – When sorting a container, keep sub-containers at the bottom

  – Treat sub-containers the same as other entities.

- Use teaming mode by default

When the Use teaming mode by default option is checked, the JDE automatically starts in the Teams Mode, without the need to separately select the Teams Mode option in the View menu.

- In teaming mode, show Agent types (Agent types must extend Team)

  When selected, this option ensures that Agent Types are shown in Teams Mode. When programming with JACK Teams, the basic agent type is actually the Team so the Agent type is not usually needed in this mode.

- Check the amount of available memory at startup

  If this checkbox is ticked, the user will be warned if the JDE is started with less than the recommended amount of memory (this can only occur if the JDE is run from the command line).

- Show main Tool Bar by default

  If this checkbox is ticked, the JDE Tool Bar will be visible to the user by default.

3.11.5.3 Text Editor tab

The JDE allows you to use your own choice of text editor for viewing and/or modifying textual components such as code blocks and documentation. There are a number of preferences that can be set under the Text Editor tab of the Preferences dialog.

The options available on the Text Editor tab are as follows:

- Show line numbers in the internal editor
- Confirm changes made via "Edit As JACK File" (0=always, 0.5=usually, 5=never)

  After making changes to an entity with Edit As JACK File, the JDE examines how much of the text has changed. The JDE notices how many chunks of text have been added, removed or changed and combines these numbers into a single metric that ranges roughly from 0 to 5. The higher the number, the larger the degree of change being represented. This preference simply sets the threshold required before confirmation will be requested in order to load the changes back into the JDE. The default value of 0.5 requests confirmation after most changes except the most trivial. The numbers given in the label are just a guide.

- Use an external application for editing text (e.g. "Notepad", "open -e" or "gvim -f")

  This option is used to specify an external editor (if any) to use in place of the JDE's inbuilt Code Editor. The text box should contain the command line required to invoke the external editor. If the text box is left blank (or the given external editor cannot be located), the JDE will use its inbuilt Code Editor as a default.

- Prepend the following text to the temporary file created when using an external editor

  This option enables the user to prepend specified text to any temporary filename created for an external editor. This option defaults to JACK.

- Append the following text to the temporary file created when using an external editor

  This option enables the user to append specified text to any temporary filename created for an external editor. This option defaults to `.jack`. This is very useful for editors that can recognise syntax based on the filename extension.

The following additional checkboxes are also available within the Text Editor tab:

- Load changes from an external editor as soon as the file is saved/written

  When this option is checked, the JDE automatically loads changes made from an external editor. If unchecked, the user can load the changes manually by using the Load External Edits option on the File menu. The default for this checkbox is *off*.

- The external editor detaches itself so don't try to keep track of the external process

  The external editing works best when the JDE can keep track of the external editor process. Some editors detach themselves and run in background. They may seem to exit immediately (which the JDE will detect) or they may seem to never exit (which won't be detected). On Windows, using Notepad or Wordpad as the external editor can have this effect. In this case it is recommended that you select this preference and also the Popup a control window for each external editor (checkbox) preference as described below. If the detaching is detected, the JDE will automatically set this preference for the current session only. Some editors have an option to run in the foreground (e.g. `gvim -f`) or background.

  This option lets the JDE know to expect that the external editor will detach its process from the JDE. This means that the JDE will not try to track when the process is finished and therefore will not attempt to make checks that are normally made when the editor exits. Note that the JDE will still automatically load changes whether the external process detaches or not (if the appropriate preference is set as mentioned above). The default for this checkbox is *off*.

Note: External editors will run in their own window and will not belong to the JDE desktop. Consequently, external editors will often seem to disappear because most JDE actions will bring the quite large JDE desktop to the front of the screen and hide any external editors.

- Popup a control window for each external editor

  Changes made by an external editor can be automatically loaded into the JDE as described above. If the user wants more control over their external editor however, they can choose to show the control window for each editor session.

  For files other than Other Files in the JDE, the control window has at most three buttons: Edit..., Load Changes and Close.

  – The Edit... button re-opens the external editor window. On some systems, or with certain editors, this will bring the current editor window (if open) to the front of the screen, while on others it will attempt to start the external editor again.

– The Load Changes button will only be available if the user has not checked the preference Load changes from an external text editor as soon as the file is saved/ written. This button will load any changes made by the external editor - saving them back into the JDE.

– The Close option will load any saved changes and close the control window.

When editing Other Files in the JDE, the control window has four buttons: Edit..., Commit, Revert... and Close.

– The Edit... button re-opens the external editor window. On some systems, or with certain editors, this will bring the current editor window (if open) to the front of the screen, while on others it will attempt to start the external editor again.

– The Commit button will save any changes to the file.

– The Revert... button allows the user to revert to the last saved state of the edited file, or the original file at the beginning of this edit session.

– The Close option will load any saved changes and close the control window.

3.11.5.4 Design Tool tab

The various options available on the Design Tool tab all relate directly to the Design Tool, and not to the JDE as a whole.

To select an option, simply click on the textbox associated with the desired option. The options available within the Design Tool tab are as follows:

- Show design bar on all design windows

  When this option is checked, the design bar is visible on all design windows within the Design Tool, enabling all tool bar functionalities to be within easy reach of the user. However, if this option is selected it will not apply to designs that are already open.

- Show labels on design bar/palette

  When this option is checked, the name labels on each of the icons on the design bar and palette are visible. However, this does not apply to designs that are currently open.

- Ask for confirmation on deletion of links in your design

  When this option is checked, the JDE provides a confirmation window that checks that the user wishes to delete links in an open design before they are deleted. This provides a safeguard, to ensure that links are not deleted in error.

- Show tick marks in the zoom slider

  This option enables the tick marks on the Zoom slider to be viewed. However, depending upon the degree of magnification of a given design, tick marks may not be helpful, and can be removed from view with this option to provide a cleaner view.

This option is not applied to designs that are already open in the JDE.

Figure 3-9: Zoom slider

- Show zoom slider

  The Zoom slider can be made visible, or can be hidden from view by toggling this option on or off. This option is not applied to designs that are already open in the JDE.

- Show editable zoom selection list

  This option displays a drop-down list of magnification percentage options to the left of the Descriptive Mode On/Off button in the tool palette. It should be noted that when this option is selected it is not applied to Design Tool windows that are already open.

- Design tool is initially in descriptive mode

  When this option is checked, the Design Tool's descriptive mode is selected as a default.

- Always return to selection mode immediately after drawing a link

  This option creates Selection mode as a default rather than as an option of equal value with Link mode, so that after drawing a link the Design Tool returns to Selection mode, ready to enable the next link to be selected.

For further details on the Design Tool and the functions provided by these tabs, please refer to the *Design Tool Manual*.

3.11.5.5 Graphical Plans tab

The various options available on the Graphical Plans tab all relate to functions provided within the Graphical Plan Editor environment, including the Plan Graph Editor. To select an option, simply click on the textbox associated with the desired option. The options available within the Graphical Plans tab are as follows:

- Generate traceable plans

  The Generate traceable plans option creates plans with traceable reasoning methods when JACK files are generated for a project.

  Note that only graphical reasoning methods may be traced graphically. However, the exit and entry points of textual reasoning methods can be traced.

- Plan Graph code generator warns about missing links in graphs

  When JACK files are generated, any warnings about missing links are shown in the Error Log. For example, a warning will appear if a Code node has no incoming links and thus is not able to be executed.

- Show labels on plan editor palette

  This option displays the names of node types (beneath the node icon) that are located in the Plan Editor Palette. This option is particularly useful when first becoming acquainted with the Graphical Plan Editor, and can be toggled off when the user becomes more familiar with the various node types.

- Show tick marks in the zoom slider

  This option enables the tick marks in the Zoom slider to be displayed or hidden from view, according to the user's needs. The tick marks indicate the extent to which a plan is increased or decreased in size.

- Show zoom slider

  The entire Zoom slider can be visible or hidden from view depending on whether or not the Show zoom slider checkbox is checked.

- Show editable zoom selection list

  This option displays a drop-down list of magnification percentage options to the left of the Descriptive Mode On/Off button in the tool palette. It should be noted that when this option is selected it is not applied to Plan Graph Editor windows that are already open.

- Plan graph editor is initially in descriptive mode

  This option declares descriptive mode as the initial mode for the Plan Graph Editor within the JDE when the Plan Graph Editor is opened.

- Always return to selection mode immediately after drawing a link

  This option declares selection mode as the default mode when the Plan Graph Editor is open. If this checkbox is checked, the Plan Graph Editor will return to Selection mode after a link has been drawn.

For further details on the Graphical Plan Editor, and the functions provided by the options within the Graphical Plans tab, please refer to the *Graphical Plan Editor Manual*.

3.11.5.6 Fonts tab

The Fonts tab provides textboxes with the options discussed below:

- Font for tree labels in browser (requires save and restart)

  This option provides a means by which different fonts can be selected for use on the tree labels within the JDE browser. A project must be saved and the JDE closed and restarted in order for this to take effect.

- Font for tree values in browser (requires save and restart)

  This option provides a means by the font for tree values in the JDE browser can be changed according to user needs. A project must be saved and the JDE closed and restarted in order for this to take effect.

- Font for object name text in design diagrams (close and reopen opened designs)

  This option provides a means by which different fonts can be selected for use in the object name text in design diagrams. To make a newly entered font visible, the user must close and reopen any diagrams that are currently open.

- Font for link text in design diagrams (close and reopen opened designs)

  This option provides a means by which different fonts can be selected for use in the link text in design diagrams. To make a newly entered font visible, the user must close and reopen any diagrams that are currently open.

- Font for object type text in graphical plans (close and reopen opened graphs)

  This option provides a means by which different fonts can be selected for use in the object type text in graphical plans. To make a newly entered font visible, the user must close and reopen any graphs that are currently open.

- Font for object code text in graphical plans (close and reopen opened graphs)

  This option provides a means by which different fonts can be selected for use in the object code text in graphical plans. To make a newly entered font visible, the user must close and reopen any graphs that are currently open.

- Font for linked text in graphical plans (close and reopen opened graphs)

  This option provides a means by which different fonts can be selected for use in the linked text in graphical plans. To make a newly entered font visible, the user must close and reopen any graphs that are currently open.

3.11.5.7 Advanced tab

The following options are available in the Advanced tab:

- Standard messages should go to the following file (type "console" for console)

  This option provides a means by which the user can specify which file standard messages are directed to.

- Error messages should go to the following file (type "console" for console)

  This option provides a means by which the user can determine which file error messages are directed to.

- Name of local portal (created once during first connection attempt)

  This sets the name for the local portal within the JDE. It will be created on the first connection attempt with an external process. The portal is required for the JDE to be able to communicate with a running JACK process.

3.12 Window menu

When a project has been loaded into or created in the JDE and has been saved, the project name appears in the Window menu next to this heading, and project name information is visible in the menu.

For example, if the name of the project is `NewProject.prj`, the Window menu will contain the text Project: NewProject. Should the JDE be opened with no project loaded, this project name text will not be visible.

The Window menu will also display a list providing the names of any individual project windows that are open in the current JDE session.

3.13 Help menu

The Help menu provides the following options:

- JACK Documentation
- JACK Frequently Asked Questions
- Agent Oriented Software Home Page
- Create Project from Example
- About JACK Intelligent Agents (Windows only).

3.13.1 JACK Documentation

The JACK Documentation option provides a shortcut to documentation about the JDE.

3.13.2 JACK Frequently Asked Questions

The JACK Frequently Asked Questions option provides a shortcut to a HTML page providing JACK FAQs.

3.13.3 Agent Oriented Software Home Page

The Agent Oriented Software Home Page option provides a shortcut to the Agent Oriented Software home page at `http://www.agent-software.com.au`

3.13.4 Create Project from Example

The Create Project from Example option details how to create a project from the examples listed in the JDE. Various examples are provided.

To load one of the JACK examples:

1. Ensure that no other projects are open in the JDE, as only one project can be loaded at a time.
2. Select from the items displayed in the sub-menu of the Create Project from Example menu item.
3. A dialog box will provide a prompt to enter the directory in which the project should be saved.
4. Navigate to that directory in the dialog box.
5. Select the directory folder/name and click on the OK button.
6. The example project will open in the JDE browser, with instructions displayed in an editing window.

3.13.5 About JACK™ Intelligent Agents

About JACK™ Intelligent Agents provides information about Agent Oriented Software Pty. Ltd. It also provides information about the particular release version of the JDE and JACK being used, and further copyright information.

4 The JDE tool bar

The JDE tool bar lies directly below the JDE menu bar. The JDE tool bar provides a series of shortcut buttons that can be clicked to bypass the drop-down menus of the menu bar for many of the basic functions of the JDE.

The buttons available to the left perform various actions in the JDE, and are referred to as *action buttons*. The action buttons of the JDE Tool Bar are described in the next section.

There are also three buttons to the right end of the JDE tool bar. These buttons toggle various modes on and off, and are referred to as *toggle buttons*. The toggle buttons of the JDE tool bar are:

- Full Package Path On/Off
- Documentation Nodes On/Off
- Teams Mode On/Off.

The JDE tool bar can be dragged around the JDE canvas, docked to a different edge of the canvas, or can be used as a separate floating window on the JDE canvas, depending on the requirements of the user.

4.1 JDE action buttons

The JDE tool bar action buttons are described below.

- New Project button

  The New Project button provides a shortcut to creating a new project, without the need to file through the drop-down menus.

- Open Project button

  The Open Project button provides a shortcut to opening an existing project in the user's directory files, without the need to use the drop-down menus.

- Save Project button

  The Save Project button provides a shortcut to saving a project, without the need to use the JDE's drop-down menus. To save a project in this manner, simply click the Save Project button, and the project will automatically be saved.

- Update JACK Files button

  The Update JACK Files button provides a shortcut to updating the JACK files of a project, without the need for the user to file through the JDE's drop-down menus.

- Generate All JACK Files button

The Generate All JACK Files button provides a shortcut to generate all JACK files, without the need for the user to file through the JDE's drop-down menus.

- Cut Text button

  The Cut Text button provides a shortcut for cutting text, so that the user does not have to file through the JDE's drop-down menus.

- Copy Text button

  The Copy Text button provides a shortcut for copying text and placing it on the 'clipboard', so that the user does not have to file through the JDE's drop-down menus.

- Paste Text button

  The Paste Text button provides a shortcut for pasting text, without the need to use the Paste text option from the drop-down menus.

After the Paste Text button there are a series of *add* buttons enabling the user to shortcut through to the functions provided by the Entity menu.

These *add* buttons are listed below.

- Add Design button
- Add Agent button
- Add Capability button
- Add Plan button
- Add Event button
- Add Team button
- Add TeamPlan button
- Add Role button
- Add Named Role button
- Add Named Data button
- Add Beliefset button
- Add View button
- Add TeamData button
- Add External Class button.

The remainder of the *action* buttons are described below.

- Preferences button

The Preferences button opens the Preferences window directly, without the need for the user to file through the JDE's drop-down menus. The user can then alter the various preference options as necessary.

- Compiler Utility button

  The Compiler Utility button opens the JDE's Compiler Utility directly in its own window, without the need for the user to file through the JDE's drop-down menus for the Compiler Utility option.

- Error Log button

  The Error Log button opens the Error Log in its own window, without the need for the user to file through the JDE's drop-down menus.

Figure 4-1: JDE action buttons

4.2 Toggle buttons

The JDE tool bar toggle buttons are described below.

