Neuron Data Elements Environment
Intelligent Rules Element

Version 4.0

User's Guide

Contents

2. Application Implementation

3. Application Editing

4. Application Processing

5. Application Testing

6. Application Documentation

7. Application Data

8. Application Delivery

A. Main Menu Options

B. Popup Menu Options

C. Network Icons

D. Customizing the Environment

E. Text KB Syntax Description

. Index

User's Guide

Preface

Purpose of this Guide

This manual gives instructions for using the Intelligent Rules Element shell. The Intelligent Rules Element shell is a unique environment for developing knowledge-based applications. It has a complete set of tools that lets application developers design, test, and refine their applications. This manual addresses the operations of these application development tools.

Description

The graphical user interface of the Intelligent Rules Element shell is an invaluable aid for learning your way around the Rules Element inference engine and its rule and object features. The graphical user interface also provides a visual metaphor for the application design process. In essence the interface eliminates the need for writing code and helps you visualize the growing application with the aid of its tools.

Capabilities

The shell possesses a number of unique capabilities worth mentioning at this point.

- Interactive windowing utilities that let you arrange and view textual and graphical information about your application.
- An extensive context-sensitive, popup menu system that lets you perform actions from anywhere in the application.
- Menu-based, template editors that guide data entry and let you verify and compile the application incrementally.
- Facilities that let you inspect and modify the application at any time, either statically or dynamically (during runtime).
- Global editing capability that lets you easily delete unneeded items from the application.

Hardware Platforms Supported

The Rules Element shell runs fundamentally unchanged on most hardware platforms that provide graphics-display capabilities. Neuron Data Inc.

based the shell on a portable standard in order to preserve the "look and feel" described in this manual across all platforms. In turn this means the knowledge base files you generate on one hardware platform can run under the Rules Element "as is" on another machine. The hardware platforms currently supported by Neuron Data Inc. include:

- DEC VAX running under VMS or Ultrix
- Most Unix-based systems
- Apple Macintosh
- IBM PCs and PS/2's running under DOS or OS/2.

Note: The developer operates the Rules Element with the aid of a mouse. Due to the variety of mouse devices available, important guidelines for using the mouse appear in Chapter One, "The Windowing Environment."

Audience

This manual is for application developers who want to use the graphical editing and testing tools provided by the Rules Element shell to develop their application. Developers can also save files generated by the shell tools in a special text format (identified by the "tkb" filename extension) to inspect and modify knowledge base code with the text editor of their choice. However, modifying text files is not required when using the graphical user interface. The text file format is provided primarily as an option for those application developers accustomed to a code-intensive design technique. For a description of the syntax the system uses to save knowledge base files, refer to Appendix E, "Text KB Syntax."

Developers who want to embed Rules Element functionality directly into the code of another application should also refer to the Intelligent Rules Element Application Programming Interface (API) Programmer's Reference Manual. This alternative approach to applications design completely bypasses the graphical user interface and is therefore not addressed in the User's Guide.

How to Use this Guide

The organization of this manual is intended to ease new users into the application development process. The chapters of this manual group the Rules Element tools into six separate design tasks. A tool's function is therefore defined by the design task of the current chapter. New users will

find instructions for using the tools provided by the Rules Element under the following topics.

1. Implementation of the rule and object structures

2. Editing the existing rule and object structures

3. Processing the application using the inference engine

4. Monitoring and testing the application

5. Documenting the application

6. Integrating with databases

7. Delivering the application files

As you gain experience with the graphical user interface, the boundaries between these categories will become less important. In essence you will continue to design your application as outlined by these five steps, but you will also learn to focus the tools on smaller and smaller structures. This technique, known as "incremental design," lets you analyze the interaction of new structures with the established rule and object structures of the application. The Rules Element shell exploits this powerful technique by allowing you to inspect, modify, and add rules and objects during runtime (also known as "designing on the fly").

This manual is a member of the document set. See "Related Manuals" for a complete list of prerequisite and corequisite manuals.

Organization

To locate specific tasks, refer to the general table of contents, the chapter table of contents, or the index. This manual has seven chapters and five appendices:

Chapter One, "The Windowing Environment" gives background information about the operations you can perform in the graphical user interface. Specifically, it details the windowing features of the graphical user interface that let you interact with the shell.

Chapter Two, "Application Implementation" explains how to use the Rules Element editors to create rules, objects, methods, and related structures.

Chapter Three, "Application Editing" explains how to use the Rules Element features to modify your application files. This chapter also introduces features that let you inspect the rule and object structures you build.

Chapter Four, "Application Processing" instructs you on processing your application in the Rules Element. Inference engine processing performed by

the developer is primarily a debugging aid; the way the end user conducts a session depends on the final delivery method.

Chapter Five, "Application Testing" explains how to use the graphic network browsers and agenda monitor to test your application. The Rules Element permits access to these unique features even during inference engine processing, this chapter therefore gives techniques for debugging on the fly.

Chapter Six, "Application Documentation" explains how to use Rules Element features to document the rule and object structures of your application. You can use these features to aid in debugging, to provide help information for the end user, or just to keep a record of the application development effort.

Chapter Seven, "Application Data" explains some of the concepts behind how to use the Rules Element database bridge, and gives the basic steps to using the bridge.

Chapter Eight, "Application Delivery" gives instructions for encrypting completed applications and describes issues related to delivering knowledge base files to platforms other than the one used in development.

Appendix A, "Main Menu Options" gives brief explanations of the options you select from the main menus.

Appendix B, "Popup Menu Options" gives brief explanations of the options you select from the popup menus.

Appendix C, "Network Icons" gives an overview of the icon functions of the network windows for quick reference.

Appendix D, "Customizing the Environment" demonstrates procedures you can follow to customize the appearance of the graphical user interface, including the network windows and others.

Appendix E, "Text KB Syntax" describes the format in which knowledge base files are saved. Familiarity with the KB syntax is useful for directly editing saved files.

Documentation Conventions

Although the Rules Element's graphical user interface operates very much the same on all hardware platforms, you may notice small differences in the appearance of the Rules Element screen images. Screen images for this guide were all taken running either theMotif look typical of Unix platforms or the Windows95 look.

The following conventions are used throughout the procedures and text in this guide:

user action	Text printed in boldface indicates an important key or instruction that you must enter exactly as given.
FILENAMES	File names and directories are shown in uppercase when used within the body of a paragraph.
`screen text`	Text in this typeface indicates text that appears on the screen when something is displayed by the program.
`source code`	Monospaced text in this typeface indicates source code for the KB text format.

This guide refers to the person building the application with the Rules Element as the developer and the person operating the application generated by the Rules Element as the end user.

Related Manuals

The following manuals contain information related to this User's Guide. Read prerequisite manuals before using this guide. Read corequisite manuals for background information as explained.

Prerequisite Manuals:

Getting Started

This manual gives an overview of the development tools, including the Rules Element shell: its graphical user interface, the inference engine, and application structures. It emphasizes the development environment's ease of interaction through hands-on exercises you perform. Many of the terms expressed in the Intelligent Rules Element User's Guide are explained in the Getting Started manual.

Corequisite Manuals:

Language Reference

This manual is the application developer's reference guide to the entire Rules Element shell. It explains the operators of the graphical user interface and shows the correct syntax to use. The User's Guide gives general procedures for using the graphical user interface. Look up topics in the Reference Manual when you want to know more about the way to specify rule and object structures in the graphical user interface.

The Bibliography, located in the Introduction Manual, gives a complete list of your manuals.

Users who received the Intelligent Rules Element packaged with other Neuron Data Elements, including the Open Interface Element and the Data Access Element, will have other documents in addition to the Intelligent Rules Element documents described above.

1

The Windowing Environment

This chapter gives important background information about the operations you can perform in the graphical user interface. Read this chapter to understand the underlying concepts of the interface.

Overview

The Intelligent Rules Element graphical user interface is essentially a windowing environment for application development. The system automatically displays the windowing environment upon starting the Rules Element on your machine. It requires a graphics-based workstation and mouse to run. Figure 1-1 shows the graphical user interface during a typical application development session, although the interface on your particular hardware platform may appear slightly different.
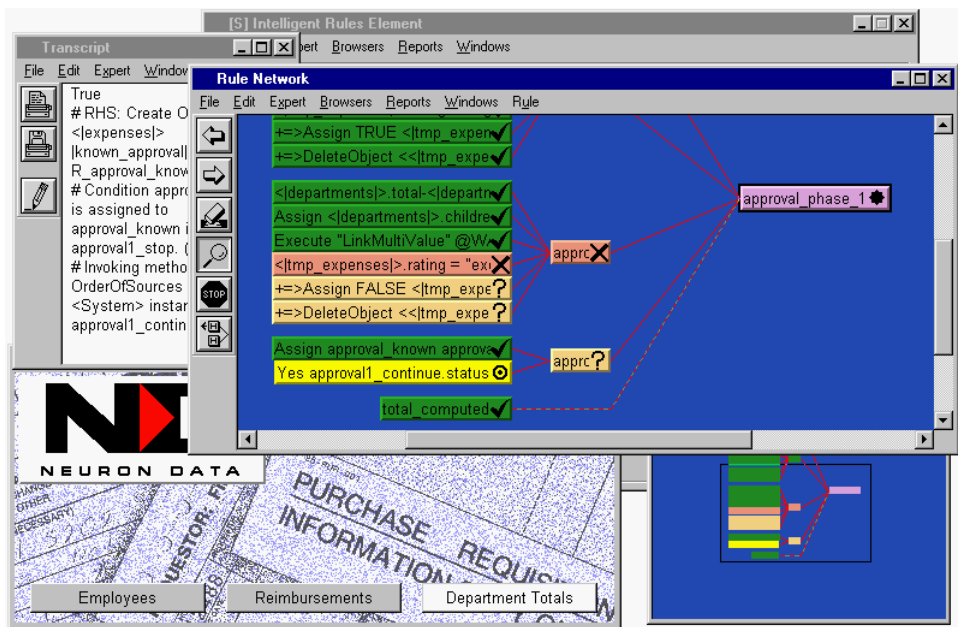
Figure 1-1   Rules Element Application Development Session

The graphics display requirement accounts for one of the interface's most prominent features – multiple windows. In the Rules Element, multiple windows let the developer view different aspects of the application at the same time. Windows also accept application design input and give the developer easy access to the Rules Element tools by compartmentalizing tool functions.

> **Note:** The following sections describe the Intelligent Rules Element® of the Elements Environment. For details about the Resource Browser see the Open Interface Element User's Guide.

The Mouse and Menus

You perform all operations with the aid of a mouse device connected to your computer. The mouse controls the position of a graphic cursor that appears on the screen. This moveable arrow cursor is called a "pointer". Once the pointer is overlaid on a desired screen element, pressing and releasing the mouse button initiates the action (the left button for multi-button mouse users). This two-step selection operation (overlaying the mouse cursor and clicking the mouse button) is commonly called "point and click."

One of the most common point and click operations involves selecting an option from a menu. In general menus consist of a list of options that let you focus the Rules Element windowing environment. You select an option on a menu by moving your mouse pointer over the list until the desired option appears highlighted on the screen. You cause the system to initiate the highlighted option by clicking with your mouse button.

There are several types of mice available that you can use. The Rules Element works with one, two, or three button mice. The way to select is different for each mouse type. Table 1-1 describes these functions for the three mouse types shown in Figure 1-2.