- Full Package Path On/Off button

  The Full Package Path On/Off button provides a shortcut to toggle the Full Package Path option. When the Full Package Path On/Off option is toggled on, the entire package path of the current project is listed.

- Documentation Nodes On/Off button

  The Documentation Nodes On/Off button provides a shortcut to toggle the Documentation Nodes On/Off option. When the Documentation Nodes On/Off is toggled on, the user is able to view and edit documentation for types and files, including the current project.

- Teams Mode On/Off button

  The Teams Mode On/Off button provides a shortcut to toggle the Teams Mode On/Off option. When the Teams Mode On/Off button is toggled on, the JDE is in Teams Mode and all Teams Mode functionalities are available.

Figure 4-2: JDE toggle buttons

5 JDE Compiler Utility

5.1 Opening the Compiler Utility

The Compiler Utility can be opened at any time by selecting Tools > Compiler Utility from the JDE menu bar. If the Compiler Utility is already open, its window will be brought to the front when the Compiler Utility option is selected.

A shortcut Compiler Utility icon is also available in the JDE Tool Bar. To open the Compiler Utility using this method, simply click on the Compiler Utility icon.

5.2 Closing the Compiler Utility

To close the Compiler Utility window at any time, click the Close button located at the bottom of the window.

5.3 Compiler Utility window

The Compiler Utility window consists of five tabs. These are discussed below.

- Options tab

  The choices made in the Options tab will affect the behaviour of the other tabs. Most of the options are equivalent to command-line arguments passed to the `aos.main.JackBuild` utility.

- Compile application tab

  Use this tab to convert an application from source code into an executable form.

- Run application tab

  Use this tab to execute a compiled application (or indeed any Java application).

- Convert non-JDE JACK tab

  Use this tab to generate files in graphical format (G-Code) from existing JACK code.

- Output/Errors tab

  This tab displays the output from compiling or running an application, as well as any output associated with the generation of G-Code.

The five Compiler Utility tabs are discussed in detail below.

5.4 Options tab

Most of the options in the Options tab are equivalent to command-line arguments passed to the `aos.main.JackBuild` utility. The choices made in this tab affect the behaviour of the other tabs.

5.4.1 Project Java properties

Each entry in the Project Java Properties: scrolling list is passed to the compiler; the entries are then passed to each subsequent stage in the build process.

To add an entry to the list, type it into the text box below the Project Java Properties textbox. Use the same format here as with the `-D` option with the `aos.main.JackBuild` utility. For example, entering `JAVAC=<property>` is equivalent to `-DJAVAC=<property>` being provided to `aos.main.JackBuild`.

5.4.2 Add entry to list

Once the desired property has been added, click the up-arrow button to insert it in the list.

To remove an entry from the list, click the entry once to select it then click the Remove button below the list.

5.4.3 Project Classpath

This list contains the locations that will be searched by the various compilers when building an application and by the Java runtime environment when running applications. This list initially contains the Project Classpath inherited from the environment in which the JDE was started.

To add a new location to the project classpath, click the Add button. A dialog box appears allowing the selection of a folder or a `.jar` file. Clicking the Choose button will add a selection to the Project Classpath list.

To remove an entry from the Project Classpath, click the entry in the list once to select it, then click the Remove button below the list.

The Up and Down buttons can be used to change the position of the selected item in the Project Classpath.

5.4.4 Checkbox options

The Options tab window contains several checkbox options that can be toggled on or off. These options are listed here:

- Compile in sub-folders too

This option causes the JACK compiler to look for JACK source files in all sub-folders as well as the current folder.

- Automatically add folder to CLASSPATH

  This option causes the current folder to be added to the classpath before compiling or running.

- Handle JACOB source code

  Provides the `-x` option to `JackBuild` to recognise JACOB `.api` files.

- Clean folder before compile

  This option causes the results of a previous compilation to be removed before each compile. This is slightly slower, but it avoids possible problems with incremental compilation where generated Java files may be mistaken for original source code causing 'multiple definition' errors.

- Show all files in listings

  This causes even hidden files to be displayed in the file listings.

- Disable compilation (report only)

  This option inhibits file generation by the JACK Compiler, and can be used to enable a 'dry run'.

- Clear output window before compile

  This option clears the output box in the Output/Errors tab before each compile.

- Clear error window before compile

  This option clears the errors box in the Output/Errors tab before each compile.

5.4.5 Save Options

To save the settings selected under the Options tab, click on the Save Options button. These settings will be stored in the configuration file: `.jack.properties` in the user's home area (system dependent).

Note: If options are changed and not saved, they will apply to the current JDE session only, and will need to be reset should a different JDE session be opened.

5.4.6 Close

To close the JACK Compiler Utility window, click on the Close button at the base of the Compiler Utility window.

5.5 Compile Application tab

The Compile Application tab provides access to the functionality of the JackBuild Utility. Use this tab to convert an application from source code into an executable form.

5.5.1 Folder

The Folder textbox lists the current folder. If there is a project open in the JDE, the current folder is the project's root directory; otherwise, it is the folder from which the JDE was launched.

The current folder can be changed by double-clicking another folder in the list, or by typing a new location into Folder and pressing ENTER or RETURN.

5.5.2 Choosing Files

The main part of the Compile Application tab is comprised of two lists, labelled Contents: and Specific Selection:(optional).

5.5.2.1 Contents:

The Contents: list contains all of the items in the current folder which are displayed in Folder.

5.5.2.2 Specific Selection: (optional)

The Specific Selection: (optional) list specifies which files to compile.

To add a file, select a file in the Contents: list and double-click with the mouse, or click the right arrow button on the Compile Application tab.

To remove a file, select a file in the Specific Selection: (optional) list and double-click on the file with the mouse, or click the Remove button on the GUI.

If the Specific Selection: (optional) list is empty, all JACK/Java files in Contents: are compiled.

The JACK compiler automatically detects and compiles the files required from existing JACK code in the current folder if no files are added to the specific selection.

Note: Files from different folders can be added to Specific Selection: (optional).

5.5.3 Java Arguments

The Java Args: text box is used for arguments to be passed to the Java compiler.

5.5.4 Extra Arguments

The Extra Args: text box is for any extra information to be passed to the compiler, or for options not accessible from the Compiler Utility's graphical user interface (GUI). Anything entered into this box should be a valid command-line argument to the JackBuild utility.

Take care not to duplicate any information or options already provided by the Compiler Utility's graphical user interface. The `-classpath` parameter, for example, is already supplied by the Preferences tab, and the `-dj` and `-dc` options respectively come from the Generated Java Sub-folder and the Generated Class Sub-folder attributes in the project window.

5.5.5 Compile

To compile an application, click the Compile button. If there is currently a project open in the JDE, the JACK files are updated (this is equivalent to selecting File > Update JACK Files in the JDE menu bar) and the compilation process itself begins. The Output/Errors tab will be activated, displaying any output from the compiler as it builds an application. When the procedure is complete, the Compile Application tab is displayed again unless an error has occurred.

5.5.6 Stop

While an application is being compiled, a Stop button is visible, with which the user may stop the compilation of the project.

5.5.7 Clean Up

To remove any files from a previous build, or to remove intermediate files from a failed build, click the Clean Up button. This is equivalent to JackBuild's `-clean` option. The Output/Errors tab will be activated, displaying any output from the compiler as it removes files. When the procedure is complete, the Compile Application tab is displayed.

Note: there is usually no need to do a manual cleaning if you have selected the Clean folders before compiling preference in the Preferences tab of the Compiler Utility.

5.5.8 Close

The Close button closes the JACK Compiler Utility window.

5.6 Run Application tab

A compiled application can be executed with the Run Application tab. Like the Compile Application tab, there is a list displaying all of the items in the current folder (which is shown in Contents:). When an item in the list is clicked, information about the file or folder is shown on the right-hand side of the window.

The user specifies the class file to be run by first selecting it from the Contents: list, then clicking the Select File button. The class name is shown in the text box next to Select File. Alternatively, one can double-click on the item, or enter the name of the file into the text box adjacent to the Select File button.

When a class file is selected in the Contents: list it becomes highlighted, and information about the file becomes visible in the right side of the Run Application tab window. This information is specifically: Kind: (the type of file), Size: (the size of the file) and Modified: (when the file was last modified, including the date and time in the format *DD/MM/YYYY HH:MM:SS*).

If any command-line arguments are required by the application, they are entered in the Extra Args: text box. Similarly, the Java Args: text box is used for arguments to be passed to the Java interpreter. Also ensure that the Folder: entry is correct for running the application – the Folder: entry will be added to CLASSPATH as the application's root point.

Note: If the main class is located within a sub-directory, make sure that you return to the applications root point within the JDE browser once it has been selected, before proceeding with the next step. At this point, the folder entry should correspond to the root point of the application and the main class will be specified relative to the root point.

To run the application, click the Run button. The Output/Errors tab is not automatically activated so you need to select it manually to display any output or errors from the application.

When an application is started, the Run button is transformed into a Stop button until the running process exits.

5.7 Convert Non-JDE JACK tab

The Convert Non-JDE JACK tab is used to generate a new project from existing JACK code that was not developed using the JDE.

When creating editor files from existing JACK source files and compiling the application from within the JDE, original JACK source files will be overwritten. To keep those original files, ensure that a backup has been made.

5.7.1 Folder

As with the Compile Application tab and the Run Application tab in the JACK Compiler Utility window, the Convert Non-JDE JACK tab contains a Folder textbox, in which the current directory is displayed.

If there is a project open in the JDE, the current folder is initially the project's root directory. If no project is open, then it is the folder from which the JDE was launched.

The location of the current project can be changed by entering the new location and pressing ENTER or RETURN, or by using Contents:.

5.7.2 Choosing Files

The top section of the tab is similar to the Compile Application tab. It consists of two lists, labelled Contents: and Specific Selection: (optional).

5.7.2.1 Contents:

The Contents: list contains all of the items in the current folder.

5.7.2.2 Specific Selection: (optional)

The Specific Selection: (optional) list specifies which files to generate G-Code for by adding them to its list.

5.7.3 Creating a Project File

The editor files generated via the Convert Non-JDE JACK tab can be inserted into a project file, thereby making them available as a group to the JDE. To specify the name for this project file, either enter the path to the file in the project file text box or click the Browse button to specify a file in a different folder.

5.7.4 Specifying the Package

If the JACK source files that editor files need to be generated from are part of a Java package, the root package can be entered in the Root Package text box. When the files are generated, this prefix will be inserted into each object's package statement.

5.7.5 Java Arguments

This text box is used for arguments to be passed to the Java compiler.

5.7.6 Extra Arguments

The Extra Args: text box is used to provide any extra information to the compiler when it generates the files, or use options not directly accessible from the JACK Compiler Utility. Anything entered in this box should be a valid command-line argument to the JackBuild utility.

5.7.7 Create Project File

The Create Project File: option allows the user to type in the name of the required project file, or to browse through existing files using the file chooser option, which opens when the Browse button is clicked.

5.7.8 Root Package

If the user specifies a root package, any generated import statements will omit that part of the package name.

5.7.9 Generating the Files

To start the file generation:

1. Click the Generate New JDE Files button.
2. The Output/Errors tab is activated, displaying any output from the compiler as it generates files.
3. When the procedure is complete, the Convert Non-JDE JACK tab is reactivated unless an error occurred.

5.7.10 Close

The Close button closes the Compiler Utility.

5.8 Output/Errors tab

The Output/Errors tab displays the output of the commands available from the other tabs. This tab can be manually selected to view the output of previous commands.

To save the output from previous commands, click the Save to File button.

To empty the text fields of the output from previous commands, click the Clear button for the field to be emptied. Save the output or errors to a file by clicking the relevant Save To File button. The Output/Errors tab also includes a Stop button with which the compilation process can be stopped.

5.9 Project files and file listings

When a project is open in the JDE, certain aspects of the file listings displayed in the Compile Application, Run Application and Convert Non-JDE JACK tabs are slightly different. When there is no project open, the current folder defaults to the one from which the JDE was launched. However, if there is a project open, the current folder defaults to the project's root directory.

When a project is open, files that have been modified but not saved are displayed in italics. This also applies to files that have not yet been generated (in the case of generated JACK source files). These files can still be selected because they will be generated when the Compile button in the Compile Application tab is clicked.

6 Code Editor

The Code Editor appears whenever a text item is edited.

The kinds of text items that are edited with the Code Editor include:

- Documentation
- Java code (contained in the Other Code folder of an entity)
- Reasoning methods (only textual, not graphically edited reasoning methods)
- Posting methods
- External Files (contained in the Other Files folder).

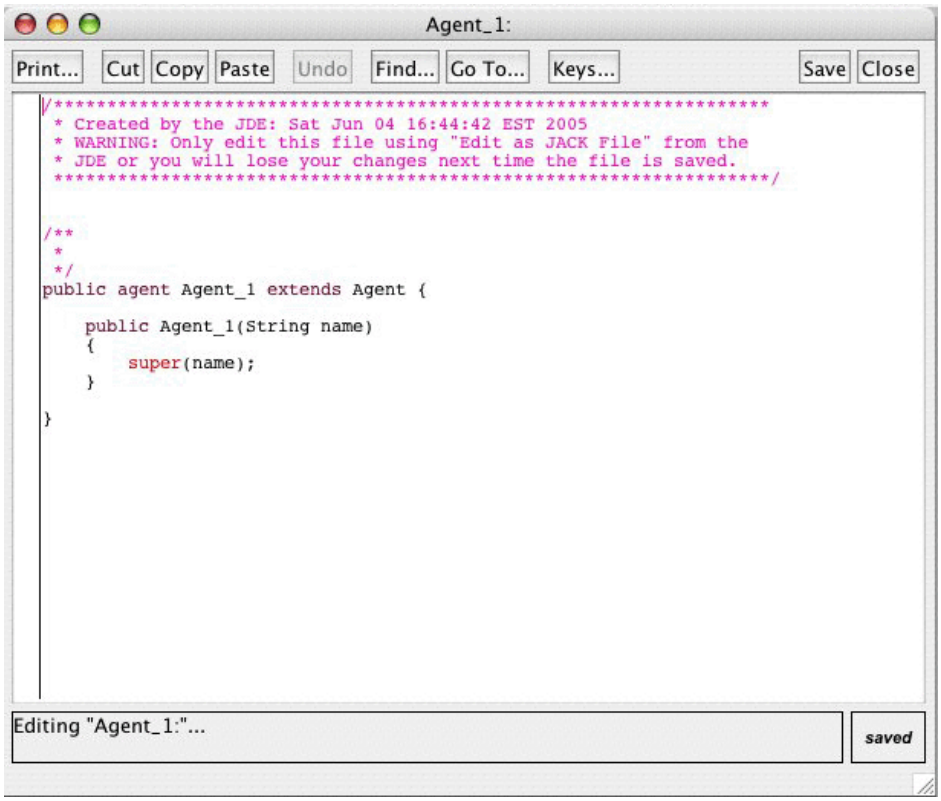

Figure 6-1: Example of the Code Editor window

In addition to the basic editing functions listed in the *Code Editor tool bar* section, the Code Editor offers the following features:

- Automatic indentation

Whenever a new line is added to the code, it is automatically indented to match the line above. For example, if a line of code that begins with four spaces is entered, when ENTER is pressed to move to the next line, four spaces will automatically be entered.

- Bracket balancing

  When a closing bracket], brace } or parenthesis) is entered, the matching opening symbol is highlighted. If there is no matching symbol, the brackets, braces or parentheses are unbalanced and a warning beep is sounded.