Function	Button	Purpose
Select	Left Button, or: On Macintosh press the single button.	Selects and manipulates window elements. Selection sometimes requires a double-click of the mouse button to initiate the operation.
Windows Popup	Middle Button, or: On Mac press Command (⌘) key and button. On PCs with a two button mouse, press Ctrl and right button.	Shows the list of open windows. Selecting a window name in the list displays that window in the foreground and makes it the active window. On Unix machines with a two button mouse, press both buttons.
Local Popup	Right Button, or: On Macintosh press option key and button.	Displays a context-sensitive menu that shows options related to the selected item only. Mouse pointer must overlap a particular window element.

Table 1-1 **Mouse Button Functions**

Popup menus form an extensive system in the Rules Element that lets you take shortcuts and keep the windowing environment focused. Popup menus bring choices on the screen which depend upon the current status of the windowing environment or a particular item displayed in a window. For more information about using popup menus, see "Selection Operations" in this chapter. For brief descriptions of popup menu options, refer to Appendix B, "Popup Menu Options."

Figure 1-2 shows the mouse button functions for righthanded mice (reversed for lefthanded mice).
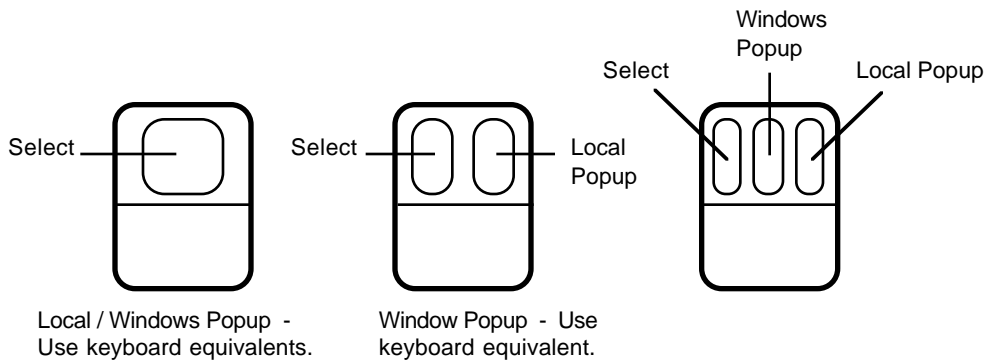
Figure 1-2 **Mouse Buttons**

Starting the Rules Element

The procedure to start the Rules Element depends on the type of operating system your computer runs under. Some systems require that you supply the pathname of the application. Others require you to type the command to launch the application from specific directories. After you successfully start the Rules Element, the system displays the Main Window.

Here are a few examples of starting the Rules Element on different machines.

Macintosh

1. Position your mouse pointer over the Rules Element application icon.

2. Double-click the selection mouse button. The system launches the Rules Element.

3. Select the Acknowledge button to view the Main Window.

Workstations

1. Set the $PATH environment variable to include the \smartelt\bin directory.

2. Set the ND_DATA environment variable to include the \smartelt\lib directory.

   The Installation Guide for the Rules Element describes the environment variables that you can use to configure your application.

3. Type: smartelt
   The system launches the Rules Element.

4. Select the Acknowledge button to view the Main Window.

IBM PC running DOS Windows or Windows NT

1. Set the PATH environment variable in your autoexec.bat file to include the \smartelt\bin directory.

2. Set the ND_DATA environment variable in your autoexec.bat file to include the \smartelt\lib directory and reboot the system.

   The Installation Guide for the Rules Element describes the environment variables that you can use to configure your application.

3. Type: win smartelt
   The system launches the Rules Element .

4. Select the Acknowledge button to view the Main Window.

IBM PC running OS/2 Presentation Manager

1. Set the LIBPATH environment variable in your config.sys file to include the \smartelt\bin directory.

2. Set the ND_DATA environment variable in your config.sys file to include the \smartelt\lib directory and reboot the system.

    The Installation Guide for the Rules Element describes the environment variables that you can use to configure your application.

3. Display the Full Screen window and type: smartelt
   The system launches the Rules Element.

4. Select the Acknowledge button to view the Main Window.

Types of Windows

There are three general categories of windows in the Rules Element windowing environment as shown in Figure 1-1:

■ Main Window
■ Description Windows
■ Dialog Windows.

The windowing environment lets you arrange these windows on your display in a highly flexible manner. For instance, a few basic mouse operations let you display, hide, resize, move, and overlap windows to suit your own needs. You choose which windows to display by selecting options from the menu windows as described in the following sections.

Main Window

The Main Window displays important options for interacting with the Rules Element. The Main Window appears when you successfully launch the Rules Element. It displays a list of the knowledge bases that you load into memory, a session control panel to monitor events, a customizeable control panel that lets you add button equivalents for commonly used menu items, an inference engine status field, and graphical user interface (GUI) engine

status field. Figure 1-3 shows the Main Window and its associated popup menus that you can display for the areas as indicated.
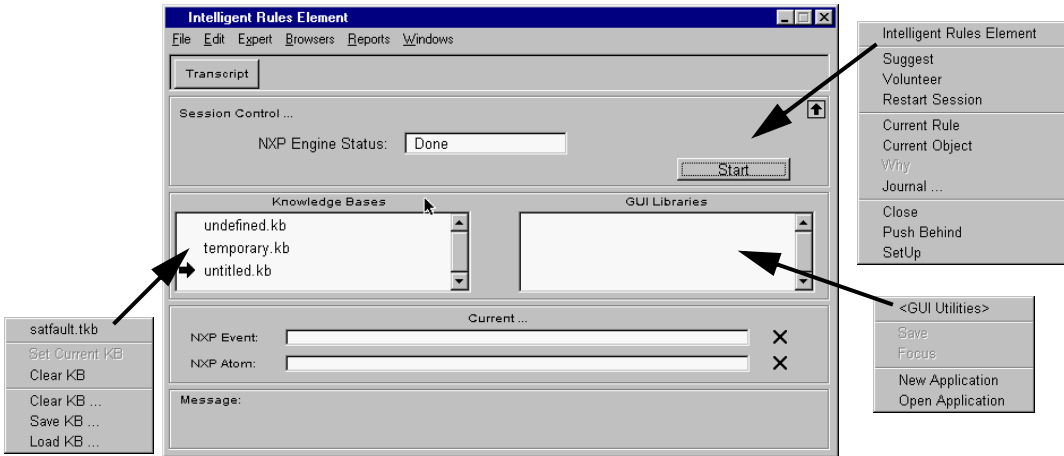
Figure 1-3   Main Window and Associated Popup Menus

Note: The menu bar that appears with the Main Window is accessible from any window. For a complete description of the main menu options, refer to Appendix A, "Main Menu Options."

During a typical application development session which includes loading, editing, and testing your knowledge base files, you will perform many tasks using the Main Window. The following sections give an overview of its important features. Later chapters of this manual describe these features in more detail:

Customizing the Main Window

The top portion of the Main Window lets you create a button equivalent of any command that appears in the menu bar menus. Figure 1-3 shows the scrollable area of the Main Window with a button equivalent of the Transcript command. Adding button equivalents of the menu commands to the Main Window lets you easily reuse these commands during a session (for example, Restart and Start With... Knowledge Base).

To add buttons to the scrollable area of the Main Window, hold the SHIFT key down, select the desired command from the menu bar menu, and click in the scrollable panel. A button with the command name appears in the

panel. You can further customize the buttons you add to the scrollable panel by displaying a local popup menu on the button (press the selection button on your mouse). The following button options are available:

Label   Change the default button label (same as the menu item name).

Icon   Add or change the icon displayed on the button, if any.

Move   Move the button within the scrollable panel.

Delete   Remove the button from the scrollable panel.

Note that command buttons that you add to the scrollable panel are saved with the Rules Element settings.

Session Control Panel

The area just below the button command equivalents displays status information about the inferencing process when you test your application. This area of the Main Window is called the session control panel. During application processing the panel displays information about the application depending on the inferencing state:

■   When the system is not asking a question, it displays the state of the inference engine and a Start/Interrupt button.

■   When the system is asking a question, the panel shows the default question listing the different value options.

Note: During application processing you can reduce the size of the Main Window to display only the session control panel by clicking on the up arrow displayed on the right side of the panel.

A local popup menu that you display for the session control panel gives you commands to interact with the facilities of the Rules Element shell during application processing (Suggest, Volunteer, and Restart Session). It also gives you the standard actions such as Close and Push Behind. Descriptions of the processing commands and the facilities they invoke appear in Chapter Five, "Application Testing."

Application Processing

The button labeled "Start" in the Main Window session control panel is used to control application processing. It lets you interrupt/continue application processing. After you click on the button, the label changes to reflect the current state; it can have one of the following values:

Start   Equivalent to selecting the Start With... Knowledge Base command.

Interrupt   Equivalent to stop session.

Continue   Resume session.

The display text box labeled "NXP Engine Status" gives you information about the status of the Rules Element inference engine. During application processing the system will display one of the following values:

STOPPED	Pause for a question or a breakpoint that you insert in the agenda or networks.
RUNNING	Processing the knowledge base.
DONE	Knowledge processing complete. Not running.

Descriptions of the Rules Element facilities that you use to produce these states appear in Chapter Four, "Application Processing" and Chapter Five, "Application Testing."

Loading Your Knowledge Base Files

The knowledge base (KB) files that you load into the Rules Element memory appear in the list box on the left of the Main Window (see Figure 1-3). When you first start the Rules Element the system always displays the default internal KBs: `undefined.kb`, `untitled.kb`, and `temporary.kb`.

A local popup menu attached to the list box, and also to each KB file name, duplicates several commands that appear in the Expert menu of the menu bar. The local popup menu may be more convenient to use than the Expert menu when you want to clear one or all KBs, set the current KB, or load a KB. These operations are described in Chapter Three "Application Editing."

Loading GUI Libraries

When you use the Resource Browser (available from the Browsers menu bar menu) to load a graphical user interface (GUI) application or the Open Application command from the Expert menu of the Rules Element main menu bar, the list box on the left of the Main Window displays the loaded GUI libraries which contain application windows. A local popup menu provides options to load and unload the libraries displayed in the Resource Browser.

Note: See the Open Interface Element User's Guide to build and use graphical user interfaces with your application. Also see that manual for details about the Resource Browser that you display from the Browsers menu bar menu.

Debugging Facilities

During application processing the current hypothesis and current slot can be displayed by enabling the NXP Event and NXP Atom fields respectively.

Description Windows

Description windows display the tools of the Rules Element shell you use to inspect, create, modify, and delete application structures. Windows in this category contain a title bar that identifies their particular function.

Editors

To guide the application design process, the Rules Element provides seven highly-formatted editor windows that act as templates. The seven templates contain fields that you enter data into. The field associated with your data determines how the data functions: as a value, an expression, or the name of a particular application structure. You select the editor windows from the Edit menu on the main menu bar or from the popup menus of other windows. Figure 1-4 shows the common elements shared by all editor windows.
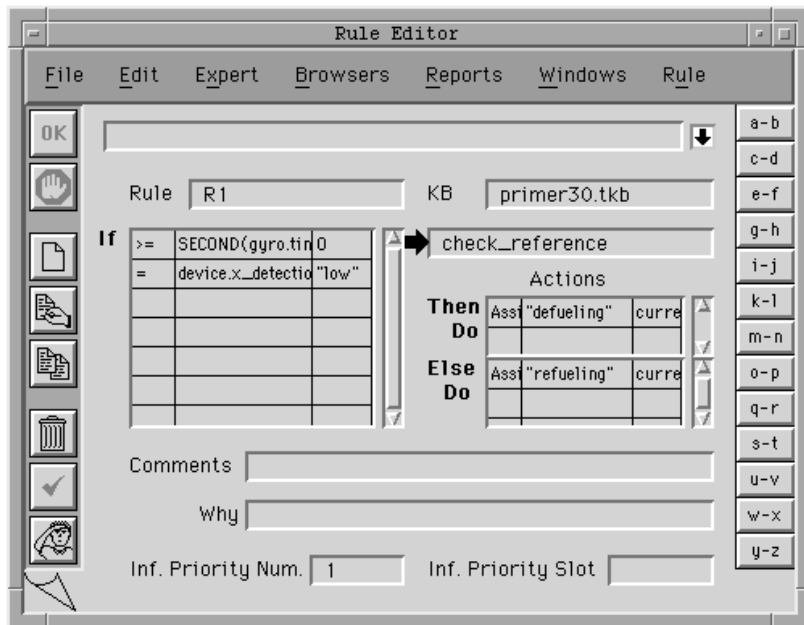
Figure 1-4   Sample Editor Window

List Windows

List windows in the Rules Element give a textual display of all the information contained in your application. These windows group application structures into related categories. Rules, methods, hypotheses,

data, classes, objects, and properties each have their own window. You select these windows from the Browsers menu in the menu bar or from the popup menus of other windows. Figure 1-5 shows the common elements shared by all list windows.
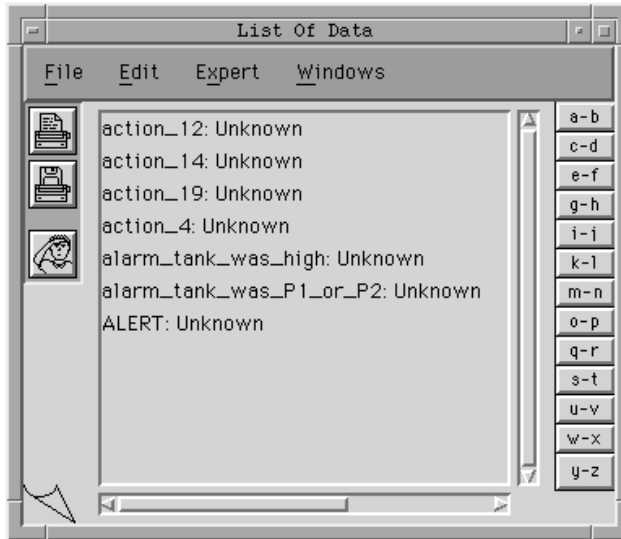
Figure 1-5 Sample List of Data Window

Network Windows

The graphical networks are a third type of description window that you can display in the Rules Element. These windows let you view and browse the network formed by your application's existing structures. There is a separate network window for the rules and objects that you create. The rule network diagram represents the reasoning pathways. In the case of objects, the network represents the inheritance pathways used by the Rules Element to obtain values. You select these windows from the Browsers menu in the

menu bar or from the popup menus of other windows. Figure 1-6 shows the Rule Network with sample rules displayed.
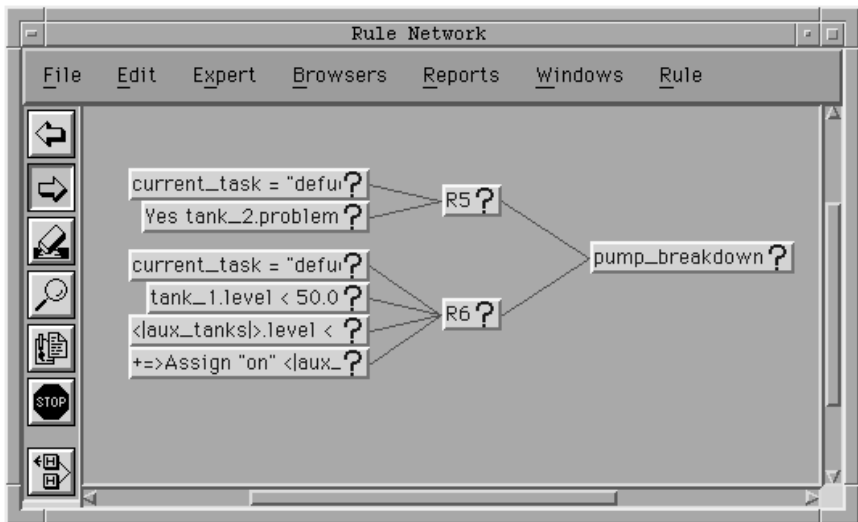
Figure 1-6   Sample Network Window

Dialog Window

Dialog windows let you supply system information to the Rules Element during the application development session. The system opens dialog windows automatically in response to some action you performed. An example of a dialog window is one that asks for confirmation before allowing you to delete something from the application. A sample dialog window appears in Figure 1-7 below.
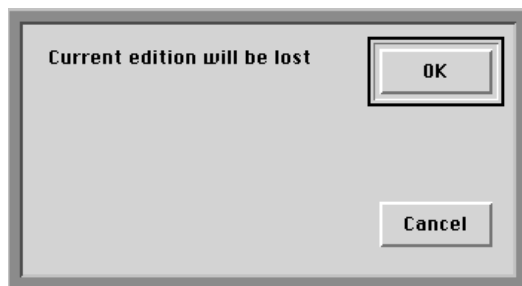
Figure 1-7 Sample Dialog Window

This category of window is special in that it requires a specific reply from you before the system can proceed. For example, the dialog window may require you to select an OK button (telling the system that whatever prompted the window to display is permitted) or a Cancel button (telling the system to exit without modification) before it lets you initiate any other action. Once the system opens a dialog window you must provide the requested information to continue your session. The complete list of dialog window replies follows; the dialog window message determines which reply to use.

- Clicking on buttons (single words enclosed in a box)
- Double-clicking on one or more items from a list
- Clicking on an edit line, typing a response, and pressing the Return key.

Global Operations

Use the following operations to control the appearance of the windowing environment. In the browsing mode, the system gives unlimited access to all windows. However, once you begin interacting with a window the other windows become inactive. The system therefore maintains one and only one active window at a time. By rearranging the windows on your screen you can optimize your interaction with the application development tools currently displayed. If you forget which window is active, look at the title bar of the displayed windows. Only the active window has the darkened title bar, all other window title bars are plain.

Open and Close

You open and close windows in the graphical user interface to change your focus of attention on the application development tools. You can open as many windows as the screen can practically display. The windows let you browse or modify the contents of your application. When you open a new window the system automatically makes it the active window in the environment.

To open a window:

1. Position the mouse pointer over the desired main menu option and click. The system displays the options list.

   The Control-key and letter combination also give keyboard shortcuts to display windows. The menus identify the corresponding letter next to the option when available.

2. Position the mouse pointer on top of the desired option and click. The system opens the window.

    A new window that you display may overlap an existing window thus hiding a portion of the other window from view.

To change the status of an inactive window that is visible:

1. Position the mouse pointer anywhere in the area of the inactive window. The area must be visible on the screen.

2. Click on the area. The system makes the selected window active and darkens the window's title bar.

To change the status of an inactive window that is hidden from view:

1. Position the mouse pointer anywhere in the area of the active window and press the middle mouse button. The system displays the "Windows" popup menu.

    One-button mouse users must hold down the Command key on their keyboard while clicking the mouse button.

2. Position the mouse pointer on top of the desired window name and click the mouse button. The system makes the selected window active and the popup menu disappears from the screen.

    The "Windows" popup menu also duplicates application processing commands from the Expert menu that you can select. It also contains the same list as the Windows menu on the main menu bar.

When you no longer need a particular window you can remove it from the windowing environment. This action is called closing a window.

To close a window:

1. It is necessary to make the window active as described above.

2. Reposition the mouse pointer on the active window and click. The window's popup menu appears.

3. Position the mouse pointer over the Close option and click. The window disappears from the windowing environment.

    You can also close a window from its title bar in the top-left corner.

Move

Opening a new window in the windowing environment may result in overlapping windows. This is a normal situation that helps to conserve screen space when working with multiple windows. You can, at any time, move a window so it is more fully displayed.

To move a window:

1. Position the mouse pointer over the desired window's title bar.

2. Press the mouse button and drag the cursor to the window's new location, the window also moves. Release the mouse button once the window reaches the desired location.

Resize

Another way you can conserve screen space is to reduce the size of a window in the windowing environment. Reducing window size will, however, affect the visibility of the window's contents. You can, at any time, reduce or enlarge a window as desired. Enlarging windows is sometimes helpful to more fully display data that appears "cropped" inside a window. You can enlarge windows up to the full size of the screen display.

To resize a window on all hardware platforms:

1. It is necessary to make the window active as described above.

2. Resizing takes place by stretching the window from one of its four corners. Decide which direction you want to stretch or reduce the window from.

3. Move the mouse pointer toward the corner you want to resize until the cursor changes into one with double arrowheads. The area that this cursor exists in is quite small; take care not to move the cursor out of the corner area.

4. Press the mouse button and drag the resize cursor toward the center to reduce the window or away from the center to enlarge the window. Release the mouse button when the window reaches the desired size.

Push Behind

Another operation you can use to view windows is called "push behind." This operation is particularly useful when you want to display hidden windows without moving, resizing, or closing the window in the foreground. The window you push behind merely changes positions with another window in the background.

To push a window behind all others:

1. Position the mouse pointer over an area of the active window that contains no field or button and click the right mouse button. The system displays the global popup menu.

   One-button mouse users must take care that the mouse cursor looks like ▤ and not ➤ when clicking on non-specific areas.

2. Position the mouse pointer on top of the Push Behind option and click the mouse button. The system places the window in the background, thus bringing the next window in full view as the active window.

   The Windows menu can also be used to go directly to any window that is currently displayed.

Selection Operations

The graphical user interface provides a few standard operations you use to interact with the Rules Element during an application development session. You interact with the Rules Element with the aid of a mouse device (to display the Rule editor or view the network window for example.) The mouse lets you select semi-graphical controls commonly known as buttons. Buttons that you select from an active window will either trigger immediate actions in the windowing environment or simply store the particular state of a window.

Function Buttons

Function buttons appear in windows as boxes that enclose a command name. The system initiates the function identified by the command name when you use your mouse to point and click on the button. This type of button always initiates an immediate action that the windowing environment reflects.

Note: Some window areas that resemble buttons are actually label boxes for a field. You can distinguish function buttons from label boxes by positioning the mouse cursor over the box and observing the shape of the cursor. If the mouse cursor has the pointer shape, the box is a function button.

Buttons that you select can appear individually or in a group.

State Buttons

System buttons do not initiate an immediate action when you use your mouse to point and click. The selection operation merely gives these buttons a selected or unselected status. It is the selection state itself that has some meaning to the Rules Element. The system saves the state selection so you do not have to make the same selections for each new session.

> **Note:** You can select these buttons to change states at any time during the application development session, even during runtime.