- Line wrapping

  If a line of text is too long to fit in the Code Editor window, it will automatically wrap onto the next line so that it can be seen. Note that the line will not be split when it is saved to a file.

6.1 Code Editor options bar

The Code Editor has a number of features designed to facilitate writing JACK code. Many of these are accessible from the options bar located in the Code Editor window. These features include the following buttons:

- Print button
- Cut button
- Copy button
- Paste button
- Undo button
- Find button
- Go To button
- Keys... button
- Save button
- Close button.

6.1.1 Keys... button

The Keys... button opens the Key Bindings window. In the categories drop-down menu, the following options are available:

- Edit Functions
- Move/Scroll Functions
- Control Functions
- Customisation

- Miscellaneous.

The Key Bindings options enable the user to attach various functions to various non-printable keys on the keyboard.

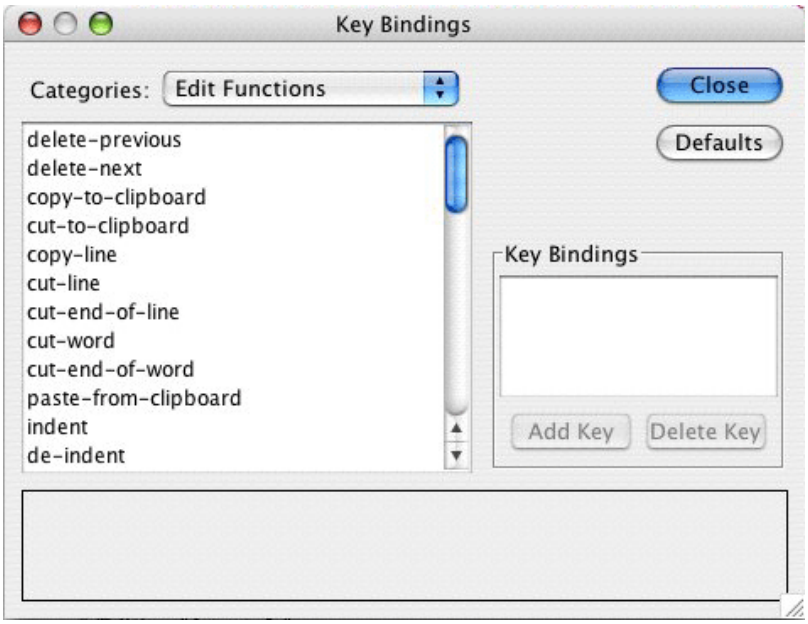

Figure 6-2: The Key Bindings window

6.2 Saving changes internally

For editing code that is internal to the JDE (i.e. not an external file), the Code Editor window also provides the following options: Save and Close.

- Click the Save button to store work in the project and close the Code Editor window.
- Click the Close button to close the Code Editor window without storing work. A confirmation prompt will appear, and if the user confirms their intent to close the window, changes made since the window was opened (or the last time Apply was clicked) will be discarded.

Note: Storing work in the project does not mean it is saved on the file system. To save stored changes to the file system, choose the Save Project option from the File menu.

6.3 Saving changes to external files

When editing an external file, the Code Editor window has three buttons: Save, Reload... and Close.

- Click the Save button to save work, but leave the window open so work can continue.
- Click the Reload... button to reload the file from the file system. Any changes made since the window was opened (or the last time Save was clicked) will be discarded and the file will be reloaded into the editor.
- Click the Close button to close the Code Editor window without saving work. A confirmation prompt will appear, and if a user confirms their intention to close the window, changes made since the window was opened (or the last time Save was clicked) will be discarded.

Note: In contrast to code that is internal to the JDE, when changes are saved to an external file, they are immediately stored on the file system.

7 JDE browser

7.1 Introduction

The JDE browser allows the user to view and modify the components in a project from within one or more browser windows. When a project is opened, a special browser window is opened which is known as the project window. Only one project window can be open at a time within a JDE session, and this window contains the details of all the components in a project. Additional browser windows can be opened by dragging selected folders or components into the work area.

The project window contains the details of the project:

- Name
- Documentation
- Design Views folder
- Agent Model container
- Data Model container.

The Agent Model and Data Model containers have folders in which the various programming element types can be stored. Further, the project window contains the Other Files folder, into which non-JACK files of an application (e.g. Java files) can be placed.

These folders allow the grouping of elements in a project. For large projects, some folders will contain many elements. In such situations, it is useful to be able to group the elements within a folder into sub-folders. This can be achieved through the use of *nested containers*. Containers can be nested within containers, with no limit to the number of nesting levels. Elements of the same type can also be moved about between nested containers.

This chapter describes the general functionality of the JDE browser. The specific operations that can be performed within the Agent Model and Data Model containers are discussed in separate chapters.

7.2 Browser interaction

The elements in a JDE browser window's folder hierarchy can be expanded or contracted by clicking on the arrow to the left of the element (Windows) or by double-clicking on the element's folder or name.

Note: All JDE folders must have contents in order to be expanded or contracted this way.

When an element is expanded, the right-arrow to the left of it transforms into a down-arrow. Clicking on the down-arrow (or double-clicking on the element next to the down-arrow) collapses the expanded tree.

The JDE browser tree can also be navigated with the keyboard rather than the mouse. To do this, use the UP and DOWN arrow keys to highlight the desired folder, then press ENTER or RETURN to expand or contract the highlighted folder.

Note: On Unix and Windows systems, the icon next to an element label to be expanded or collapsed is represented as a + or a – sign. On a Macintosh, the icon next to an element label to be expanded is represented as an arrow pointing to the right, and to be collapsed as an arrow pointing down.

Figure 7-1: Unexpanded models in the JDE browser

Context-sensitive menus are provided to perform operations on folders and folder elements – these are accessed by placing the mouse cursor over an item and pressing the right mouse button. A menu then appears providing operations that can be applied to the selected item.

Drag-and-drop is used to associate one element with another (such as a plan with an agent). The icon to be dragged is first selected by clicking and holding the left mouse button down while over the icon. The left button is then held down while the cursor is moved onto the destination icon. Releasing the left mouse button will then associate the dragged element with the destination element.

Folders that can have elements dragged and dropped onto them have a small rectangular depression in the middle. These are known as *target folders*.

Note: If a drop operation is permitted, the Drop Allowed cursor is displayed. However, if the operation is not permitted, the No Drop Allowed cursor is displayed.

Multiple drag-and-drop can also be performed. The JDE allows multiple drag-and-drop by holding down the SHIFT key and selecting elements in sequence by clicking on their names or associated icons, then dragging the cursor over to the element with which they will be associated and releasing the mouse button. This operation allows selection of multiple items that can then be dropped on a target.

Figure 7-2: Multiple Drop Allowed cursor

The CTRL key can be held down to select individual entities in a list without having to select a range.

7.3 Naming elements

A JDE project consists of programming elements that appear in the Agent Model folder, the Data Model folder, and the Other Files folder. Some of these elements are type definition elements (e.g. agents, plans, beliefsets, Java class files), and some elements are naming elements (e.g. data naming elements and role naming elements).

The purpose of naming elements is to associate names with data types, to support consistent use of data within the agent program. The naming elements themselves belong to the Agent Model container of the project, while their types belong to the Data Model container or Other Files folder. In particular, naming elements may be used in the design diagrams, whereas Data Model elements cannot occur in such diagrams.

Within the project, all references to data are via naming elements. The data naming elements can be dragged and dropped in the usual way to achieve the required references in the type definitions. Similarly, role naming elements can be dragged and dropped onto the Roles Required folder of a team type definition and the Roles folder of a teamplan type definition.

7.4 Element reuse

In an agent-based application, agents often form only part of the overall application. The agent part of an application is referred to as the *agent model*. For example, if agents are used to control a manufacturing cell, the agents must interact with the machines in the cell. These machines will have been programmed using their individual controllers. The interaction between an agent and its corresponding machine may then be achieved by using an existing communications infrastructure. Thus the application (cell control) involves a number of software elements in addition to the agent model. An agent model may utilise existing agent models or parts of those models (e.g. agents and/or capabilities) that have been developed for other applications.

A JDE project acts as the repository for an agent model developed with the JDE. This agent model may be a stand alone application, but as noted above, it might also be a part of a larger application. The elements of an agent model can be either of the following:

- Defined by the user

The support provided by the JDE browser for the definition of new elements is the focus of the remainder of this chapter and the subsequent *Agent Model* and *Data Model* chapters.

- Reused from other projects

  A project contains elements that the user has defined specifically for the project via the JDE. It can also contain elements from other projects.

The JDE supports two types of element reuse – either a link can be created for an element in another project, or an element definition in another project can be incorporated directly into the project. In the latter case, the G-Code for the element is shared by any projects that use the element. This means that a change made to the element definition in one project will be seen by all projects that use that element. Note that each project that uses the element will generate JACK code for the element during the compilation process.

In the case of a link, the linked element can be expanded, viewed and dragged around like a normal element, but it will not be saved in any file and no JACK code will be generated for it. Linked elements are marked as external, and have a hyphen before their package name. Both types of reuse are supported through the JDE browser Importing an Element function, which is described in the *General functions* section of this chapter.

7.5 General functions

7.5.1 Type definition/naming element folder functions

The JDE browser supports the following operations for both type definition folders (e.g. the Agent Types folder) and naming element folders (Named Data folder and Named Roles folder). Type definition folders contain type definitions; naming element folders contain naming elements. In the following, type definitions and naming elements are collectively referred to as *entities*.

7.5.1.1 Editing a folder's label

The label of a folder can be edited by selecting the Edit Label option from the folder's pop-up menu.

To edit the label of a folder:

1. Right-click on the label of the folder that requires changing.
2. Click on the Edit Label option in the pop-up window. The label of the selected folder will now become a text box, and it can be edited as required.
3. Press ENTER or RETURN to commit, or ESCAPE to discard.

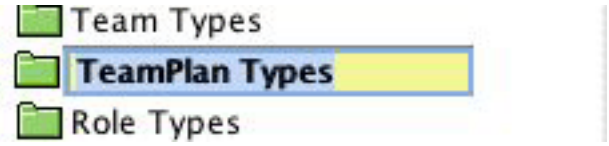

Figure 7-3: Label of the selected (TeamPlan Types) folder becomes an editable textbox

7.5.1.2 Creating/adding a new element definition

To create a new element definition in a folder, right-click on the relevant folder label and select the Add New option.

For example, to create a new agent type definition, right-click on the Agent Types folder label, and select Add New Agent.

The Project View tab in the Preferences dialog has a checkbox named Pop-up dialog when creating new objects. If that checkbox has been ticked when the Add New Agent option is selected, the pop-up dialog for defining a new type will appear. Otherwise, a new element with a generated name is added to the folder, and thereafter the JDE browser enters the Edit Name function for that element. Press ENTER to accept the generated name, or to change it as needed.

Pressing the ESCAPE key at any point during this editing operation will abort the operation and cause the name to revert to its previous state.

Each type must have a unique name within a given project. If a name that already exists is entered in the current project, the user will be alerted and will be returned to the Type Name-entry text box.

7.5.1.3 Importing an element

The JDE supports several different ways of importing elements into the current project.

- Import the element definition directly into the current project

  Importing the element definition directly into the project results in the G-Code for the element being shared among projects that use the element. Changes made in one project will be reflected in other projects that use the element. Each project that uses the element will generate its own JACK code for the element during the compilation process.

  An imported (but not external) component from a different project is marked as changed so it will be saved within the current project. This effectively makes a copy of the component.

- Import a link to the element into the current project

Importing a link to the element means that the actual G-Code for that element is not accessible for modification within the project. However, the link can be expanded, viewed, and dragged around like a normal element. Links are saved within the `.prj` file, but no G-Code or JACK code is generated for them within the project. A link will appear identical to other elements within the JDE browser tree, except that its label will be coloured grey instead of black. Linking an element into the current project is achieved by importing it as an external element. An element is identified as external within the JDE by the fact that its package name begins with a hyphen.

Importing an external component (via a project file) will attempt to automatically import any other component it references. These components are marked as external and are grouped in their own folder based on the package from which they came. If a component is imported by loading its G-Code directly (rather than via a project file), it may be necessary to load the G-Code of any other components it references.

The original contents of any external components become cached in a folder called `ext-cache` instead of being saved in the project file. This helps avoid contention for the project file when multiple users are working on the same project.

- Import stubs for the element

  Importing stubs for an element enables the user to import an element but discard its original contents.

In order to import an element, its G-Code (the graphical file format used by the JDE) must be available. In other words, either it was created as part of another project using the JDE, or the original JACK code must have been converted into the graphical file format. This conversion can be done by the JACK compiler from the command line, or by using the JACK Compiler Utility within the JDE.

To import an element:

1. Right-click on the folder that holds elements of the type in question, such as the Plan Types folder. This will elicit a context-sensitive menu with an Import option.
2. Click on the Import option.
3. A directory window will appear, through which the user can navigate to select the appropriate element to be imported.

Within the Import directory window discussed above, the following checkbox options are also present:

- Create Stubs Only

  This option enables the user to import a component without the actual contents of the component.

- Mark As External

This marks the element as external by adding a hyphen before the package name. This means that the project only has a link to the element.

If neither of these options are selected, the element is imported directly into the project.

An element that has been imported directly into the project can be made external by prefixing its package name with a hyphen. The element is then treated by the JDE as a normal external element. However, if an element which is declared to be external is de-externalized (by the removal of the package prefix), the element will be treated as if it had been defined within the project. In particular, it will have its own G-Code, which is a copy of the G-Code for the original (external) element.

7.5.1.4 Sort Elements

The Sort Elements option enables the user to sort the elements of any given type into alphabetical order.

7.5.1.5 Adding a nested container

The Add Nested Container menu item is used to create a sub-folder of a Types folder, or an Agent Model or Data Model container. The sub-folders can then be used to organise types into meaningful groups.

To add a nested container:

1. Right-click on the folder to which the nested container will be added. A pop-up menu will appear, providing the Add Nested Container option.
2. Click on the Add Nested Container option.
3. A new nested container will appear within the folder, and its name can be edited as required. By default, the name of the new nested container is generated by appending (sub) to the parent folder's name.

Right-clicking on a nested container displays a menu which gives the user access to the same operations that were available when right-clicking on a top level container. In addition, the user has the option of removing the nested container. If a nested container is removed, its contents will be moved up one level and placed within the parent container. Thus, elements of a project are protected from being accidentally lost.

7.5.2 Type definition/naming element functions

The JDE browser supports the following operations on individual type definitions and naming elements. Type definitions and naming elements are collectively referred to as entities.

7.5.2.1 Editing an entity name

Once an entity has been created, its name can be edited at any time.

1. Right-click on the entity and select the Edit Name option from the pop-up menu.
2. After selecting this option, enter the replacement name, or move the insertion point with the mouse to modify the existing name.
3. Press the ENTER or RETURN key to commit changes, or the ESCAPE key to abort the editing operation and keep the original name.

If the new name already exists, the JDE will alert the user to that fact and it will be necessary to find another name or package for the element.

When an entity is renamed, the JDE will ask whether to locate and change all references to it (unless the Automatically Update References... preference option is selected).

7.5.2.2 Edit as JACK file

Right-clicking on a JACK type definition in the JDE browser elicits the Edit as JACK File option. This will present the entire entity in an editing window in the same form as would be passed to the JACK compiler. The code can be edited, and when the changes are saved (or the editor is closed) the changed text will be parsed and broken up into its individual elements, and the entity in the JDE browser will be updated accordingly. Depending on the number and degree of changes, the user may be asked to confirm the changes first. The amount of change required to trigger this confirmation can be set in the Preferences option of the Edit menu.