State buttons can resemble small checkboxes that let you choose among alternative choices. Darkened checkboxes or checkboxes with an "X" have a selected status while empty boxes have an unselected status:

Inheritance Strategy

◇ **Class First**

◇ **Object First**
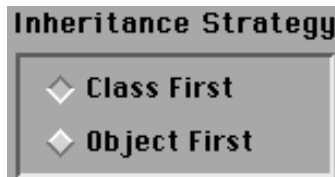
Other state buttons have a shape closely related to or indicative of the function they represent. An example of these graphical checks is the inheritance arrows that appear in the Strategy window. Darkened (or highlighted) graphical checks have a selected status while gray (or plain) graphical checks have an unselected status:

Inheritability

⇧

Slot | **Default**

⇩

⇧

Value | **Default**

⬇
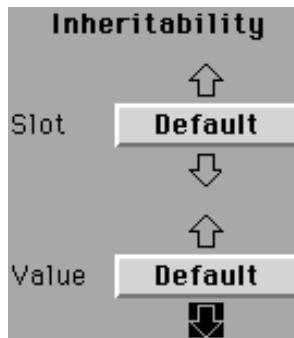
Popup Menus

Popup menus are another prominent feature of the Rules Element windowing environment. These temporary menus give the developer additional flexibility for initiating actions. Popup menus act similar to the

function buttons described above and often duplicate the commands for a particular window. Once displayed, you must hold the mouse button down to keep the menus displayed in the Rules Element.

Note: Refer to Appendix B, "Popup Menu Options" for a complete description of options available.

There are three types of popup menus that let you make selections for the active window as described in the following sections.

Local Popup Menus

This popup menu shows options related to the selected field or item only (a list of valid operators for example). The mouse cursor must be on top of a particular field item in a window as described in the following selection procedure.

To select local popup menu options:

1. Position the mouse pointer over the area of a particular entry field of the window and press down the right mouse button. The system displays the local popup menu.

    Macintosh users must press the option key and the mouse button and must take care that the mouse cursor looks like ▤ and not ✛

    when clicking on the desired field. Move the cursor slightly to the right if necessary.

2. Without releasing the mouse button, position the mouse pointer on top of the desired option.

3. Release the mouse button. The system initiates the action and the popup menu disappears from the screen.

Global Popup Menus

This popup menu shows options related to the entire window and usually duplicates the window's function buttons. The mouse cursor can be anywhere that is not a field or button in a window as described in the following selection procedure. In some windows the only place to display the global popup menu is in the title bar.

To select global popup menu options:

1. Position the mouse pointer over an area of the window that contains no field or button and press down the right mouse button. The system displays the global popup menu.

    The button is the same one used to display the local popup menu, but

the menu displayed is not for a specific item in the window.

2. Without releasing the mouse button, position the mouse pointer on top of the desired option.

3. Release the mouse button. The system initiates the action and the popup menu disappears from the screen.

Windows Popup Menu

This popup menu provides options for initiating application processing from the active window. It duplicates options of the Expert menu, but makes it convenient to work entirely from the active window. This popup menu also shows the list of currently opened windows. Selecting a window name in the list displays that window in the foreground and makes it the active window.

To select Windows popup menu options:

1. Position the mouse pointer anywhere in the area of the active window and press down the middle mouse button. The system displays the Windows popup menu.

   One- and two-button mouse users must hold down the Command (or Ctrl) key on their keyboard while clicking the left mouse button. Unix users press both mouse buttons.

2. Without releasing the mouse button, position the mouse pointer on top of the desired option.

3. Release the mouse button. The system makes the selected window active and the popup menu disappears from the screen.

   The "Windows" popup menu also duplicates application processing commands from the Expert menu that you can select.

Browsing Operations

Many windows give you access to more information then can be displayed at once. To access the "hidden" information you use the window's built-in browsing mechanisms. The types of browsing mechanisms available depend on the window displayed. For example, some windows permit scrolling of the contents while others permit direct access to alphabetized lists. Each is explained in the following sections.

Note: Browsing operations are only possible on an already active window. It may be necessary to click on the desired window before using the browsing mechanisms.

Scrollbar

Editor and network windows have scrollbar mechanisms that let you navigate the contents of the window. They consist of a scroll box (or "thumb") capped by an arrow on either end of the box. This mechanism lets you navigate the contents of the window when the window contains more information than can actually be seen. The scrollbar located along the right side of the window scrolls the contents vertically. The scrollbar located along the bottom of the window scrolls the contents horizontally. Figure 1-**8** shows the vertical scrollbar of the Transcript window.
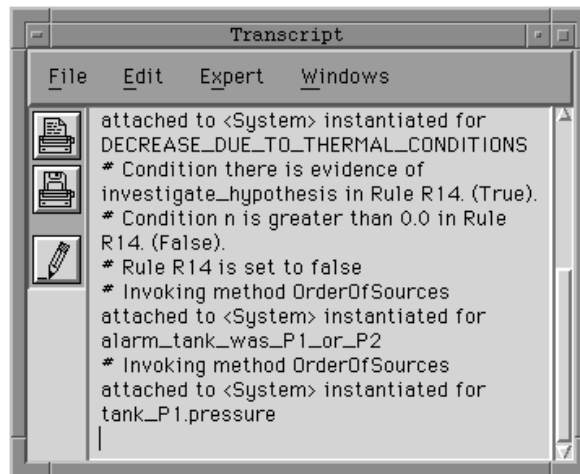
Figure 1-8   Scrollbar

You can tell at a glance whether a window has scrollable information by the size of its scroll box. The amount of area between the two arrows that the scroll box takes up reflects the proportion of the contents displayed. For example, a window with half of its contents displayed will have a scroll box that is about half of the size of the scrollable area (see Figure 1-8).

To scroll line by line:

1. Position the mouse pointer on top of the arrow located at either end of the scrollbar and click. The window moves in the direction pointed to by the arrow thus revealing a new line.

2. Click on the area again to scroll the window by another line.

To scroll rapidly:

1. Position the mouse pointer on top of the scroll box.

2. Press the mouse button and drag the scroll box in the desired direction. Release the mouse button when the box approximates the desired location of the window.

    During this procedure the scroll box remains in the same place while you drag its outline along the scrollbar.

3. Alternately, position the mouse pointer inside the gray area of the scrollbar (outside of the scroll box itself) and click. This will scroll the contents by the size of the current display.

Page Flip Graphic

Editor and notebook windows have a page flipping graphic to facilitate browsing. This particular graphic is located at the bottom left corner of the window. It consists of two triangles that resemble a turned up page corner. The page flipping graphic lets you browse independent of the window's contents. However, it is available only on those windows that display contents in a page by page fashion. Figure 1-9 shows the page flipping mechanism for the Rule editor.
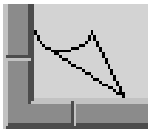
Figure 1-9   Page Flipping Graphic

To browse by page flipping:

1. Position the mouse pointer on the upper triangle and click. The window displays the next item in alphabetical order, similar to turning to the next page.

2. Position the mouse pointer on the lower triangle and click. The window displays the previous item in alphabetical order, similar to turning back a page.

Lateral Index

Editor and notebook windows include a lateral index that gives you more direct access to their contents than the page flipping mechanism. In order to have a lateral index the window must organize information in alphabetical

order. These windows display only the information that corresponds with
the currently selected letters of the alphabet. Figure 1-10 shows the lateral
index for the List of Rules window.
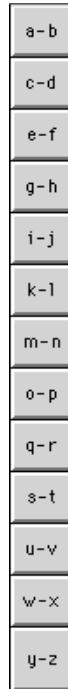
Figure 1-10   Lateral Index

To access information alphabetically:

1. Position the mouse pointer on the appropriate index box and click. The
   window displays the first item starting with the first two letters in the
   selected index box.

   Clicking on an index box that has no listing automatically displays the
   contents that come closest in alphabetical order to the letters selected.

2. Alternately, type the first letter of the desired item's name on your
   keyboard. The window automatically brings the list window to the
   desired item.

Find Button

Editor windows also have a Find iconic button to facilitate browsing. This
particular mechanism is located at the bottom right corner of the window. It

consists of a box with a question mark. The Find button lets you browse by locating the name of the desired application structure. The system automatically displays the structure whose name matches a string you type into a special dialog window. For example, in the case of the Rule editor the system tries to find the rule name of the rule which contains the string you supply. If no match is found, the match may be on the hypothesis name. In this case the choice can be made by selecting the desired Sort By option. Figure 1-11 shows the Find button for the Object editor.

 ——*Find Button*
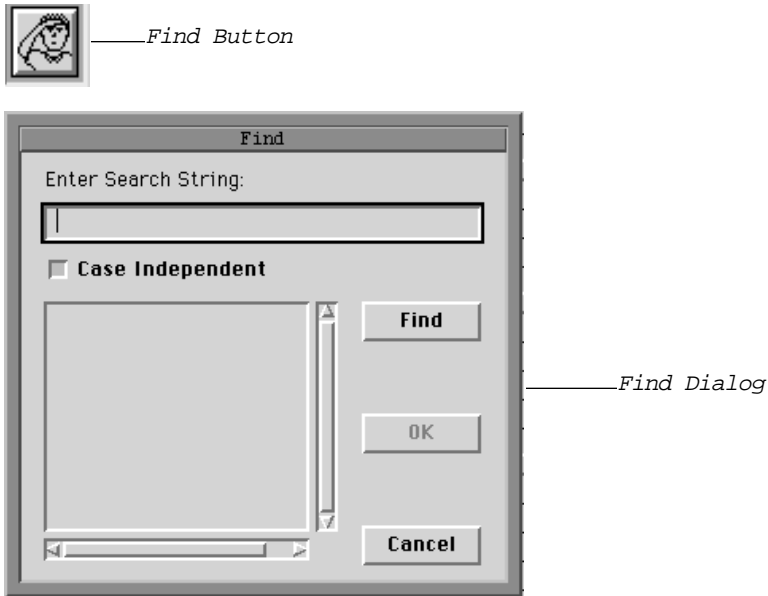
 ———*Find Dialog*

Figure 1-11   Find Button and Dialog Window

To browse by locating a string:

1. Select the desired Sort By option from the editor if available.

    In the case of the Rule editor the option lets you base the search on rule names or hypothesis names.

2. Position the mouse pointer on the Find button of the editor window and click. The system opens a dialog window with a text edit line.

3. Type the string you want to locate in the text edit line and click on the **Find** button of the dialog window. The system displays every entry in the application that contains the string.

4. Position the mouse pointer on the desired entry and click. The system highlights the selection.

5. Click on the **OK** button of the dialog window. The system closes the dialog window and displays the application structure that contains the selected string in the editor.

Network Overviews

The network windows provide their own special browsing mechanism called an "overview." You use the overview to rapidly browse a network diagram that does not display in the network window all at once. The overview itself provides a much reduced, one-page view of the entire network. The area of the overview enclosed by a dotted line identifies the portion of the network that currently appears in the network window. You can zoom to another portion of the network by moving the box around the overview as described in the following procedure. Figure 1-12 shows the overview mechanism for the Rule Network window.
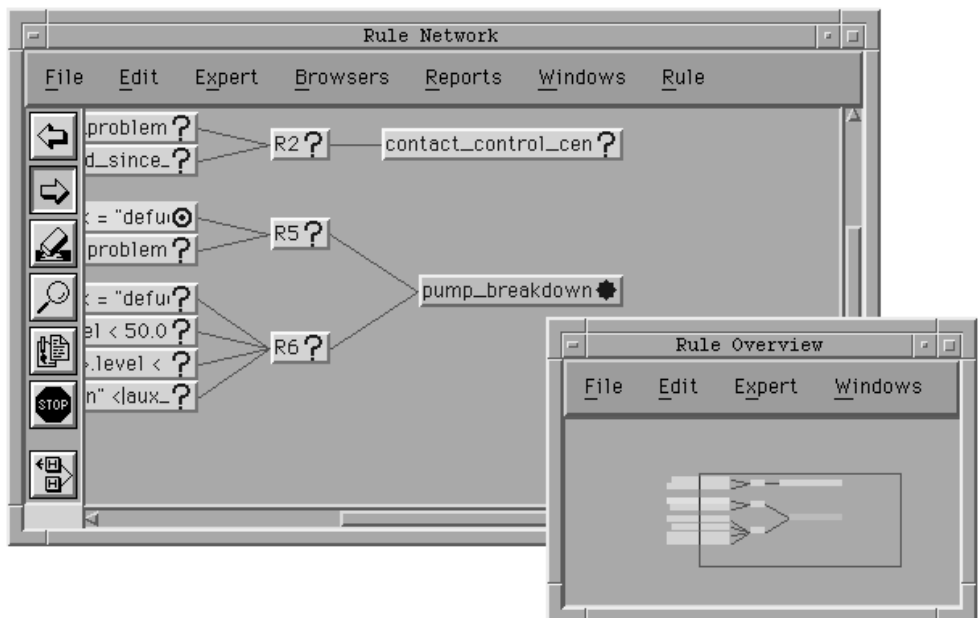
Figure 1-12   Network Overview Window

Note: The network windows have parameters that let you control the length of information displayed. Refer to Appendix D, "Customizing the Environment" for details.

To zoom the network window using the Overview:

1. Select the desired network option from the Browsers menu. Position the mouse pointer on top of the window-specific menu in the main menu bar and press the mouse button. The system displays the list of options.

   The Overview option is also available through the popup menu associated with the network window.

2. Move the mouse pointer over the list until the Overview option is highlighted. Release the mouse button to display the Overview window.

3. Position the mouse pointer on top of the area enclosed by the box in the Overview window and press the mouse button.

4. While holding the mouse button down, drag the mouse pointer toward the area of the network you want to display. Release the mouse button when the dotted line encloses the desired portion of the network.

5. Click on the network window. The window displays the new portion of the network.

   The Overview window remains conveniently displayed to let you zoom again. Close the Overview window when no longer needed.

As an alternative to zooming the network window through its overview, you can scroll the contents of the network window itself.

To zoom the network window using the network diagram:

1. Position the mouse pointer in an area of the network diagram that does not overlap with a specific item and press down the mouse button.

2. While holding the mouse button down, drag the mouse pointer. Release the mouse button when the desired portion of the network diagram appears in the network window.

Data Editing Operations

Entering data into the various Rules Element editor windows is an essential operation for creating application structures. To accomplish this task the editor windows give you a set of mode buttons that you choose to make the window accept your data. These functions are also duplicated in the

window-specific menu that appears in the main menu bar. Table 1-2 shows the complete set of iconic buttons available with each editor window.

Iconic Button	Text Equivalent	Description
OK	ACCEPT	Accepts the currently displayed structure into the application and readies the editor window for further additions. Verifies the syntax of entries of the current application structure.
(hand icon)	CANCEL	Returns the currently displayed application structure to its original unmodified state.
(page icon)	NEW	Clears the editor window and highlights the first field to accept your entry for a new application structure.
(edit icon)	EDIT	Places the currently displayed application structure in edit mode and lets you make changes where needed.
(copy icon)	COPY	Makes a duplicate of the currently displayed application structure and lets you make changes where needed.
(trash icon)	DELETE	Deletes the currently displayed structure from the application. This operation is not reversible.
(check icon)	CHECK	Verifies the syntax of the current application structure.
(find icon)	FIND	Searches the application to locate the named structure.

Table 1-2   Editor Window Iconic Buttons

Note: The system permits you to edit one application structure at a time. Once you begin a data editing operation, you must complete the operation before opening another editor window.

The data you type after selecting one of the buttons shown in Table 1-2 does not occupy a field in the editor template immediately, but is instead held temporarily in an area of the editor window called the "text edit line." This reserved area lets you manipulate the data directly with the aid of a mouse device. Once you are satisfied with the data, you can press the Return key

to enter the data into the selected field of the editor window. The following procedures show how to modify the contents of the editor windows with the text edit line.

Edit

In order to edit entries that currently appear in the template fields of the editor window, the desired data must appear in the text edit line. Use the following procedure to manipulate editor window entries.

To edit data in the editor windows:

1. Open the editor window that controls the type of application structure you want to edit. Display the desired application structure.

2. Position the mouse pointer on the **Modify** button and click the selection mouse button.

    You can also select the **Modify** option from the window-specific menu.

3. Position the mouse pointer on the template field that you want to edit and click the selection mouse button. The system duplicates the field entry in the text edit line as highlighted text.

    One-button mouse users must take care that the mouse cursor looks like ⬥ and not ▤ when clicking on the desired field. Move the cursor slightly to the left if necessary.

4. Move the mouse pointer to the highlighted text in the text edit line. The mouse pointer changes into the ⌶ cursor.

5. Position the mouse pointer between two characters that you want to edit and click the selection mouse button. The system removes the text edit line highlighting and a flashing vertical bar marks the spot in the text where editing begins.

6. Type the desired text or use the backspace key to delete text.

    Press the Delete key to clear the text edit line. You can enter new text as desired.

7. Once you are satisfied with the data that appears in the text edit line, press the Return key. The system replaces the previously selected field entry with the text edit line data.

    Clicking back on any template field completes the change and displays the new field.

8. Repeat the procedure to modify data in the currently displayed application structure. When done, click on the **OK** button to verify and compile the new data.

Insert

The editor windows let you enter data into the template fields through an area called the "text edit line." All data you type for entry into the editor window is first displayed in this reserved area. Use the following procedure to enter new data in a blank editor window.

To enter data into the editor windows:

1. Open the editor window that controls the type of application structure you want to enter.

2. Position the mouse pointer on the **New** button and click the selection mouse button.

    Can also select the **New** option from the window-specific menu.

3. Position the mouse pointer on the blank template field that you want to fill and click the selection mouse button. A flashing vertical bar marks the spot where you can begin typing in the text edit line.

    One-button mouse users must take care that the mouse cursor looks like ⬥ and not ▤ when clicking on the desired field. Move the cursor slightly to the left if necessary.

4. Type the desired data and use the backspace key to delete data.

    Press the Delete key to clear the text edit line. You can enter new data as desired.

5. Once you are satisfied with the data that appears in the text edit line, press the Return key. The system enters the data into the previously selected editor window field.

    Clicking back on any template field completes the change and displays the new field.

6. Repeat the procedure to enter data in the currently displayed application structure. When done, click on the **OK** button to verify and compile the new data.

Copy

The Rules Element has several powerful copying techniques that assist in application structure creation from the editor windows. Although these techniques apply to every editor window, they are most useful with the Rule editor's numerous fields. For example, you can perform the following copying tasks.

■ Copy entire structures to modify
■ Copy existing data from the various list windows
■ Copy a few lines to modify.

Procedures for each of these copying techniques follow.

Copy using Editor Button

To duplicate the entire structure displayed in the editor windows you can select the Copy button from the window's list of buttons. This operation often helps save you time when creating new structures. If the new structure resembles an existing one, you can modify the duplicate rather than create an entirely new one.

Note: Most editor windows, with the exception of the Rule editor, will not allow exact duplicates of structures to exist. Therefore when you use the Copy button, the system automatically places the editor window in edit mode. You are required to modify some aspect of the display before selecting the OK button.

Use the following procedure to copy existing structures in the editor windows.

To copy and modify an entire structure:

1. Open the editor window that controls the type of structure you want to copy. Display the desired structure.

2. Position the mouse pointer on the **Copy** button and click the selection mouse button. The system duplicates the structure and places the editor window in edit mode.

   You can also select the **Copy** option from the window-specific menu.

3. Position the mouse pointer on the template field that you want to edit and click the selection mouse button. The system duplicates the field entry in the text edit line as highlighted text.

   One-button mouse users must take care that the mouse cursor looks like ⬦ and not ▤ when clicking on the desired field. Move the

cursor slightly to the left if necessary.

4. Move the mouse pointer to the highlighted text in the text edit line and edit the field as desired. Press the Return key to return the modified data to the highlighted template field.

   Clicking back on any template field completes the change and displays the new field.

Copy using Popup Menu Option

Many data entry fields of the editor windows provide a local popup menu that gives you access to the existing data of the system. This includes hypotheses, methods, classes, objects, and data. Use the following procedure to copy existing data from the application into the current editor window.

To copy individual data from the application:

1. Open the editor window that controls the type of structure you want to edit. Display the desired structure.

2. Position the mouse pointer on the **Modify** button and click the selection mouse button.

3. Position the mouse pointer on the template field that you want to copy data into and click the local popup mouse button. The system displays the local popup menu for that field.

   Note: One-button mouse users must take care that the mouse cursor looks like 🔷 and not 🗒 when clicking on the desired field. Move the cursor slightly to the right if necessary.

4. Position the mouse pointer on top of the desired Select option and click the selection mouse button. The system opens a selection dialog window that lists the data available.

   The Select option is also available from the text edit line. One-button mouse users must press the Option key and mouse button together to display the local popup menu in the text edit line.

5. Select the data from the options list and click on the **OK** button. The system closes the selection dialog and pastes the selected item directly into the highlighted template field.

For certain operators you enter into the Rule editor window the local popup menu will also give you access to the list of functions, execute library routines, and system files that you can specify. The options displayed by the popup menu are determined automatically by the system for that field. For

detailed information about using the functions and execute library routines provided by the system, refer to the Language Reference manual.

Copy and Paste

Special keyboard commands let you perform copy/cut and paste operations on currently selected data entry fields. These commands are available as a result of the windowing environment and work on most hardware platforms. The keyboard commands include the following.

■ Copy: -C on the Macintosh, Ctrl-C on Unix and VAX/VMS

■ Cut: -X on the Macintosh, Ctrl-X on Unix and VAX/VMS

■ Paste: -V on the Macintosh, Ctrl-V on Unix and VAX/VMS.

The copy/cut keyboard commands place the selected data field in a buffer that you can use to paste the item back into the editor window. The difference between copy and cut is that cut deletes the selected item from the current display while copy leaves it intact. Use the following procedure to copy data displayed in the current editor window and paste it back into a data entry field.

To copy and paste displayed data:

1. Open the editor window that controls the type of structure you want to edit. Display the desired structure.

2. Position the mouse pointer on the **Modify** button in the window and click the selection mouse button.

3. Position the mouse pointer on the template field that you want to copy and click the selection mouse button. The system duplicates the field entry in the text edit line.

    One-button mouse users must take care that the mouse cursor looks like 🕀 and not 📄 when clicking on the desired field. Move the cursor slightly to the left if necessary.

4. Press the Command and C keys together on your keyboard. The system copies the item displayed in the text edit line into its memory buffer.

5. Position the mouse pointer on the target template field and click the selection mouse button. The system highlights the blank field.

6. Press the Command and V keys together on your keyboard. The system pastes the previously copied item into the text edit line.

7. Press the Return key to return the pasted item to the highlighted template field.

Additionally, the Rule editor or Method editor windows lets you select one or more lines of data before applying the above keyboard commands to the selected items. This operation is particularly useful when you want to duplicate several complex lines of these editors to modify. Use the following procedure to copy several lines of the Rule editor or Method editor windows.

To copy and paste one or more lines of data:

1. Open the editor window that controls the type of structure you want to edit. Display the desired structure.

2. Position the mouse pointer on the **Modify** button and click the selection mouse button.

3. Position the mouse pointer on the leftmost template field of the data entry line that you want to copy and press down the selection mouse button. Without releasing the mouse button, move the mouse to the rightmost field of the same line. This action highlights the fields in the line.

    One-button mouse users must take care that the mouse cursor looks like ⊹ and not ▤ when clicking on the desired field. Move the cursor slightly to the left if necessary.