After editing an entity in this way, some elements may be reordered, renamed, restructured or even removed if they are not required. Some textual elements, notably those resulting from graphical reasoning methods, can be edited, but any changes to them will be ignored. These parts of the text will have clear comments stating this.

Note: Avoid making changes via the JDE browser while also using Edit as a JACK File on the same entity. Any changes made directly from the JDE browser may be lost when the edits from Edit as a JACK file are saved and loaded. The JDE will attempt to warn you before this happens.

7.5.2.3 Editing an entity

Right-clicking the Edit menu option brings up a dialog box in which the entity's name, package and documentation can be changed.

7.5.2.4 Adding a copy of an entity

Right-clicking on an entity and selecting the Add Copy option in the pop-up menu bar will create a copy of the selected entity, which can then be renamed with its own unique name.

7.5.2.5 Adding a new entity

For convenience, the menu at this level also allows the user to select the Add New option that was available from the folder level menu. In this case, the new entity will be added to the encompassing folder.

7.5.2.6 Importing an entity

For convenience, the menu at this level also allows the user to select the Import option that was available from the folder level menu. In this case, the imported entity will be added to the encompassing folder.

7.5.2.7 Reloading an entity

Reloading is equivalent to removing an entity and reopening the last saved version.

To reload from the file system, right-click on the type and select the Reload option. Effectively, when this menu item is selected, the type is first removed and then loaded again.

7.5.2.8 Removing a type definition or naming element

To remove an element from a folder, and hence from the current project (for example, to remove an agent type):

1. Right-click on the relevant entity.
2. Select the Remove menu item. A prompt will appear confirming the element's removal from the project.
3. To cancel the operation, press the Cancel button. To proceed, specify whether the JDE should locate and remove all references to this element.
4. Click on Delete File to confirm this.
5. To keep the references and the G-Code file, click on Keep Ref's.

7.5.3 Operations on type definition and naming element attributes

Depending on the particular type definition or naming element, one or more of the following attributes can be modified using the JDE browser:

7.5.3.1 Extends field

The base class that types extend can be altered by editing the Extends field. Base classes are provided in a pop-up context-sensitive menu. To edit the Extends field, right-click the field.

If Edit Value is selected, a personalised base class can be entered, but the base class entered must be defined in Java code and must itself extend a type base class.

7.5.3.2 Documentation field

To edit documentation, right-click on that field, select Edit Documentation and type the documentation into the editing window that appears.

The label associated with the Documentation field can also be edited. To edit this label, right-click on it, select Edit Name, and type in the new label for the Documentation field. The new documentation label will be displayed as a Doc: (documentation) option. If a space is entered before the label, the Doc: prefix and the space will be omitted from the label.

7.5.3.3 Constructor

To edit the constructor of a type, right-click on the Constructor field to pop up a menu, then select Edit Constructor. This opens a Code Editor window in which the text of the constructor can be edited.

Instead of hard-coding the name of the constructor into the text, the word `<<Type>>` can be used as a place-holder.

7.5.4 Operations on Java code

Type definitions can contain Java code. The JDE browser provides support for manipulation of the following Java constructs:

7.5.4.1 Package declarations

Java code and declarations outside of JACK Agent Language constructs that would normally appear as part of a type definition are placed in the Java sub-folder of the type definition folder.

To change the Java code's package:

1. Right-click on the Package field.
2. Select Edit Value from the pop-up menu that appears.
3. Next, enter the name with which the original name will be replaced, or move the insertion point with the mouse to modify the existing name.
4. Press the ENTER or RETURN key to enter changes, or the ESCAPE key to abort the editing operation and keep the original name.

Any element can be made external by prefixing the package name with a hyphen, which then displays the Make External dialog. This has a similar effect to that of adding an entity to the External Classes folder except that the latter only refers to Java classes. By prefixing the package with a hyphen, a reference to the entity is effectively imported.

7.5.4.2 Implements declarations

Use the Implements folder to declare interfaces that the type definition implements. Right-click on the Implements folder and select Add New Interface. In the text box that appears type the name of the interface that this type definition implements and press ENTER or RETURN.

Figure 7-4: Add New Interface in the Implements folder

The `implements` keyword does not need to be supplied as it is automatically inserted when JACK code is generated.

To edit the name of an interface, right-click on the interface and select Edit Value. In the text box that appears type the new name of the interface and press ENTER or RETURN.

To remove an interface from the project, right-click on the interface and select the Remove option.

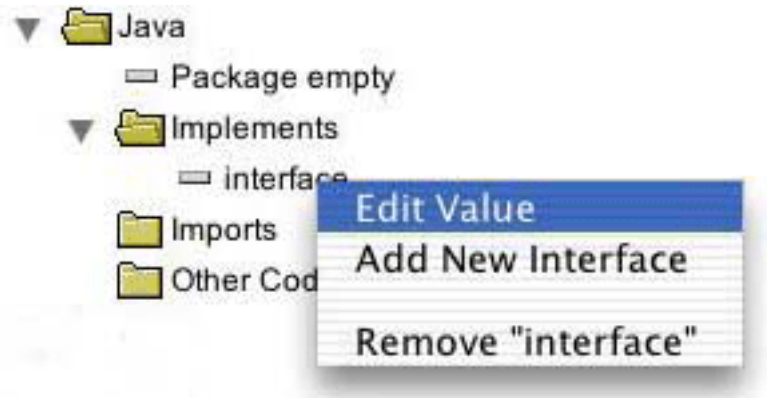

Figure 7-5: Editing an Interface

7.5.4.3 Import statements

Use this folder to declare any imports that are required for the type definition. This will be needed if classes from another package have been used in Java code added to the type definition in the Other Code folder.

Right-click on the Imports folder and select Add New Import. In the text box that appears type the full package path of the class that this type definition needs to import and press ENTER or RETURN. The `import` keyword does not need to be supplied as it is automatically inserted when JACK code is generated.

To edit the name of an import, right-click on the import and select Edit Value. In the text box that appears type the new name of the declaration and press ENTER or RETURN.

To remove an import from the project, right-click on the declaration and select the Remove option.

The following figure shows the pop-up menu that you get by right-clicking on an existing declaration in the Imports folder.

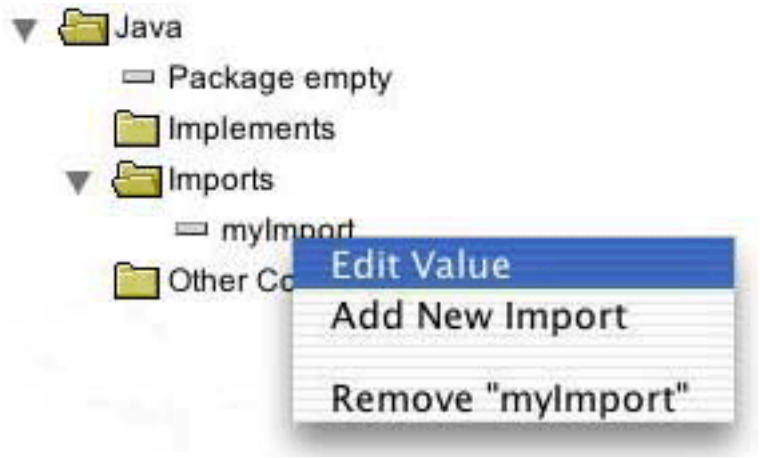

Figure 7-6: Editing an import declaration

7.5.4.4 Members and methods

To add members and methods to a type definition use the Other Code folder of the type definition. Use this folder to add methods or members to the type. Right-click on the Other Code folder and select Add New Code. This opens a new editing window.

Each of these code blocks can be named using the Edit Name option obtained by right-clicking on the field in question. This name is only for reference, and appears in a comment above the generated code.

To edit a given code block, right-click on the item and select the Edit option.

To remove a code block from the project, right-click on the item and select the Remove option.

7.5.5 Operations on references

One entity can refer to another type of entity if it contains a reference to that entity. These references usually map to JACK declarations such as `#handles MyEvent` or `#uses MyPlan`. These references can be made in the JDE by using the JACK Design Tool, or by a drag-and-drop method in the JDE browser as described below.

The JDE browser supports the following operations for references.

7.5.5.1 Adding a type definition reference

To add a type definition reference, drag the relevant JDE browser entity and drop it onto the target folder.

For example, to add a new plan reference to an agent type definition, drag the relevant plan from the Plan Types folder and drop it on the agent type defintion's Plans folder. This creates a plan reference from the agent type definition back to the plan that was dragged into the target folder.

The icons that indicate whether or not an operation is allowed are described in the *Overview* section of this manual.

7.5.5.2 Editing the reference name

To edit the reference name (if present):

1. Right-click on the reference and select Edit Name.
2. Enter a preferred name, or select an insertion point with the mouse to modify the existing name.
3. Press ENTER or RETURN to complete the change of name.
4. Clicking ESCAPE at any time during this editing operation will abort the operation and cause the name to revert to its original state.

7.5.5.3 Editing the referenced type definition

To edit a referenced type definition, right-click on the reference and select the Edit option. This opens a new window offering an editable view of the actual type definition. Any changes made here are changes to the original type definition and affect all references to that type definition.

7.5.5.4 Removing a type definition reference

To remove a reference from a type definition, right-click on the reference and select the Remove option. This will remove the reference from the type definition. The type definition that was being referenced is untouched.

7.5.5.5 Defining the posting type of an event type definition reference

To change an event reference posting type, click on the arrow beside the event reference icon to expand the tree below the reference (or double-click on the event reference). Then right-click on the Attrib: (attributes) field to obtain a pop-up menu of choices. Select either `handles`, `posts` or `sends` to indicate the posting type.

7.5.6 Operations on naming element references

The JDE browser supports the following operations for naming element references.

7.5.6.1 Adding a naming element reference

To add a naming element reference to a type definition, drag the naming element to the target folder in the type definition. For example, to add a data naming element reference to a capability, one would drag the required data naming element from the Named Data folder to the target Belief Data folder within the capability type definition. This would cause the reference to appear in the Belief Data folder.

7.5.6.2 Editing a naming element reference

As with other references, the original data naming element or role naming element can be opened and edited by right-clicking on the reference and selecting the Edit option.

7.5.6.3 Removing a naming element reference

To remove a naming element reference from a type definition, right-click on the reference in the type definition and select the Remove option.

7.5.6.4 Adding parameters (data naming elements only)

In some cases the reference requires parameters. In such cases, right-clicking on the reference will display a menu which contains an Add parameters option. Selecting this option will allow the user to add parameters to the reference.

7.5.6.5 Defining the access mode (data naming elements only)

To change the access mode (internal data) or status mode (external data) click on the arrow beside the data reference icon to expand the tree below the reference (or, equivalently, double-click on the reference). Right-clicking on the access or status fields displays a pop-up menu of choices. Select the appropriate access or status mode from the pop-up menu.

7.5.6.6 Editing the specification (role naming elements only)

Where the reference to a role naming element specifies that the team type will require subteams to perform a role on its behalf, the specification should include details of the

maximum and minimum number of sub-teams required in the role. To edit the specification to add this information, right-click on the reference and select the Add Specification option from the menu.

7.6 Agent Model

The Agent Model container is discussed in the *JDE browser: Agent Model* chapter of this document.

7.7 Data Model

The Data Model container is discussed in the *JDE browser: Data Model* chapter of this document.

7.8 Other Files folder

The Other Files folder can be considered to be an index of the data files and Java files required by a project. That is, the files themselves are stored external to the JDE, and the Other Files folder contains references to the files, which are displayed as path names.

While the JDE provides support for the creation and modification of files referenced in this folder, it is anticipated that other development environments might be utilised for this aspect of project development.

7.8.1 Operations on the Other Files folder

Right-clicking on the Other Files folder elicits the Other Files options menu.

The Edit Label and Add Nested Container selections are common to all folders and are described in generic terms in the *General functions* section of this chapter.

The Add New File selection results in the creation of an editor window for entering the new file. Note that the label is initially unnamed – this will be modified when the the edited file is saved. Use of the default editor is described in the *Code Editor* chapter of this document.

The Import File selection creates a reference to an existing file. It uses your system's file chooser to select the file to be imported.

7.8.2 Operations on a file in the Other Files folder

Right-clicking on a file in the Other Files folder, results in the appearance of the following menu:

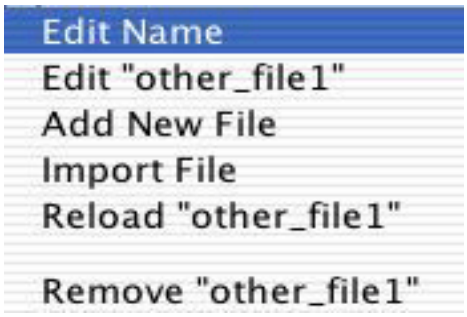

Figure 7-7: Other Files definition menu

The Add New File and Import File selections behave in the same way as the corresponding selections in the Other Files folder. The Edit option behaves similarly to the Add New File selection, except that when the editor window is created, it contains the specified file.

The Remove option removes the label from the folder. Note that the file itself is not deleted – if this is desired, it must be done externally of the JDE.

The Reload option provides the option of reverting to the last saved version or the original version of the file.

8 JDE browser: Agent Model

8.1 Agent types

8.1.1 Agent Types folder

Right-clicking on the Agent Types folder results in the following menu:

Figure 8-1: Agent Types folder options menu

The operations embodied in the menu choices are common to all folders and are described in generic terms in the *General functions* section of the *JDE browser* chapter.

8.1.2 Agent type definitions

Right-clicking on an agent type definition results in the appearance of the following menu:

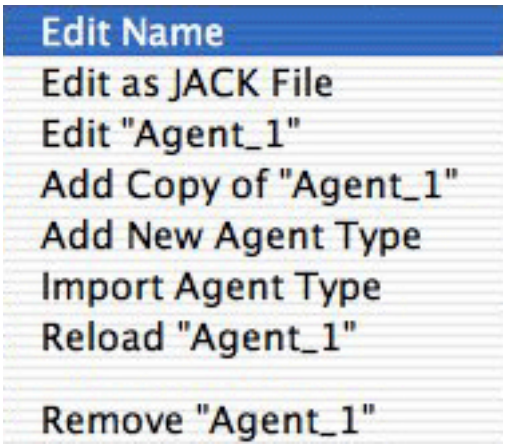

Figure 8-2: Agent type definition options menu

The operations embodied in the menu choices are common to all agent type definitions and are described in generic terms in the *General functions* section of the *JDE browser* chapter.

8.1.3 Agent type definition attributes

The Extends, Documentation and Constructor fields of an agent type definition can be modified via pop-up menus which are generated by right-clicking on the appropriate field.

These operations are described in generic terms in the *General functions* section of the *JDE browser* chapter.

8.1.4 Java code

Agent type definitions may contain Java code. The JDE browser supports the following operations on Java code within a type definition:

- declaration of interfaces
- declaration of imports
- addition of methods or members.

These operations are described in the *General functions* section of the *JDE browser* chapter.

8.1.5 Type definition references

An agent type definition can contain references to the following type definitions:

- capabilities
- events
- plans

The JDE browser supports the following operations on the above references:

- adding the reference
- editing the type definition
- removing the reference
- editing the reference name (capabilities and events only)
- defining the reference type (events only).