4. While still pressing the mouse button, move the mouse pointer through as many lines as you want to copy. When you have enclosed the desired lines in the stretched box, release the mouse button. The selected lines remain highlighted.

5. Press the Command and C keys together on your keyboard. The system copies the highlighted lines into its memory buffer.

6. Position the mouse pointer on the leftmost template field of the first blank line and click the selection mouse button. The system highlights the blank field.

    One-button mouse users must take care that the mouse cursor looks like ⊹ and not ▤ when clicking on the desired field. Move the cursor slightly to the left if necessary.

7. Press the Command and V keys together on your keyboard. The system pastes the previously copied lines directly into the blank lines of the Rule editor window.

Append

A less often used data copying technique lets you append data. It lets you join strings of data together to form new data. Use the following procedure to append data that you type to data in the editor window.

To append data to displayed data:

1. Open the editor window that controls the type of structure you want to edit. Display the desired structure.

2. Position the mouse pointer on the **Modify** button and click the selection mouse button.

3. Position the mouse pointer on the template field that you want to modify and click the selection mouse button. The system duplicates the field entry in the text edit line as highlighted text.

    One-button mouse users must take care that the mouse cursor looks like ⊕ and not ▤ when clicking on the desired field. Move the cursor slightly to the left if necessary.

4. Position the mouse pointer where you want the appended data to appear in the text edit line and type the data to append.

5. Press the Return key to return the new data to the highlighted template field.

Application Implementation

This chapter shows how the various Intelligent Rules Element editor windows let you implement rule and object structures. It describes the data entry fields and the local popup menus of each editor window.

Introduction

There are seven editor options that you can select on the Edit menu in the Rules Element's menu bar. These editors let you create, view, and edit any aspect of the current application. They are as follows.

- Rule editor
- Context editor
- Object editor
- Class editor
- Meta-Slot editor
- Property editor
- Method editor.

All seven editor windows work together to establish different aspects of the same rule and object structures. As described in the previous chapter, you choose the particular structure to display through the editor window's browsing mechanisms. This does not mean you will always use every editor to describe the rules and objects you create. Sometimes a particular editor won't apply to your rule or object structure. Additionally, the system saves you some work by automatically distributing related information from the current editor to other editors.

This chapter describes the interrelationship of the editors in detail. It also shows how they help you maintain a fully-functioning application even during implementation. Some familiarity with the terminology of rules and objects is required. For background information refer to the Getting Started manual.

Creating a Rule

This section describes the two editors you use to create rules in the Rules Element. In general, these two editors let you specify the following aspects of the application:

■ Bi-directional links for strongly linked rules

■ Context links for weakly linked rules

■ Default property data types

■ Actions that rules initiate.

The List of Rules window also appears in this section as an aid to creating rules.

The Rule Editor

The Rule editor window lets you create, modify, or display a single rule. The window provides the minimum number of fields required to define the rule. The window's overall clean design is due in part to the bi-directionality of the Rules Element rules. The consistent use of local popup menus also contributes to the Rule editor's clean design. Popup menus you can display provide many additional options for individual fields in the editor window. Each field group (or component) includes a label that clearly identifies the

component's function. Figure 2-1 shows these fields and their associated popup menus.
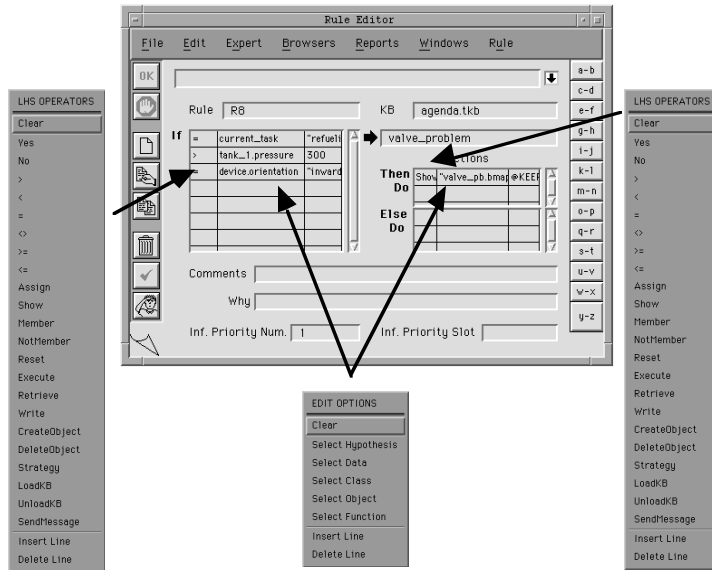
Figure 2-1   Rule Editor Window

Note: Refer to the Intelligent Rules Element Language Reference manual for information about the individual operators provided by the Rule editor popup menus.

The following sections describe the task of creating a rule in the Rule editor and do not describe the Rule Name, Comments, and Why fields. Refer instead to Chapter Six, "Application Documentation" for information about these fields.

Left-Hand Side Conditions

The Left-Hand Side (LHS) component of the Rule editor is the area of the template where you list the rule's one or more conditions, up to any number desired. The LHS component follows the "If" label to remind you that the system tests the conditions you list to determine the applicability of the rule. Tests you can perform in the LHS component include:

- TRUE/FALSE test
- Numerical test
- Numerical test on an expression
- Multi-value (or string data) test.

The LHS component comprises three fields (see Figure 2-1). The first field holds a wide array of test operators that you can select from its local popup menu. The first argument field (middle column) holds the data or expression the test operator acts on. The second argument field holds a constant that the first argument is measured against.

Hypothesis Test Conditions

You use LHS conditions to establish reasoning pathways by testing the value of *other* rules' hypotheses. The hypothesis test condition includes one of the boolean test operators, "YES" or "NO," and a hypothesis variable. Rules you link in the Rule editor are referred to as "strong links" because they maintain a cause and affect relationship. This approach lets you write your rules in a single format, leaving runtime parameters to determine whether the rule is used in forward or backward chaining. Use the following procedure to link the hypotheses of one rule to the conditions of another rule in the Rule editor.

To establish strong links:

1. Open the Rule editor and click on the desired mode button.

2. Display the popup menu for the LHS Operator field and select a boolean test operator from the list. The system automatically highlights the next field.

3. Type the desired hypothesis name in the edit line and press return when done. The system automatically highlights the next field; press the return key.

   The second argument field is left blank because constants do not apply to hypothesis variables (or other boolean variables).
   An example hypothesis test condition appears below.

   ```
        YES         Liquidity_Assessed
   ```
   Evaluates the status of a hypothesis (or other boolean variable).

4. Complete the remaining Rule editor fields and click on the **OK** button to verify and compile the rule.

Data Test Conditions

You also use LHS conditions to test data that are not hypotheses. This test involves the use of a wide array of test operators that you select for the type of data you want to test. Table 2-1 shows the relationship between test

operators and data types. Refer to the Intelligent Rules Element Language
Reference manual for details about specific operators.

Operator	Data Type	Constant You Supply
Yes or No	Boolean data	Not applicable.
Member, NotMember	Object name	A string.
Symbols: ≥, ≤, =, <>	Numeric data	A numeric value (floating point or integer) or string value (applies to = only).
Comparison Symbols: =, <>	Any data	A string or numeric value.

Table 2-1 Test Operator and Data Type Relationships

The test data used must belong to a public slot; private slot data cannot
appear in rule conditions. See "Specifying System Attributes" for more
information about public and private slots. If you specify data in the
condition that has not been already assigned a slot name, the Rules Element
automatically creates a public slot for the data. In most cases the system
determines the data type of the slot based on the operator used. This
approach lets you create rules naturally, without requiring frequent exits to
the Object editor. (See "Creating an Object" to explicitly create slots.) Use
the following procedure to automatically create objects as data in the Rule
editor.

To establish data test conditions:

1. Open the Rule editor and click on the desired mode button.

2. Display the popup menu for a blank LHS Operator field and select an
    operator from the list. The system automatically highlights the next
    field.

3. Type the data being tested in the edit line and press return when done.
    The system automatically highlights the next field.

    Unless you specified salience factors for the data in the Meta-Slot editor,
    the system evaluates rule conditions from top to bottom.

4. Type the constant being compared in the edit line and press return
    when done. Add other conditions to the list as needed. Examples of
    LHS conditions appear below.

    ```
    <        Total                    1200.50
    >        (Interest+Inflation)/2   10
    ```
    Performs a numerical test on data and an expression.

    ```
    =        Color                    "BLUE"
    ```
    Performs a multi-value test.

```
        =           Color#1       Color#2
```
Evaluates the equality of two pieces of data.

5. Complete the remaining Rule editor fields and click on the **OK** button to verify and compile the rule and its data.

   A data type selection window may appear depending on your conditions' data. Select the desired data type from the options list.

Left-Hand Side Actions

The LHS component of the Rule editor (see Figure 2-1) also lets you perform actions similar to the actions performed on the right-hand side (RHS) of the rule. Unlike RHS actions, however, LHS actions do not depend on the evaluation of other LHS conditions. Actions you enter as LHS conditions always evaluate to TRUE except in certain instances. Table 2-2 shows the relationship between action operators and data types. Refer to the Intelligent Rules Element Language Reference manual for details about specific operators.

Operator	Data Type	Constant You Supply
Assign	Any type	Any type.
CreateObject, DeleteObject	Object name	Not applicable.
Reset	Data or pattern matching	Not applicable.
Show, Load, Unload, Retrieve, Write	File Name	Not applicable.
Execute	Routine Name	Not applicable.
Strategy	Strategy Name	Not applicable.
SendMessage	Method Name	Name of addressees and arguments to pass.

Table 2-2   Action Operator and Data Type Relationships

Use the following procedure to establish as many actions as your rule requires in the Rule editor.

To establish actions:

1. Open the Rule editor and click on the desired mode button.

2. Display the popup menu for a blank LHS Operator field and select an action operator from the list. The system automatically highlights the next field.

3. Type the data being assigned, execute library routine, or filename in the edit line and press return when done. The system automatically highlights the next field.

    The local popup menu for this field lets you display the list of functions, execute library routines, and system files to copy. For detailed information about using the functions and execute library routines provided by the system, refer to the Language Reference manual.

4. Type the assigned to constant, if necessary, and press return when done. Add other actions to the list as needed. Examples of LHS actions appear below.

        EXECUTE   Real_Rate
    Performs a numerical computation and assigns an object to a list of objects.

        SHOW   Town_Map
    Displays an information file that you supply.

        RESET   Total_Cash
    Resets a datum to the value: UNKNOWN.

        STRATEGY   PWT
    Changes the forward chaining strategy of the inference engine.

5. Complete the remaining fields of the editor window as required and click on the **OK** button to verify and compile the rule and its data.

Hypothesis

The single right-hand side (RHS) component of the Rule editor window (see Figure 2-1) holds your rule's hypothesis. This component follows the "arrow" symbol to remind you that the LHS condition(s) lead to the hypothesis. The system regards the hypothesis you supply as a boolean variable with the value TRUE, FALSE, or NOTKNOWN depending on the outcome of the rule's LHS conditions. It is important to note that every LHS condition must be true to produce a true hypothesis. Use the following procedure to establish a hypothesis name for your rule in the Rule editor.

To establish a hypothesis name:

1. Open the Rule editor and complete the LHS conditions as described above.

2. Click on the Hypothesis field and type the hypothesis name of your rule in the edit line. Press return when done.

    The name you assign to a rule's hypothesis should reflect the fact that the hypothesis' value is determined *after* the system evaluates the LHS conditions. An example is "status_determined," written in past tense.