These operations are described in the *General functions* section of the *JDE browser* chapter.

8.1.6 Data naming element references

An agent type definition can contain references to data naming elements.

The JDE browser supports the following operations on data naming element references:

- adding the reference

- adding parameters
- editing the data naming element
- removing the reference
- defining the access mode.

These operations are described in the *General functions* section of the *JDE browser* chapter.

8.2 Capability types

8.2.1 Capability Types folder

Right-clicking on the Capability Types folder results in the appearance of the Capability types options menu.

The operations embodied in the menu choices are common to all folders and are described in generic terms in the *General functions* section of the *JDE browser* chapter.

8.2.2 Capability type definitions

Right-clicking on a capability type definition results in the appearance of the Capability types definition options menu.

The operations embodied in the menu choices are common to all type definitions and are described in generic terms in the *General functions* section of the *JDE browser* chapter.

8.2.3 Capability type definition attributes

The Extends and Documentation fields of a capability type definition can be modified via pop-up menus which are generated by right-clicking on the appropriate field.

These operations are described in generic terms in the *General functions* section of the *JDE browser* chapter.

8.2.4 Java code

Capability type definitions may contain Java code. The JDE browser supports the following operations on Java code within a type definition:

- declaration of interfaces
- declaration of imports
- addition of methods or members.

These operations are described in the *General functions* section of the *JDE browser* chapter.

8.2.5 Type definition references

A capability type definition can contain references to the following type definitions:

- capabilities
- events
- plans
- teamplans.

The JDE browser supports the following operations on the above references:

- adding the reference
- editing the type definition
- removing the reference
- editing the reference name (capabilities and events only)
- defining type of the reference (events only).

These operations are described in the *General functions* section of the *JDE browser* chapter.

8.2.6 Data naming element references

The JDE browser supports the following operations on data naming element references:

- adding the reference
- adding parameters
- editing the data naming element
- removing the reference
- defining the access mode
- defining the belief status.

These operations are described in the *General functions* section of the *JDE browser* chapter.

8.3 Plan types

8.3.1 Plan Types folder

Right-clicking on the Plan Types folder results in the appearance of the Plan types options menu.

The operations embodied in the menu choices are common to all folders and are described in generic terms in the *General functions* section of the *JDE browser* chapter.

8.3.2 Plan type definitions

Right-clicking on a plan type definition results in the appearance of the Plan types definition options menu.

Apart from the Edit Graphically option, the operations embodied in the menu choices are common to all type definitions and are described in generic terms in the *General functions* section of the *JDE browser* chapter. The Edit Graphically option allows the user to access the Graphical Plan Editor to edit the plan.

The Graphical Plan Editor is described in detail in the *Graphical Plan Editor Manual*.

8.3.3 Plan type definition attributes

The Extends and Documentation fields of a plan type definition can be modified via pop-up menus which are generated by right-clicking on the appropriate field. These operations are described in generic terms in the *General functions* section of the *JDE browser* chapter.

8.3.4 Java code

Plan type definitions may contain Java code. The JDE browser supports the following operations on Java code within a type definition:

- declaration of interfaces
- declaration of imports
- addition of methods or members.

These operations are described in the *General functions* section of the *JDE browser* chapter.

8.3.5 Type definition references

A plan type definition can contain references to event type definitions. It can also contain references to its enclosing capabilities or agents – this is discussed in a later section.

The following operations on event type definitions are supported within the JDE browser:

- adding the reference
- editing the object
- removing the reference
- editing the reference name
- defining the event type.

These operations are described in the *General functions* section of the *JDE browser* chapter.

Note that since a plan can only ever handle one event, the semantics associated with the dropping of an event on the handles container is different to that associated with other containers. In particular, if the handles event reference already has an event associated with it, dropping another event on the event reference will cause the old event to be replaced with the new event.

Also note that to explicitly remove a handles event reference, the menu selection is reset (there is no Remove selection as with other references).

8.3.6 Data naming element references

The JDE browser supports the following operations on data naming element references:

- adding the reference
- adding parameters
- editing the data naming element
- removing the reference
- defining the access mode.

These operations are described in the *General functions* section of the *JDE browser* chapter.

8.3.7 Plan selection

Operations on plan selection and reasoning methods include the following:

8.3.7.1 Relevance

A JACK plan may contain a `relevant` method. If present, it can be used to examine the actual event instance that attempts to invoke the plan in order to determine whether the plan is relevant for that event.

When a new plan is created, a default `relevant` method is provided. When JACK code is eventually generated for the plan, the default `relevant` method looks something like the following:

```
static boolean relevant(<<HandlesType>> reference)
{
  return true;
}
```

Consequently, by default the plan is relevant to every event instance that invokes it.

8.3.7.2 Editing the relevance method

The relevance method can be edited simply by double-clicking on it, or right-clicking on the method and selecting Edit "Relevance" from the menu.

This action opens an editing window that contains the method. The method can then be edited. The default relevant method only contains `return true;`.

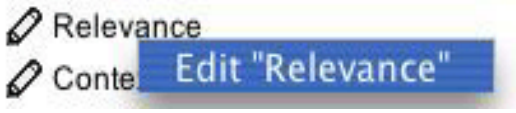

Figure 8-3: Editing the plan's relevance method

The method prototype and braces do not need to be inserted into the definition since they will be automatically inserted into generated code. The actual `EventType` and `reference` that will be generated for the method prototype (if not already present) will be obtained from the `handles` event reference. Consequently, any code added to this method is able to use the event reference name directly to examine the event.

8.3.7.3 Context

The `context` method is used to determine whether the plan is applicable in the current context. It may involve using a number of logical statements and cursors to examine the beliefsets and other data that the agent has access to. Examining this data is analogous to checking the 'state of the world' or the context that the agent is in.

The default `context` method would look like this in a JACK plan:

```
context()
{
  true;
}
```

Note the absence of any type declaration for the method and the fact that there is no `return` statement.

8.3.7.4 Editing the context method

The `context` method can be edited simply by double-clicking on it or by right-clicking on the method and selecting Edit Context from the menu. This action opens an editing window for the method. The method can then be edited. The default `context` method only contains `true;`.

Figure 8-4: Editing the plan's context method

Note: The method prototype and braces do not need to be inserted into the definition since they will be automatically inserted into the generated code.

8.3.8 Reasoning methods

The reasoning methods that define a plan must be added to the Reasoning Methods folder of the plan concerned.

8.3.8.1 Special reasoning methods

A `body()` reasoning method is automatically added when a new plan is created.

Two other special reasoning methods are `pass()` and `fail()`. These can be added to allow explicit actions to be performed on plan success or failure.

8.3.8.2 Adding a reasoning method

To define a reasoning method for use in a plan, right-click on the Reasoning Methods folder and select Add New Reasoning Method. This will add a new reasoning method to the Reasoning Methods folder and allow a label to be entered for it. In addition, this action opens a text window into which the actual reasoning method should be entered. A JACK reasoning method normally begins with a `#reasoning method` modifier. The `#reasoning method` modifier should not be entered when working in the JDE. It is automatically added when the JACK code is generated. However, the method name and any parameters must be entered.

The label is just a useful way of tagging each reasoning method. Normally the label is identical to the method name but it does not have to be.

A new plan will have an empty implementation of the `body()` method (with a label of body) that must be edited before the plan will do anything useful. After the body method has been edited, its icon will be rendered in colour rather than black and white to signify that it has been changed from the default. The default body method is simply:

```
body()
{
}
```

8.3.8.3 Editing a reasoning method label

To edit the label for a reasoning method:

1. Right-click on the label and select Edit Name.
2. Type in a label of choice, or select an insertion point with the mouse to modify the existing name.
3. Press ENTER to complete the change.
4. Press ESCAPE to abort the editing and cause the name to revert to its original state.

Figure 8-5: Reasoning Methods pop-up menu

8.3.8.4 Editing a reasoning method

To edit a reasoning method, right-click on the label and select the Edit option. This opens a text window where the method can be edited.

8.3.8.5 Removing a reasoning method

To remove a reasoning method from a plan, right-click on the reference and select the Remove option. This will remove the reasoning method from the Reasoning Methods folder and thus from the plan.

8.3.9 Enclosing interfaces

8.3.9.1 Adding enclosing interface references

To add an enclosing interface reference, drag an agent, team, or capability type definition, or a Java file (representing an interface) to the Enclosing Interfaces folder of the plan.

When a new enclosing interface reference is created in this manner, a default reference name will be assigned and automatically selected for editing. To change the name, type in a new name and press ENTER when done. The reference name can be used to refer to the enclosing agent, team, capability or interface inside any plan reasoning method.

8.3.9.2 Editing an enclosing interface

As with other references, the original object can be opened and edited by right-clicking on the reference and selecting the Edit option.

8.3.9.3 Removing an enclosing interface

To remove an enclosing interface reference from a plan, right-click on the reference and select the Remove option. This will remove the reference from the plan, leaving the reference object untouched.

Note: For further details on enclosing interfaces, refer to the *Agent Manual*.

8.4 Event types

8.4.1 Event Types folder

Right-clicking on the Event Types folder results in the appearance of the Event types options menu.

The operations embodied in the menu choices are common to all folders and are described in generic terms in the *General functions* section of the *JDE browser* chapter.

8.4.2 Event type definitions

Right-clicking on an event type definition results in the appearance of the Event types definition options menu.

The operations embodied in the menu choices are common to all type definitions and are described in generic terms in the *General functions* section of the *JDE browser* chapter.

8.4.3 Event type definition attributes

The Extends and Documentation fields of an event type definition can be modified via pop-up menus which are generated by right-clicking on the appropriate field. These operations are described in generic terms in the *General functions* section of the *JDE browser* chapter.

8.4.4 Java code

Event type definitions may contain Java code. The JDE browser supports the following operations on Java code within a type definition:

- declaration of interfaces
- declaration of imports
- addition of methods or members.

These operations are described in the *General functions* section of the *JDE browser* chapter.

8.4.5 Data naming element references

The JDE browser supports the following operations on data naming element references:

- adding the reference
- adding parameters
- editing the data naming element
- removing the reference
- defining the access mode.

These operations are described in the *General functions* section of the *JDE browser* chapter. Adding a reference is achieved using drag-and-drop; the permissible operations are summarised in the next section:

8.4.6 Operations on event fields

8.4.6.1 Adding event fields

To add a new field to an event:

1. Right-click on the Fields folder.
2. Select Add New Field.
3. An event field will appear in the folder denoted by the event field icon.

Note that the label that appears beside the event field is of the form type:field-name.

An event field can also be added by right-clicking on an existing Event field and selecting Add new field.

8.4.6.2 Editing event field names

To change the name of an event field, right-click on the event field icon and select Edit name. Type in the new event field name and press ENTER.

Pressing the ESCAPE key at any point during this editing operation will abort the operation and cause the name to revert to its previous state.

8.4.6.3 Changing the event field type

To change the event field type, click on the icon adjacent to the event field icon to expand the tree below the event field. Right-clicking on the type attribute elicits a pop-up menu, at which point the user can either select the type they would like for the event field, or select Edit Value to specify their own.

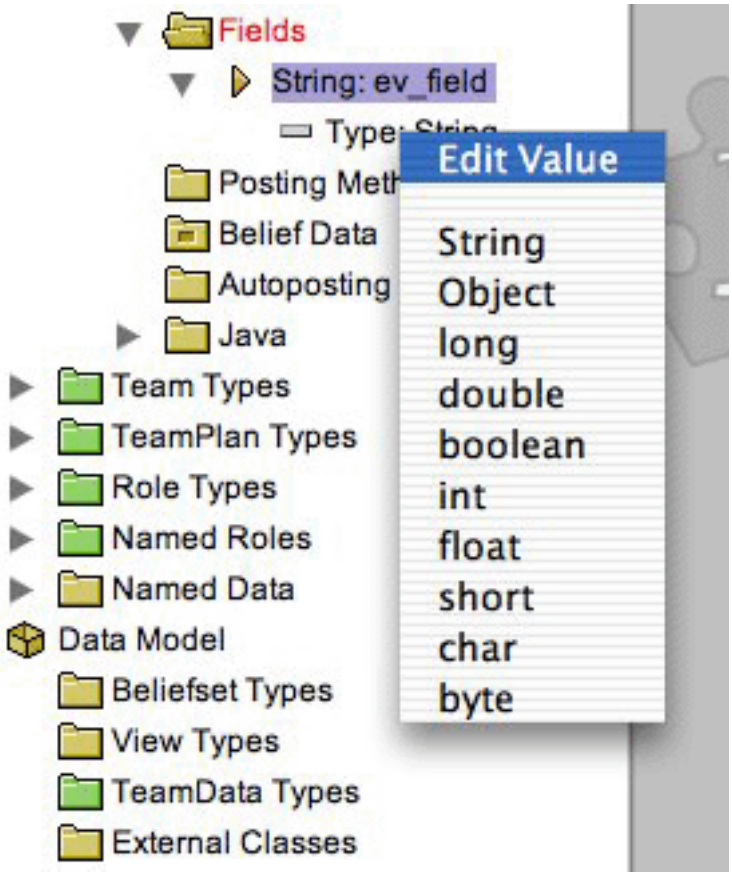

Figure 8-6: Event field type menu

8.4.6.4 Removing event fields

To remove an event field, right-click on the event field icon and select the Remove option.

8.4.7 Operations on posting methods

8.4.7.1 Adding posting methods

To add a new posting method, right-click on the Posting Methods folder and select Add New Posting Method. Alternatively, the user can right-click on an existing posting method and select Add New Posting Method. A new posting method will appear in the Posting Methods folder denoted by the icon illustrated below.

Figure 8-7: Posting method icon

If an external text editor has not been specified in the JDE preferences, a Code Editor window will appear into which the text of the posting method can be entered. When the text of the posting method has been entered, click the Done button, and then type the posting method's name into the text box that is adjacent to the posting method icon. If no name is entered, the JDE will name the posting method with the first few characters of the text that was typed into the Code Editor window.

8.4.7.2 Editing posting method names

To change the name of a posting method:

1. Right-click on the posting method icon and select Edit Name.
2. Type in the new posting method name and press ENTER. Pressing the ESCAPE key at any point during the editing operation will cause an abort and reversion to the previous state.

8.4.7.3 Editing posting methods

To edit a posting method, right-click on the posting method's icon and select Edit Object. This will open an editor window containing the existing code for the posting method.

8.4.7.4 Removing posting methods

To remove a posting method from an event, right-click on the posting method and select the Remove option.

8.4.8 Autoposting conditions

JACK allows a set of conditions to be specified (which can refer to any belief data and any fields of the event) that will cause the event to be automatically posted when any of the conditions becomes true. The conditions are essentially normal JACK logical expressions, i.e. they are composed of booleans and/or cursors.

In JACK they are specified as follows:

```
#posted when ( <condition> );
```

and

```
#posted when ( <condition> ) { <statements> }
```

Note: When entering these expressions in the JDE, the `#posted when` and the braces should not be typed in, as they will be added to the generated JACK code automatically.

8.4.8.1 Adding autoposting conditions

To add an autoposting condition to an event, right-click on the AutoPosting Conditions folder and select the Add New Condition menu item. A new autoposting condition will appear in the Posting Methods folder denoted by the icon illustrated below.

Figure 8-8: Autoposting condition icon

If an external text editor has not been specified in the JDE preferences, a Code Editor window into which the text of the autoposting condition can be entered will appear. When the text of the autoposting condition has been entered, click on the Done button and then type the condition's name into the text box that is adjacent to its icon. If a name is not entered, the JDE will name the condition with the first few characters of the text that were typed into the Code Editor window.

8.4.8.2 Editing autoposting condition names

To change the name of an autoposting condition:

1. Right-click on the autoposting condition icon and select Edit Name.
2. Type in the new autoposting condition name and press ENTER.
3. Pressing the ESCAPE key at any point during this editing operation will abort the operation and cause the name to revert to its previous state. This name is only for reference, and appears in a comment above the generated code.

8.4.8.3 Editing autoposting conditions

To edit the autoposting condition, right-click on the autoposting condition icon and select Edit Object. This will open a Code Editor window containing the existing code for the autoposting condition.

8.4.8.4 Removing AutoPosting Conditions

To remove an autoposting condition from an event, right-click on the autoposting condition and select the Remove option.

8.5 Named data

8.5.1 Named Data folder

Right-clicking on the Named Data folder results in the appearance of the Named data options menu.

The operations embodied in the menu choices are common to all folders and are described in generic terms in the *General functions* section of the *JDE browser* chapter.

When the user is creating a data naming element, the pop-up window requests a type, rather than a package name.

8.5.2 Data naming elements

Right-clicking on a data naming element results in the appearance of the Data naming element menu. The operations embodied in the menu choices are common to all naming elements and are described in generic terms in the *General functions* section of the *JDE browser* chapter.

When the user is editing the data naming element's details the pop-up window displays a type for modification (when the user edits a type definition, a package name is displayed for modification).

8.5.3 Data naming element attributes

The Documentation field of a data naming element can be modified via a pop-up menu which is generated by right-clicking on the appropriate field. This operation is described in generic terms in the *General functions* section of the *JDE browser* chapter.

8.6 Team types

8.6.1 Team Types folder

Right-clicking on the Team Types folder results in the appearance of the following menu:

Figure 8-9: Team types options menu

The operations embodied in the menu choices are common to all folders and are described in generic terms in the *General functions* section of the *JDE browser* chapter.

8.6.2 Team type definitions

Right-clicking on a team type definition results in the appearance of the following menu:

Figure 8-10: Team types definition options menu

The operations embodied in the menu choices are common to all type definitions and are described in generic terms in the *General functions* section of the *JDE browser* chapter.

8.6.3 Team type definition attributes

The Extends, Documentation and Constructor fields of a team type definition can be modified via pop-up menus which are generated by right-clicking on the appropriate field. These operations are described in generic terms in the *General functions* section of the *JDE browser* chapter.

8.6.4 Java code

Team type definitions may contain Java code. The JDE browser supports the following operations on Java code within a type definition:

- declaration of interfaces
- declaration of imports
- addition of methods or members.

These operations are described in the *General functions* section of the *JDE browser* chapter.

8.6.5 Type definition references

A team type definition can contain references to the following type definitions:

- capabilities

- events
- teamplans
- plans
- roles (for `#performs` relationships).

The JDE browser supports the following operations which are applicable to all of the above references:

- adding the reference
- editing the type definition
- removing the reference
- editing the reference name (capabilities and events only)
- defining the reference type (events only).

All of the above operations are described in the *General functions* section of the *JDE browser* chapter.

8.6.6 Data naming element references

The JDE browser supports the following operations on data naming element references:

- adding the reference
- adding parameters
- editing the data naming element
- removing the reference
- defining the access mode.

These operations are described in the *General functions* section of the *JDE browser* chapter.

8.6.7 Role naming element references

A team type definition may also contain references to role type definitions via role naming elements (for `#requires` relationships).

The JDE browser supports the following operations on role naming element references:

- adding the reference
- editing the specification
- editing the role naming element
- removing the reference.

These operations are described in the *General functions* section of the *JDE browser* chapter.

8.7 Teamplan types

8.7.1 TeamPlan Types folder

Right-clicking on the TeamPlan Types folder results in the appearance of the following menu:

Figure 8-11: Teamplan types options menu

The operations embodied in the menu choices are common to all folders and are described in generic terms in the *General functions* section of the *JDE browser* chapter.

8.7.2 Teamplan type definitions

Right-clicking on a teamplan type definition results in the appearance of the following menu:

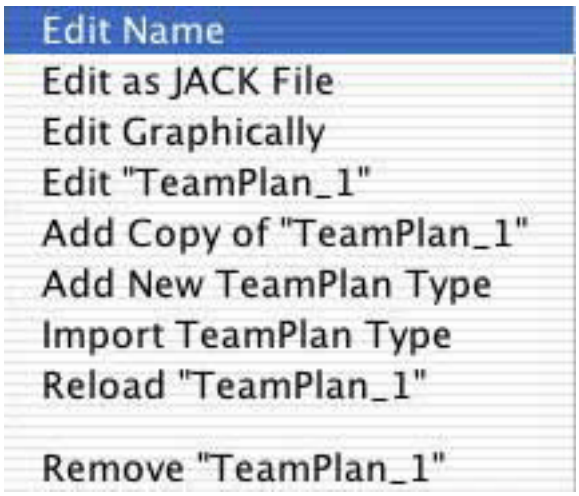

Figure 8-12: Teamplan types definition options menu

The operations embodied in the menu choices are common to all type definitions and are described in generic terms in the *General functions* section of the *JDE browser* chapter.

8.7.3 Teamplan type definition attributes

The Extends and Documentation fields of a teamplan type definition can be modified via pop-up menus which are generated by right-clicking on the appropriate field.

These operations are described in generic terms in the *General functions* section of the *JDE browser* chapter.

8.7.4 Java code

Teamplan type definitions may contain Java code. The JDE browser supports the following operations on Java code within a type definition:

- declaration of interfaces
- declaration of imports
- addition of methods or members.

These operations are described in the the *General functions* section of the *JDE browser* chapter.

8.7.5 Type definition references

A teamplan type definition can contain references to event type definitions and role type definitions. (It can also contain references to its enclosing capabilities or team – this is discussed in a later section).

When a teamplan is defined in the JDE, a default `establish()` method is generated for the teamplan. This calls `defaultEstablish()` which fills the required roles at random, if possible.

The following operations on event type and role type definition references contained in a teamplan type definition are supported by the JDE browser:

- adding the reference
- editing the object
- removing the reference
- editing the reference name
- defining the posting type of an event reference.

These operations are described in the *General functions* section of the *JDE browser* chapter. In addition, support is provided for defining the applicability type of a role reference.

To define the applicability of a role usage (*applicable for* or *applicable from*), click on the icon beside the role reference to expand the tree below (or alternatively, double-click on the reference). Right-clicking on the Attribute field will result in a pop-up menu of choices for the applicability type of the role reference. Select the required applicability type.

Note that since a teamplan can only ever handle one event, the semantics associated with the dropping of an event on the handles container is different to that associated with other containers. In particular, if the handles event reference already has an event associated with it, dropping another event on the event reference will cause the old event to be replaced with the new event. Also note that to explicitly remove a handles event reference, the menu selection is reset (there is no remove selection as with other references). These comments also apply to the dropping of a role on the applicable for container.

8.7.6 Naming element references

The JDE browser supports the following operations on naming element references contained in a teamplan type definition:

- adding the reference
- defining the access mode (for data naming elements only)
- editing the naming element
- removing the reference.

These operations are described in the *General functions* section of the *JDE browser* chapter. In addition, support is provided for:

- editing the specification for a role usage declaration
- defining the type of a role usage declaration.

To edit the specification for the role usage declaration, right-click on the reference and select Edit Specification from the pop-up menu.

To define the type of role usage (uses or requires):

1. Click on the icon beside the role naming element reference to expand the tree below. Alternatively, double-click on the reference.
2. Right-clicking on the Attribute field will result in a pop-up menu of choices for the role usage attribute.
3. Select the required role usage.

8.7.7 Teamplan selection and reasoning methods

Operations on teamplan selection and reasoning methods are the same as those for plan types. The only difference is that teamplans may also include an `establish()` reasoning method that defines how the task team is to be established.

8.7.8 Enclosing interfaces

Teamplan enclosing interfaces are the same as for plans.

8.8 Role types

8.8.1 Role Types folder

Right-clicking on the Role Types folder results in the appearance of the following menu:

Figure 8-13: Role types options menu

The operations embodied in the menu choices are common to all folders and are described in generic terms in the *General functions* section of the *JDE browser* chapter.

8.8.2 Role type definitions

Right-clicking on a role type definition results in the appearance of the following menu:

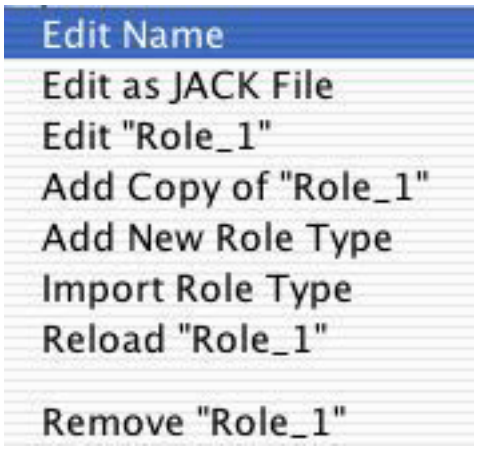

Figure 8-14: Role types definition menu

The operations embodied in the menu choices are common to all type definitions and are described in generic terms in the *General functions* section of the *JDE browser* chapter.

8.8.3 Role type definition attributes

The Extends and Documentation fields of a role type definition can be modified via pop-up menus which are generated by right-clicking on the appropriate field. These operations are described in generic terms in the *General functions* section of the *JDE browser* chapter.

8.8.4 Java code

Role type definitions may contain Java code. The JDE browser supports the following operations on Java code within a type definition:

- declaration of interfaces
- declaration of imports
- addition of methods or members.

These operations are described in the *General functions* section of the *JDE browser* chapter.

8.8.5 Type definition references

A role type definition can contain references to event type definitions.

The JDE browser supports the following operations on event type definition references:

- adding the reference
- editing the type definition

- removing the reference
- editing the reference name
- defining the event type.

These operations are described in the *General functions* section of the *JDE browser* chapter.

8.8.6 Data naming element references

The JDE browser supports the following operations on data naming element references:

- adding the reference
- adding parameters
- editing the data naming element
- removing the reference
- defining the access mode.

These operations are described in the *General functions* section of the *JDE browser* chapter.

8.8.7 Role containers

The JACK Compiler generates two classes from a role definition. The first class is the `RoleType`. The second generated class is a specialised container for instances of the `RoleType` called `RoleTypeContainer`. This latter class is used in teams and teamplans to group the performers of a role. The generated `RoleTypeContainer` class extends a base class named `RoleContainer`. This base class provides a number of useful methods for inspecting the container and accessing the role performers. Additional methods and members that are to be added to the derived container class are defined within the role type definition and can be added to the role type definition using the JDE browser. The process is described below.

8.8.8 Operations on container members

8.8.8.1 Adding a new member

To add a new member to a role type definition:

1. Expand the appropriate role type definition entry in the Role Types folder.
2. Right-clicking on the Container Members folder for the entry results in a pop-up menu with one option – Add New Member.
3. Selecting this option results in the addition of a new container member with a default type of `String` and a default name generated by the JDE browser. The member name can be modified at this point if desired.
4. To change the default type, first expand the member entry to expose the member attributes.

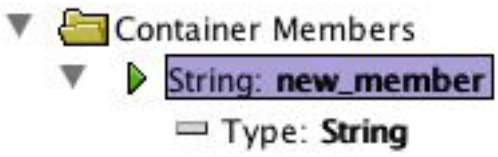

Figure 8-15: New role type container member

Right-clicking on the type attribute results in the Edit Value menu being displayed.

To change the type select the appropriate choice from this menu. The Add New Member option is also available from the menu that is obtained by right-clicking on an existing member.

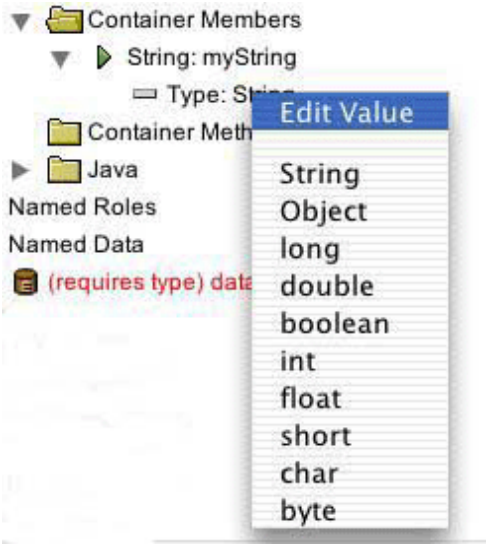

Figure 8-16: Role type Edit Value menu

8.8.8.2 Editing the member name

To edit the name of an existing member, right-click on the member and select the Edit Name option from the resulting menu.

8.8.8.3 Removing the member

To remove a member, right-click on the member and select the Remove option from the resulting menu.

8.8.9 Operations on container methods

8.8.9.1 Adding a new method

To add a new container method, right-click on the Container Methods folder and select Add New Method from the resulting menu. This will result in the creation of an editor window (for entering the method) and an edit box in the Container Methods folder (for specifying a label for the method). Note that the method label is distinct from the method name. The label can be (and is normally) set by the user to the method name, but it can also be set to any other string.

Note: The Add New Method option is also available from the menu that is obtained by right-clicking on an existing method.

8.8.9.2 Editing the method label

To edit the label of an existing method, right-click on the method label and select the Edit Name option from the resulting menu.

8.8.9.3 Editing the method

To edit an existing method:

1. Right-click on the method label and select the Edit option from the resulting menu.
2. An editor window containing the code for the method will then be created.
3. The desired changes can now be made.

8.8.9.4 Removing the method

To remove a method, right-click on the method label and select the Remove option from the ensuing menu.

8.9 Named roles

8.9.1 Named Roles folder

Right-clicking on the Named Roles folder results in the appearance of the following menu:

Figure 8-17: Named roles options menu

The operations embodied in the menu choices are common to all folders and are described in generic terms in the *General functions* section of the *JDE browser* chapter.

When the user is creating a role naming element, the pop-up window requests a role type (when the user creates a type definition, a package name is requested).

8.9.2 Role naming elements

Right-clicking on a role naming element results in the appearance of the following menu:

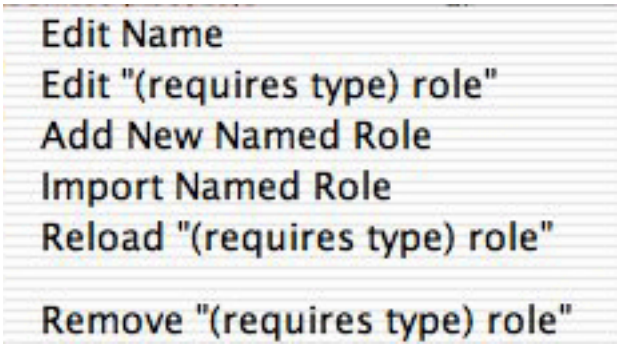

Figure 8-18: Role naming element options menu

The operations embodied in the menu choices are common to all naming elements and are described in generic terms in the *General functions* section of the *JDE browser* chapter.

When the user is editing the role naming element's details, the pop-up window requests a type for modification, rather than a package name.

8.9.3 Role naming element attributes

The documentation field of a role naming element can be modified via a pop-up menu which is generated by right-clicking on the Documentation field. This operation is described in generic terms in the *General functions* section of the *JDE browser* chapter.