3. Complete the remaining fields of the editor window as required and click on the **OK** button to verify and compile the rule and its data.

Right-Hand Side Actions

Actions appear on the Rule editor window's RHS (see Figure 2-1) to remind you that the system executes actions depending on the evaluation outcome of the rule's LHS component. The RHS components of the Rule editor let you divide the rule's actions into two separate lists. The top list labeled "Then Do" is executed only when <u>all</u> conditions are satisfied. The bottom list labeled "Else Do" is executed in the event that at least one condition fails. Either component of the Rule editor is optional, you can establish as many actions as needed or none at all. Actions you can establish in the RHS component include:

■   Value assignments to objects
■   Message passing to trigger a method
■   Creation of objects
■   Modification of object's classes.

The "Then Do" and "Else Do" components each comprise three fields. The first field holds a wide array of action operators that you select from its local popup menu. The first argument field (middle column) holds the data or filename the operator acts on. The test data used must belong to a public slot; private slot data cannot appear in rule actions. See "Specifying System Attributes" for more information about public and private slots. The second argument field holds a constant that the first argument applies to. Refer to the Intelligent Rules Element Language Reference manual for information about specific popup menu operators. Use the following procedure to establish as many actions as your rule requires in the Rule editor.

To establish Then or Else rule actions:

1. Open the Rule editor and click on the desired mode button.

2. Display the popup menu for a blank Then or Else Operator field and select an action operator from the list. The system automatically highlights the next field.

3. Type the data being assigned, execute library routine, method name, or filename in the edit line and press return when done. The system automatically highlights the next field.

    The local popup menu for this field lets you display the list of functions, execute library routines, and system files to copy. For detailed information about using the functions and execute library routines provided by the system, refer to the Intelligent Rules Element Language Reference manual.

4. Type the assigned to constant, if necessary, and press return when done. Add other actions to the list as needed. Examples of RHS actions appear below.

```
ASSIGN    (Inflation.Rate+Interest.Rate)/2    Real_Rate
```
Performs a numerical computation and assigns an object to a list of objects.

```
ASSIGN      Object.Color            "RED"
```
Assigns a string or boolean value to an object or list of objects.

```
SHOW        Town_Map
```
Displays an information file that you supply.

```
RESET       Total_Cash
```
Resets a datum to the value: UNKNOWN.

```
STRATEGY    PWT
```
Changes the forward chaining strategy of the inference engine.

5. Complete the remaining fields of the editor window as required and click on the **OK** button to verify and compile the rule and its data.

Inference Priority Number and Priority Slot

The Inference Priority Number and Priority Slot fields of the Rule editor (see Figure 2-1) let you assign a salience factor to an individual rule. The system uses these saliencies to resolve inferencing conflicts between rules. These fields perform the same task as inference priorities you assign to slots in the Meta-Slot editor (see "Creating System Attributes"), but at a higher level of granularity. For example, saliencies that you assign to a group of rules that lead to a single hypothesis predetermine the order of rule evaluation. The two fields either accept a static or dynamic salience factor as follows.

■ The Inference Priority field is a static salience factor. This field accepts an integer number that you assign the rule directly. Valid salience factors range from -32,000 to 32,000. The higher the number, the higher the rule's priority.

■ The Inference Slot field is a dynamic salience factor. The field accepts a variable name that you assign the rule for evaluation during inferencing. The evaluated structure must still be an integer number in the range from -32,000 to 32,000. And again, the higher the number, the higher the rule's priority.

Negative inferencing priority numbers you assign have a specific meaning to the inference engine for hypotheses only. Refer to the Intelligent Rules Element Language Reference manual for detailed information about priorities. Use the following procedure to assign an inference priority in the Rule editor window for an individual rule.

To assign an inference priority for a rule:

1. Open the Rule editor and click on the desired mode button.

2. Complete the rule components if necessary and click on the Inference Priority field.

3. Type the priority number or name in the edit line and press return when done.

4. Click on the **OK** button to verify and compile the rule.

The List of Rules Window

The List of Rules window maintains the rules you create in an easy to read, textual format. After compiling your rule in the Rule editor, you can view the contents of the List of Rules window. The window opens with the last rule edited, but you can browse the window to display any rule in the current application. This window shows one rule at a time and lets you read the rule as:

■ An "IF" statement that corresponds to the rule's first condition followed by a list of "And" statements that correspond to added conditions.

■ A "THEN hypothesis_name is confirmed." statement followed by "And" statements that correspond to the rule's actions to be executed after a positive evaluation outcome of the conditions.

■ An "ELSE hypothesis_name is not confirmed." statement followed by "And" statements that corresponds to the rule's actions to be executed after a negative evaluation outcome of the conditions.

Figure 2-2 shows the textual format typical of the List of Rules window.

To view the newly created rule in textual format:

1. Create or display the rule in the Rule editor.

 The rule must be compiled to proceed, click on the **OK** button to verify and compile the rule.

2. Display the Rule editor global popup menu and select the List of Rules options from the list. The system displays the previously edited rule in the List of Rules window.
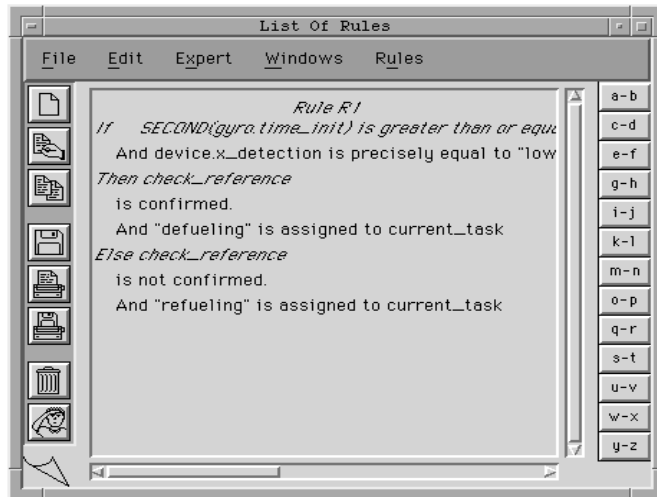
Figure 2-2   List of Rules Window

The Context Editor

The Context editor window supplements the Rule editor's ability to establish reasoning pathways. The Context editor lets you directly link an already declared hypothesis to one or several other hypotheses. This approach gives the application developer the ability to establish relationships between rules that are not strong links.

Rules that you link in the Context editor are referred to as "weak links." Instead of a cause and affect relationship, rules linked this way typically have an intuitive connection. The weak link relationship is a forwarding

link and is *not* automatically bi-directional. Figure 2-3 shows the default direction of the link you can establish in the Context editor.
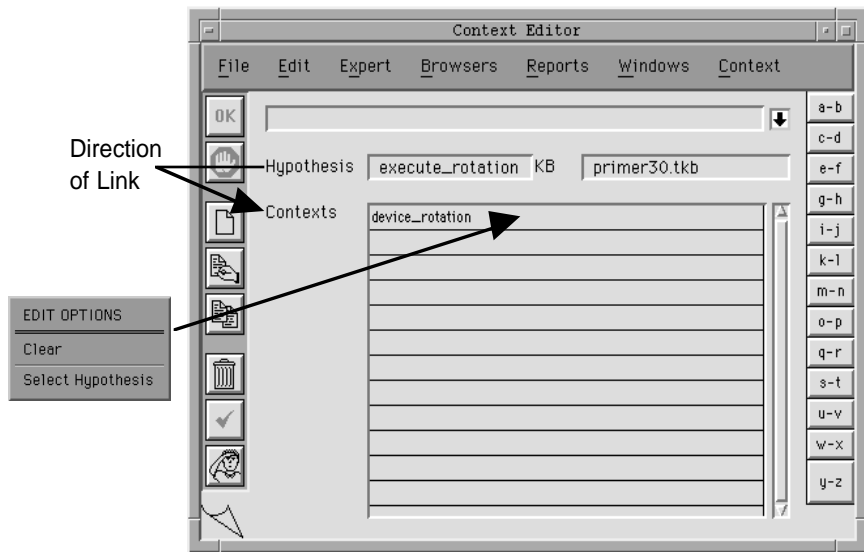
Figure 2-3 Context Editor Window

The Context editor window is comprised of two fields. The Hypothesis field holds the forwarding node of the weak link. The Contexts field is a list that holds one or several hypotheses that receive the forwarding action of the link. Use the following procedure to establish weak links between two or more rules in the Context editor.

To establish forwarding only weak links:

1. Create the rules that use the hypotheses you want to link in the Rule editor.

   If you use the Context editor to link unused hypotheses, you must later remember to establish the hypotheses in rules you create in the Rule editor.

2. Display the Rule editor global popup menu and select the Edit Context option from the list. The system opens the Context editor with the previously edited rule's hypothesis in the Hypothesis field.

   The rule you choose to display the Context editor for automatically becomes the forwarding node of the weak link as shown in Figure 2-3.

3. Click on the **Modify** button. The system automatically highlights the first Contexts field.

4. Display the popup menu for a blank Contexts field and select the Select Hypothesis option from the list. A selection window lists the hypotheses currently available.

5. Select the desired hypothesis from the options list and click on the **Paste** button. The system pastes the hypothesis in the Contexts field and highlights the next field. Add other hypotheses as needed.

6. Click on the **OK** button to verify and compile the new weak link.

The following procedure lets you establish forwarding in the reverse direction for a particular weak link. First establish the default forwarding direction of the weak link as described in the previous procedure. Use the following procedure to reverse the placement of two hypotheses in the Hypothesis and Contexts fields of the Context editor.

To establish bi-directional weak links:

1. Open the Context editor and click on the **New** button. The system automatically highlights the Hypothesis field.

2. Type the hypothesis name that you previously linked to the Hypothesis field in the Context editor and press return when done. The system automatically highlights the next field.

3. Display the popup menu for the Contexts field and select the Select Hypothesis option from the list. A selection window lists the hypotheses currently available.

4. Select the hypothesis from the options list that you previously entered in the Hypothesis field of the Context editor and click on the **Paste** button.

5. Click on the **OK** button to verify and compile the new weak link.

Creating an Object

This section describes the two editors you use to explicitly create objects in the Rules Element. In general, these editors let you specify the following aspects of the application:

■ Parent/child relationships
■ Properties of subobjects, objects, subclasses, and classes
■ Data types of properties.