9 JDE browser: Data Model

9.1 Beliefset types

9.1.1 BeliefSet Types folder

Right-clicking on the BeliefSet Types folder results in the appearance of the following menu:

Figure 9-1: Beliefset types options menu

The operations embodied in the menu choices are common to all folders and are described in generic terms in the *General functions* section of the *JDE browser* chapter.

9.1.2 Beliefset type definitions

Right-clicking on a beliefset type definition results in the appearance of the following menu:

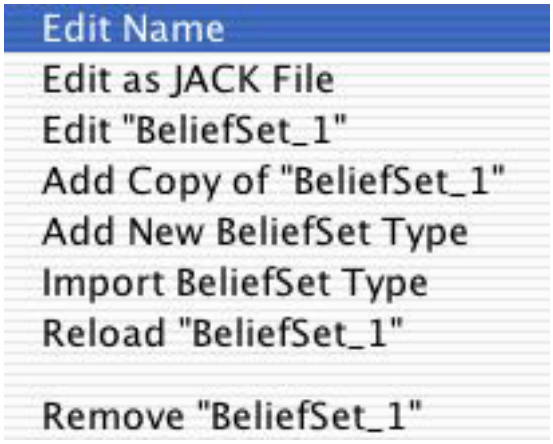

Figure 9-2: Beliefset types definition menu

The operations embodied in the menu choices are common to all type definitions and are described in generic terms in the *General functions* section of the *JDE browser* chapter.

9.1.3 Beliefset type definition attributes

The Extends and Documentation fields of a beliefset type definition can be modified via pop-up menus which are generated by right-clicking on the appropriate field. These operations are described in generic terms in the *General functions* section of the *JDE browser* chapter.

In Teams mode, a beliefset type will contain a Propagation field. To edit the Propagation field of a beliefset type:

1. Right-click on the Propagation field to pop up a menu.
2. Select Edit Value, or none, or aos.team.PropagationEvent. The default is none. Selecting aos.team.PropagationEvent means that changes to the beliefset can be propagated in a teams application.

9.1.4 Java code

Beliefset type definitions may contain Java code. The JDE browser supports the following operations on Java code within a type definition:

- declaration of interfaces
- declaration of imports
- addition of methods or members.

These operations are described in the *General functions* section of the *JDE browser* chapter.

9.1.5 Type definition references

A beliefset type definition can contain references to event type definitions. The JDE browser supports the following operations on event type definitions contained in a beliefset type definition:

- adding the reference
- editing the type definition
- removing the reference
- editing the reference name
- defining the posting type of the event reference.

The operations are described in the *General functions* section of the *JDE browser* chapter.

9.1.6 Editing beliefset fields

To add a new field to a beliefset type definition, right-click on the Fields folder and select Add New Field. A Beliefset field will appear in the folder.

Figure 9-3: Beliefset field icon

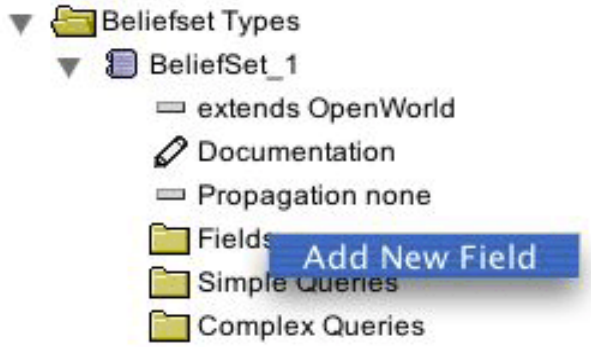

Figure 9-4: Adding a new field to a beliefset

The label that appears beside the beliefset field is of the form `key type: field-name`. The `key` portion is present if the field is a key field.

Figure 9-5: Beliefset field labels

A beliefset field can also be added by right-clicking on an existing beliefset field and selecting Add New Field.

Figure 9-6: Beliefset field label pop-up menu

9.1.6.1 Editing beliefset field names

To change the name of a beliefset field:

1. Right-click on the Beliefset Field icon.
2. Select Edit Name.
3. Type in the new beliefset field name and press ENTER.
4. Pressing the ESCAPE key at any point during this editing operation will abort the operation and cause the name to revert to its previous state.

9.1.6.2 Changing the beliefset field type

To change the beliefset field type, expand the tree below the beliefset field. Right-clicking on the type attribute pops up a menu, at which point the user can either select a type for the beliefset field, or select Edit Value to specify their own unique value.

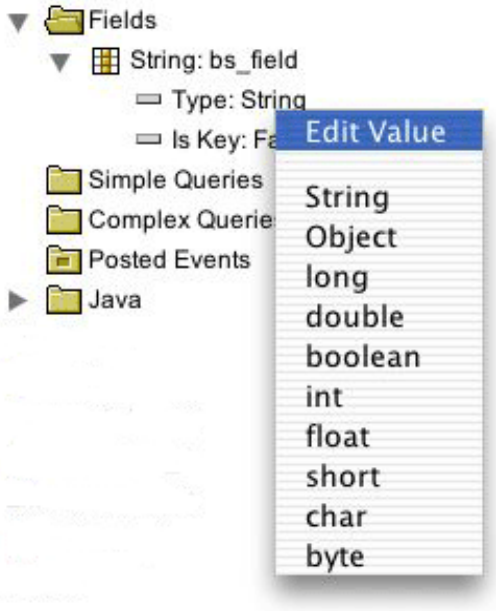

Figure 9-7: Beliefset field type menu

9.1.6.3 Changing the beliefset field Is Key value

To change the beliefset field Is Key value,expand the tree below the beliefset field. Right-clicking on the Is Key attribute pops up a menu, and the user can then select either True or False.

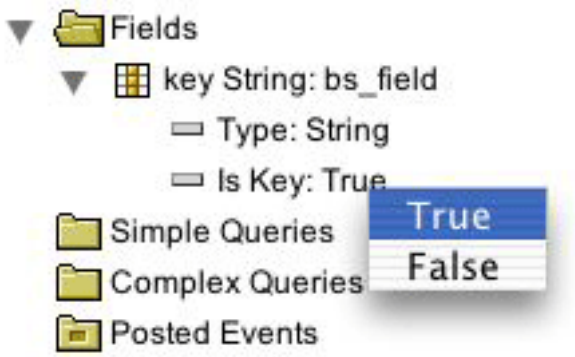

Figure 9-8: Beliefset field Is Key menu

9.1.6.4 Removing beliefset fields

To remove a beliefset field, right-click on the Beliefset Field icon and select the Remove option.

9.1.7 Editing beliefset simple queries

To add a new simple query to a beliefset:

1. Right-click on the Simple Queries folder.
2. Select Add New Query.
3. Enter a preferred name in the text box that appears beside the icon for the newly created query. A simple query will appear in the folder, denoted by the icon illustrated below.

Figure 9-9: Simple Query icon

Figure 9-10: Adding a new simple query to a beliefset

The label that appears beside the simple query is of the form indexed query_name(), where query_name is the name that was entered. The parentheses will be filled out when the output fields for the query are entered.

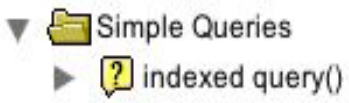

Figure 9-11: New simple query

A simple query can also be added by right-clicking on an existing simple query and selecting Add New Query.

Note: Within the JDE, simple queries refer to JACK indexed or linear queries.

Figure 9-12: Simple Query label pop-up menu

9.1.7.1 Editing simple query names

To change the name of a simple query:

1. Right-click on the Simple Query icon and select Edit Name.
2. Type in the new query name and press ENTER.
3. Pressing the ESCAPE key at any point during this editing operation will abort the operation and cause the name to revert to its previous state.

9.1.7.2 Removing a simple query

To remove a simple query, right-click on the query and select the Remove option.

9.1.7.3 Changing the query type

To change the query type, expand the tree below the simple query. Right-clicking on the Query Type attribute pops up the menu, at which point the user can select either indexed or linear. Generally, you should use indexed query methods because they provide faster access.

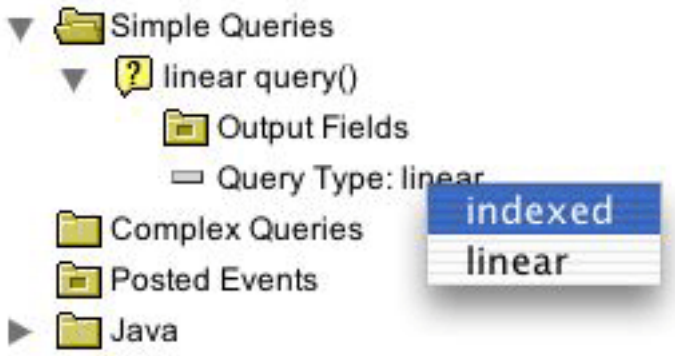

Figure 9-13: Query type pop-up menu

9.1.7.4 Adding the output fields

To define what is returned as the result of a query you must define the output fields for the query. To do this, simply drag fields from the Fields folder of the beliefset and drop them on the Output Fields target folder of the query.

9.1.7.5 Removing simple query output fields

To remove an output field, right-click on an output field and select the Remove option.

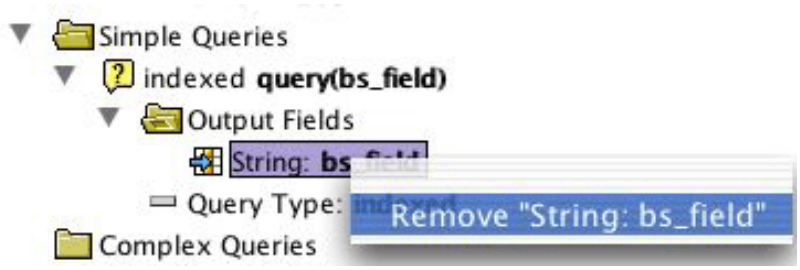

Figure 9-14: Removing an output field

9.1.8 Editing function and complex queries

To add a new complex or function query to a beliefset:

1. Right-click on the Complex Queries folder and select Add New Complex Query.
2. Enter the preferred name in the text box that appears beside the icon for the newly created query.
3. A complex query will appear in the folder, denoted by the icon shown below.

Figure 9-15: Complex Query icon

Figure 9-16: Adding a new Complex Query to a beliefset

Note: Within the JDE, complex queries refer to both complex and function queries. Function queries are defined as a special case of complex queries.

The label that appears beside the complex query is of the form complex query_name, where query_name is the name that was entered.

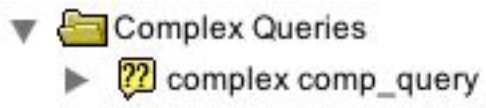

Figure 9-17: New Complex Query

A complex query can also be added by right-clicking on an existing complex query and selecting Add New Complex Query.

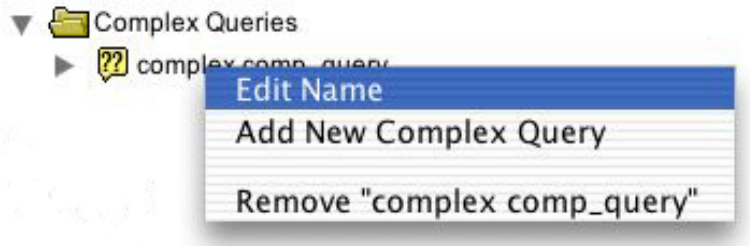

Figure 9-18: Complex Query label pop-up menu

9.1.8.1 Editing complex query names

To change the name of a complex query:

1. Right-click on the Complex Query icon and select Edit Name.
2. Type in the new query name and press ENTER.
3. Pressing the ESCAPE key at any point during this editing operation will abort the operation and cause the name to revert to its previous state.

9.1.8.2 Removing a complex query

To remove a complex query, right-click on the query and select the Remove option.

9.1.8.3 Changing the query type

To change the query type, expand the tree below the complex query. Right-clicking on the Query Type attribute pops up the menu, at which point the user can select either function or complex.

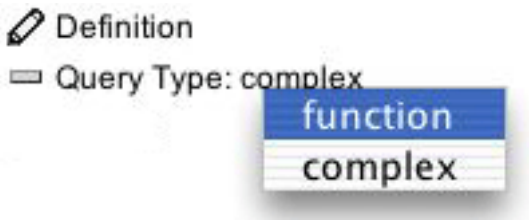

Figure 9-19: Complex Query Type pop-up menu

9.1.8.4 Editing a complex query definition

Adding a complex query definition requires adding code using the Code Editor window that appears when the query is double-clicked or the Edit option is selected from the pop-up menu. The code entered as the definition of the complex query is copied directly into the generated code. This includes the method name, its parameters and any return value.

9.2 View types

9.2.1 View Types folder

Right-clicking on the View Types folder results in the appearance of the following menu:

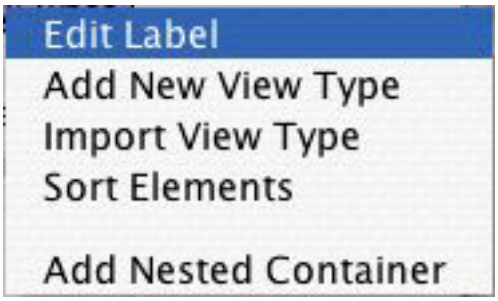

Figure 9-20: View types options menu

The operations embodied in the menu choices are common to all folders and are described in generic terms in the *General functions* section of the *JDE browser* chapter.

9.2.2 View type definitions

Right-clicking on a view type definition results in the appearance of the following menu:

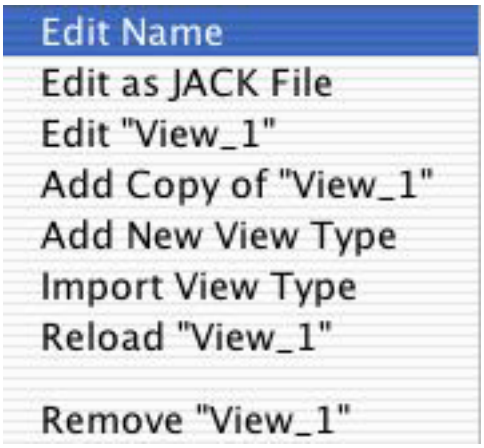

Figure 9-21: View types definition menu

The operations embodied in the menu choices are common to all type definitions and are described in generic terms in the *General functions* section of the *JDE browser* chapter.

9.2.3 View type definition attributes

The Extends and Documentation fields of a view type definition can be modified via pop-up menus which are generated by right-clicking on the appropriate field. These operations are described in generic terms in the *General functions* section of the *JDE browser* chapter.

9.2.4 Java code

View type definitions may contain Java code. The JDE browser supports the following operations on Java code within a type definition:

- declaration of interfaces
- declaration of imports
- addition of methods or members.

These operations are described in the *General functions* section of the *JDE browser* chapter.

9.2.5 Data naming element references

A view type definition can contain references to:

- beliefset type definitions
- view type definitions
- external classes

via data naming elements.

The JDE browser supports the following operations on data naming element references:

- adding the reference
- adding parameters
- editing the data naming element
- removing the reference
- defining the access mode.

These operations are described in the *General functions* section of the *JDE browser* chapter.

9.2.6 Editing function and complex queries

To add a new complex or function query to a View type definition:

1. Right-click on the Complex Queries folder and select Add New Complex Query.
2. Enter a preferred name in the text box that appears beside the icon for the newly created query. A complex query will appear in the folder, denoted by the Complex Query icon.

Figure 9-22: Add New Complex Query to a view

The label Complex appears beside the query.

A complex query can also be added by right-clicking on an existing complex query and selecting Add New Complex Query.

Note: Within the JDE, complex queries refer to both complex and function queries. Function queries are defined as a special case of complex queries.

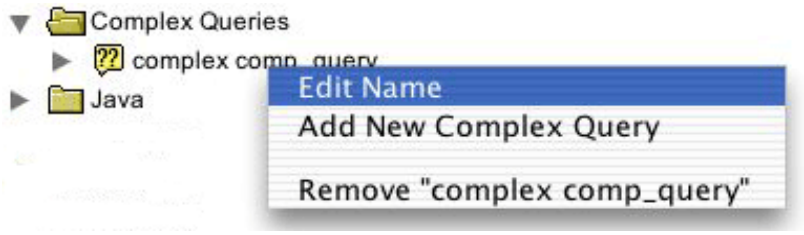

Figure 9-23: Complex query label pop-up menu

9.2.6.1 Editing complex query names

To change the name of a complex query:

1. Right-click on the Complex Query icon and select Edit Name.
2. Type in the new query name and press ENTER.
3. Pressing the ESCAPE key at any point during this editing operation will abort the operation and cause the name to revert to its previous state.

9.2.6.2 Removing a complex query

To remove a complex query, right-click on the query and select Remove.

9.2.6.3 Changing the query type

To change the query type:

1. Click on the icon adjacent to the Complex Query icon to expand the tree below the complex query.
2. Right-clicking on the Query Type attribute pops up a menu, at which point the user may select either Function or Complex.

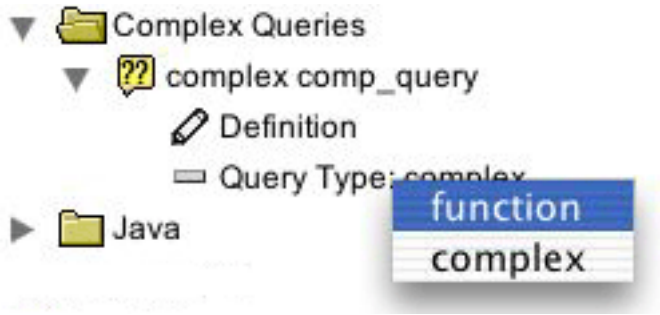

Figure 9-24: Complex query type pop-up menu

9.2.6.4 Editing a complex query definition

Adding a complex query definition requires adding code using the Code Editor window that appears when the query is double-clicked or Edit is selected from the pop-up menu. The code entered as the definition of the query is copied directly into the generated code. This includes the method name, its parameters and any return value.

Figure 9-25: Complex query definition menu

9.3 Teamdata types

9.3.1 TeamData Types folder

Figure 9-26: Teamdata types options menu

Right-clicking on the TeamData Types folder results in the appearance of the menu above.

The operations embodied in the menu choices are common to all folders and are described in generic terms in the *General functions* section of the *JDE browser* chapter.

9.3.2 Teamdata type definitions

Right-clicking on a teamdata type definition results in the appearance of the following menu:

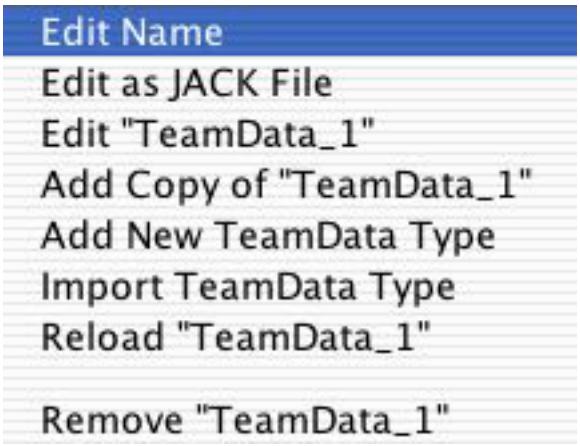

Figure 9-27: Teamdata types definition menu

The operations embodied in the menu choices are common to all type definitions and are described in generic terms in the *General functions* section of the *JDE browser* chapter.

9.3.3 Teamdata type definition attributes

The Extends, Documentation, Connection and Synthesis fields of a teamdata type definition can be modified via pop-up menus which are generated by right-clicking on the appropriate field.

These operations are described in generic terms in the *General functions* section of the *JDE browser* chapter.

9.3.4 Java code

Teamdata type definitions may contain Java code. The JDE browser supports the following operations on Java code within a type definition:

- declaration of interfaces
- declaration of imports
- addition of methods or members.

These operations are described in the *General functions* section of the *JDE browser* chapter.

9.4 External classes

The External Classes folder gives the user the ability to import classes defined outside the project.

9.4.1 External Classes folder

Right-clicking on the External Classes folder results in the appearance of the following menu:

Figure 9-28: External class options menu

The operations embodied in the menu choices are common to all folders and are described in generic terms in the *General functions* section of the *JDE browser* chapter.

9.4.2 External class operations

Right clicking on an external class entry results in the appearance of the following menu:

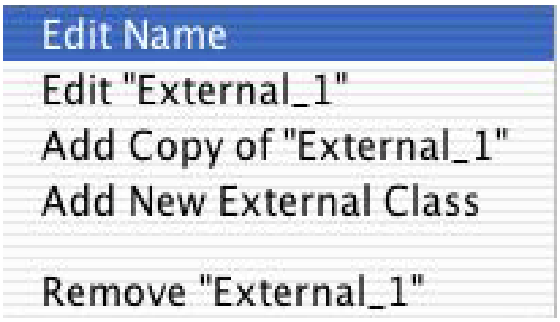

Figure 9-29: External class entry options menu

The operations embodied in the menu choices are common to all type definitions and are described in generic terms in the *General functions* section of the *JDE browser* chapter.

9.4.3 External class attributes

The Documentation field of the external class definition can be modified via a pop up menu which is generated by right-clicking on the Documentation field. This operation is described in the *General functions* section of the *JDE browser* chapter.

9.4.4 Java code

The package of the external class can be specified in the Java folder. This is achieved by opening the folder, right-clicking on the Package attribute and editing the value. Note that for external classes the package name should be prefixed with a hyphen.

10 Project details

10.1 Project structure

The details of a project appear as the topmost item in the project window.

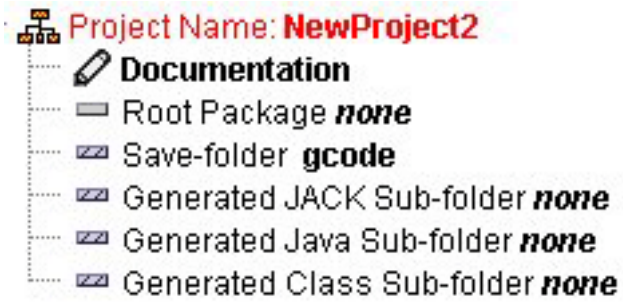

Figure 10-1: Project window

10.2 Project Name

The Project Name corresponds to the project file for the project. For example, if the name of a project is `example` the project file will be `example.prj`. To edit the name, right-click on that field, select Edit Name and enter the new name.

Figure 10-2: Editing the Project Name

10.3 Project documentation

To edit the project documentation, right-click on the Documentation field and type the documentation into the Code Editor window that appears. The label for this field can be changed by right-clicking on the field, selecting Edit Name, and typing in a new label. The field then appears as the Doc: option. If a space before the label is entered, the Doc: prefix and the space will be omitted from the label.

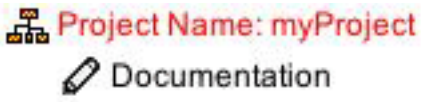

Figure 10-3: Project Documentation field

10.4 Root Package

The Root Package is the top-level Java package that all project components belong to. If the Root Package of the project is edited, the JDE will warn that the package name must match the directory structure. If it does not, the application may fail to compile. Any Java files affected by this change will have to have their package statement edited manually, with either the Code Editor or an external text editor.

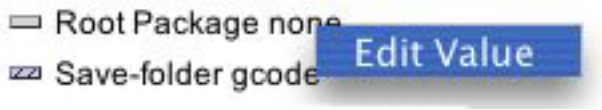

Figure 10-4: Editing the Root Package

10.5 Save-folder

G-Code is the graphical file format used by the JDE and is stored in the Save-folder. Right-click on the field to edit it, select Edit Value and enter the new value into the text box. Because the G-Code will be moved as a result of this change, the JDE pops up a confirmation dialog box.

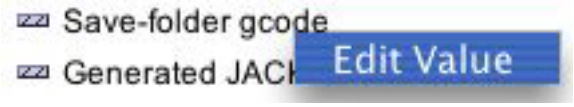

Figure 10-5: Editing the Save-folder

10.6 Generated JACK Sub-folder

This is where the JDE will generate the JACK files representing the components of the project. Right-click on this field and select Edit Value to edit it.

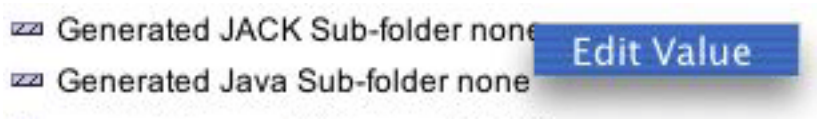

Figure 10-6: Editing the Generated JACK Sub-folder field

10.7 Generated Java Sub-folder

This is where the JACK compiler stores the Java files generated while compiling the JACK files. Right-click on this field and select Edit Value to edit it.

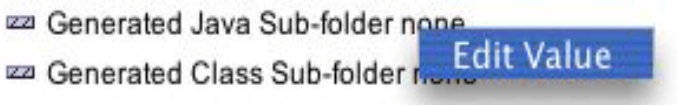

Figure 10-7: Editing the Generated Java Sub-folder field

10.8 Generated Class Sub-folder

This is where the Java compiler stores the class files generated from the Java files. Right-click on this field and select Edit Value to edit it.

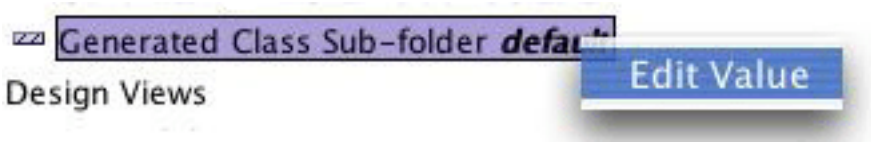

Figure 10-8: Editing the Generated Class Sub-folder field

Appendix A: Keyboard Shortcuts

The JDE provides numerous keyboard shortcuts for the user. These are categorised (according to location) and listed below in the following tables.

Note: On the Macintosh, an extra menu exists called JACK Developer. In all instances on the Macintosh, the COMMAND key replaces the CTRL key when using shortcuts.

Shortcut	Function
COMMAND+,	Preferences
COMMAND+H	Hide Preferences
ALT+COMMAND+H	Hide Others
COMMAND+Q	Quit JACK Developer

Table A-1: Macintosh shortcuts on the JACK Developer menu

Shortcut	Function
CTRL+N	New Project
CTRL+O	Open Project
CTRL+S	Save Project
CTRL+SHIFT+L	Load External Edits
CTRL+SHIFT+U	Update JACK Files
CTRL+SHIFT+G	Generate Project Report
CTRL+SHIFT+P	Page Setup...
CTRL+P	Print Project Report...

Table A-2: Shortcuts located on the File menu

Shortcut	Function
CTRL+X	Cut Text
CTRL+C	Copy Text
CTRL+V	Paste Text

Table A-3: Shortcuts located in the Edit menu

Shortcut	Function
CTRL+ALT+D	Add Design...
CTRL+ALT+A	Add Agent...
CTRL+ALT+C	Add Capability...
CTRL+ALT+P	Add Plan...
CTRL+ALT+E	Add Event...
CTRL+ALT+SHIFT+T	Add Team...
CTRL+ALT+SHIFT+P	Add TeamPlan...
CTRL+ALT+SHIFT+R	Add Role...
CTRL+ALT+SHIFT+N	Add Named Role...
CTRL+ALT+N	Add Named Data...
CTRL+ALT+B	Add Beliefset...
CTRL+ALT+V	Add View...
CTRL+ALT+SHIFT+D	Add TeamData...
CTRL+ALT+X	Add External Class...

Table A-4: Shortcuts located on the Entity menu

Shortcut	Function
ALT+SHIFT+N	Connect to Nameserver...
ALT+SHIFT+C	Connect to Portal...
ALT+SHIFT+D	Configure Design Tracing...
ALT+SHIFT+T	Design Tracing Controller

Table A-5: Shortcuts located on the Trace menu

Shortcut	Function
CTRL+SHIFT+C	Compiler Utility
CTRL+SHIFT+D	Design Palette
CTRL+SHIFT+E	Plan Editor Palette
CTRL+SHIFT+R	Error Log

Table A-6: Shortcuts located in the Tools menu

Sorted by function

Function	Shortcut
Hide Others	ALT+COMMAND+H
Hide Preferences	COMMAND+H
Preferences	COMMAND+,
Quit JACK Developer	COMMAND+Q

Table A-7: Macintosh shortcuts on the JACK Developer menu

Function	Shortcut
Add Agent...	CTRL+ALT+A
Add Beliefset...	CTRL+ALT+B
Add Capability...	CTRL+ALT+C
Add Design...	CTRL+ALT+D
Add Event...	CTRL+ALT+E
Add External Class...	CTRL+ALT+X
Add Named Data...	CTRL+ALT+N
Add Named Role...	CTRL+ALT+SHIFT+N
Add Plan...	CTRL+ALT+P
Add Role...	CTRL+ALT+SHIFT+R
Add Team...	CTRL+ALT+SHIFT+T
Add TeamData...	CTRL+ALT+SHIFT+D

Function	Shortcut
Add TeamPlan...	CTRL+ALT+SHIFT+P
Add View...	CTRL+ALT+V
Compiler Utility	CTRL+SHIFT+C
Configure Design Tracing...	ALT+SHIFT+D
Connect to Nameserver...	ALT+SHIFT+N
Connect to Portal...	ALT+SHIFT+C
Copy Text	CTRL+C
Cut Text	CTRL+X
Design Tracing Controller	ALT+SHIFT+T
Design Palette	CTRL+SHIFT+D
Error Log	CTRL+SHIFT+R
Generate Project Report	CTRL+SHIFT+G
Load External Edits	CTRL+SHIFT+L
New Project	CTRL+N
Open Project	CTRL+O
Page Setup...	CTRL+SHIFT+P
Paste Text	CTRL+V
Plan Editor Palette	CTRL+SHIFT+E
Print Project Report...	CTRL+P
Save Project	CTRL+S
Update JACK Files	CTRL+SHIFT+U

Table A-8: Keyboard shortcuts sorted by function

Note: There are no shortcuts located in the View menu, the Window menu or in the Help menu.

Appendix B: Command Line Options

Command line generation of JACK files

JACK files can be generated from the command line without having to start the JDE. The command line is as follows:

```
java aos.main.Jack [-J] [-UJ] [-D<dir>] [-P<package>] {<gcode-
files>|<project-file>
```

The command line options are described in the following table:

Option	Meaning
`-J`	Generate JACK files
`-UJ`	Generate out-of-date JACK files
`-CJ`	Remove generated JACK files
`-D`	Specify the directory to put JACK files in
`-P`	Specify a root package (not needed if generating from a project file)

Table B-1: Command line options

Index

C

D