The Object Editor

The Object editor window lets you create, modify, and display a single object and its related structures. The editor window provides the minimum number of fields required to define the object. Each field group (or component) includes a label that clearly identifies the component's function. Popup menus you can display provide many additional options for individual fields in the editor window. Figure 2-4 shows these fields and their associated popup menus.
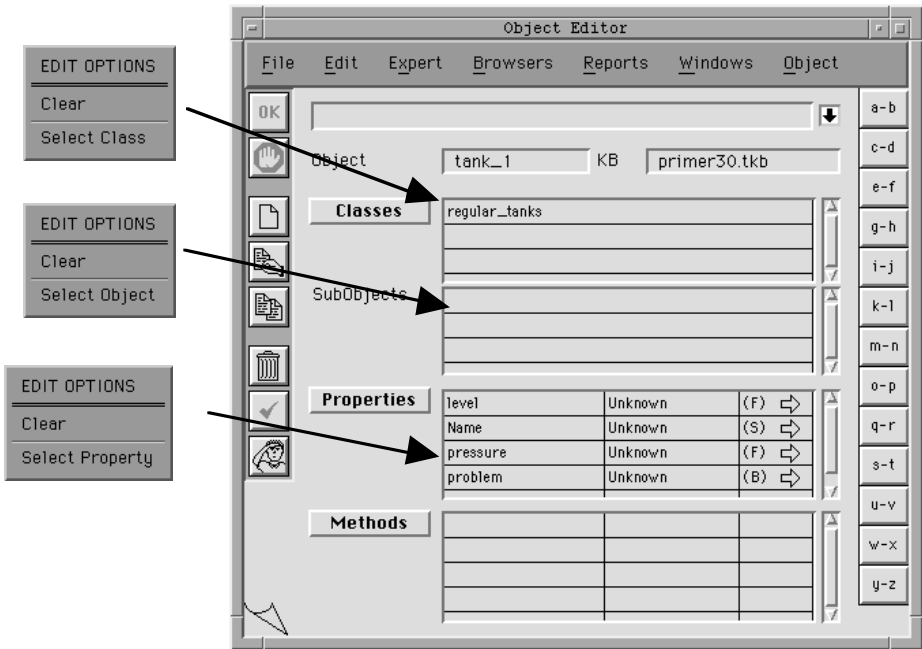
Figure 2-4 Object Editor Window

The window's Object field holds the object *or* subobject name that you want to create or modify. The Classes field is a list that holds the name of the class *or* subclasses you want to add. The SubObjects field is another list that holds the name of the subobjects you want to add. The Property component has three fields arranged by rows that let you add properties to the object to create a slot. The Method component is a display-only list of method names that have been attached to the object through the Method editor.

Object Name, Classes, and SubObjects

The first three fields of the Object editor window (see Figure 2-4) extend the application development effort started in the Rule editor window. For example, when you use the Rule editor to establish LHS conditions the system automatically creates objects and classes for the data mentioned. But the Rule editor does not create the links between the objects and classes.

To make objects and classes useful, you use the Object editor to establish hierarchical relationships. Relationships that you establish let your rules determine values from inheritance pathways at runtime. Refer to the Functional Description for more information on inheritance in rules. The Rules Element relationships are essentially of the parent/child type, as shown in the following list.

- Objects that belong to one or several classes/subclasses
- Objects that have one or several subobjects
- Subobjects that belong to one or several objects/subobjects
- Classes that have one or several objects
- Subclasses that have one or several objects.

The Rules Element also makes it easy to *create* these object structures because the system automatically compiles the objects, classes, and subobjects mentioned in the Object editor. It is therefore unnecessary to predefine anything you enter in the Object editor window. You can create objects in any order desired, using the first three fields of the Object editor or the LHS component of the Rule editor. Use the following procedure to establish objects, classes, and subobjects in the Object editor.

To establish class, object, and subobject relationships:

1. Open the Object editor and decide whether the object or subobject needs to be created.

2. If you previously mentioned the object or subobject in the LHS condition of a rule, browse the Object editor to display the desired structure in the Object field and click on the **Modify** button. The system automatically highlights the first Classes field.

3. If the object or subobject is not currently displayed in the Object field, click on the **New** button. Type the object or subobject name in the edit line and press return when done. The system automatically highlights the next field.

4. Click on a blank Classes field, if necessary, and type the class or subclass name in the edit line. Press return when done and add others to the list as needed.

   See "Class Editor" to establish actual subclasses. The Object editor does not distinguish between classes and subclasses.

5. Click on the first SubObjects field and type the subobject name in the edit line. Press return when done and add other subobjects to the list as needed.

6. Click on the **OK** button to verify and compile the object, class, and subobject structures.

   The items that you entered into the fields automatically become the currently displayed object or subobject's parent/child structures.

Properties

The Properties component of the Object editor window (see Figure 2-4) holds the object's or subobject's individual properties. The Property component is comprised of three fields arranged by rows. The first field holds the name of the property you wish to add. The second field holds the property's corresponding values you list. The third field identifies the data type of the values and displays an arrow button that lets you open the Meta-Slot editor with that specific property and object (called a slot) displayed. The third field also indicates whether the slot is private (Priv) or public (Pub). To change whether a slot is private or public, use the Meta-Slot editor.

Properties that you associate with objects form a fundamental unit in the Rules Element known as a slot. The slot stores a data value for the object and appears in rule and method conditions and actions. Slots also let your rules perform more complex test conditions called "pattern matching" as described in the Intelligent Rules Element Language Reference manual. The slot has system attributes that you specify in the Meta-Slot editor, including whether the slot is public or private. After you customize attributes through the Meta-Slot editor window, the arrow displayed in the third field appears darkened.

Note: To view previously defined system attributes of a given slot, click on the arrow button in the property field. The system opens the Meta-Slot editor in view mode.

Use the following procedure to establish properties in the Object editor.

To establish properties of an object or subobject:

1. Open the Object editor and browse the editor to display the desired object or subobject in the Object field.

2. Click on the **Modify** button. The system automatically highlights the first Classes field.

3. Click on the first blank field of the Property component (see Figure 2-4) and type the property name in the edit line. Press return when done and add other properties to the list as needed.

4. Click on the **OK** button to verify the entries. The system displays a data type menu for each new property.

5. Select a data type and click on the **OK** button. The system compiles the object.

Methods

The Methods component of the Object editor window lets you view the list of methods that have been attached to the currently displayed object or one of its properties. See the Method Editor section for more information about attaching methods to objects and properties. The first column displays the method name, the second column displays the property name if the method is actually attached to a slot that includes the currently displayed object (otherwise this field is blank), the third column indicates whether the method is public or private and displays an arrow button that lets you open the Method editor with that specific method displayed. The word "Method" for the entire component is also a button that lets you view the Method editor. None of the fields in this component are editable.

The Class Editor

The Class editor window lets you create, modify, and display a single class and its related structures. The editor window provides the minimum number of fields required to define the class. Each field group (or component) includes a label that clearly identifies the component's function. Popup menus you can display provide options for individual fields in the

editor window. Figure 2-5 shows these fields and their associated popup menus.
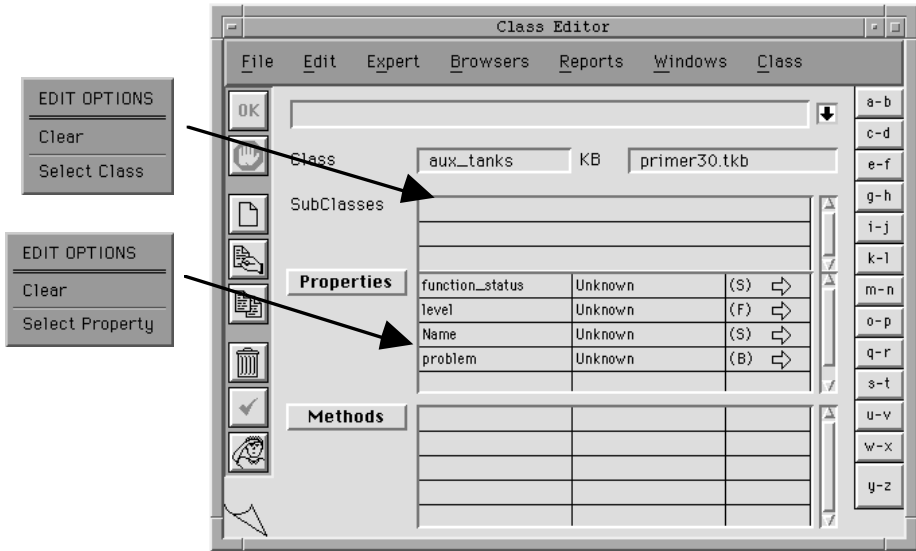
Figure 2-5   Class Editor Window

The Class field holds the class or subclass name that you want to create or modify. The SubClasses field is a list that holds the names of particular subclasses you want to add. The Property component has three fields that let you add properties to the class. The Method component is a display-only list of method names that have been attached to the class through the Method editor.

Class Name and SubClasses

The first two fields of the Class editor window (see Figure 2-5) enhance the inheritance pathways already started with the Object editor. These fields let you define additional parent/child relationships as follows.

■ Classes that have one or several subclasses
■ Subclasses that belong to one or several classes.

The Rules Element also makes it easy to *create* these object structures because the system automatically compiles the classes and subclasses mentioned in the Class editor. It is therefore unnecessary to predefine anything you enter in the Class editor window. Although the system already created the classes that you listed in the Object editor, you must use the Class editor to create *sub*classes. Use the following procedure to establish classes and subclasses in the Class editor.

To establish subclass relationships:

1. Open the Class editor and decide whether the class or subclass needs to be created.

2. If you previously mentioned the class or subclass in the LHS condition of a rule, browse the Class editor to display the desired structure in the Class field and click on the **Modify** button. The system automatically highlights the first SubClasses field.

3. If the class or subclass is not currently displayed in the Class field, click on the **New** button. Type the name of the class or subclass in the edit line and press return when done. The system automatically highlights the next field.

4. Click on a blank SubClasses field, if necessary, and type the subclass name in the edit line. Press return when done and add other subclasses to the list as needed.

5. Click on the **OK** button to verify and compile the class and subclass structures.

   The subclasses that you entered automatically become the currently displayed class or subclass' parent/child structures.

Properties

The Properties component of the Class editor window (see Figure 2-5) holds the properties of template classes and template subclasses. The Property component is comprised of three fields arranged by rows. The first field holds the name of the property you wish to add. The second field holds the property's corresponding values you list. The third field identifies the data type of the values and displays an arrow button that lets you open the Meta-Slot editor with that specific property and class (called a slot) displayed. The third field also indicates whether the slot is private (Priv) or public (Pub). To change whether a slot is private or public, use the Meta-Slot editor.

Templates that you establish let objects you create inherit these properties from their parent classes. This powerful capability is especially useful in situations where you want objects to inherit the properties of a certain class at runtime. See the Functional Description for information about the purpose of classes. Properties also let you define system attributes for the classes and objects they belong to. After you customize attributes through the Meta-Slot editor window, the arrow displayed in the third field appears darkened.

> **Note:** To view previously defined system attributes of a given slot, click on the arrow button in the property field. The system opens the Meta-Slot editor in view mode.

Use the following procedure to establish properties in the Class editor.

To establish properties of a class or subclass:

1. Open the Class editor and browse the editor to display the desired class or subclass in the Class field.

2. Click on the **Modify** button. The system automatically highlights the first SubClasses field.

3. Click on the first blank field of the Property component (Figure 2-5) and type the property name in the edit line. Press return when done and add other properties to the list as needed.

4. Click on the **OK** button to verify the entries. The system displays a data type menu for each new property.

5. Select a data type and click on the **OK** button. The system compiles the object.

Methods

The Methods component of the Class editor window lets you view the list of methods that have been attached to the currently displayed class or one of its properties. See the Method Editor section for more information about attaching methods to classes and properties. The first column displays the method name, the second column displays the property name if the method is actually attached to a slot that includes the currently displayed class (otherwise this field is blank), the third column indicates whether the method is public or private and displays an arrow button that lets you open the Method editor with that specific method displayed. The word "Method" for the entire component is also a button that lets you view the Method editor. None of the fields in this component are editable.

Creating System Attributes

This section describes the Meta-Slot editor, Property editor, and Method editor features that you will use to customize the Rules Element object structures. Customization of the rule and objects that you create fall into two general categories: attributes and behavior. This type of customization extends the Rules Element's capabilities as an object-oriented application development tool. In general, you can specify the following aspects of the application:

Customized attributes:

- The accessibility of a slot value by rules and methods.
- The global predefinition of slot types.
- The format of a slot's question and value.
- The constraints of a slot's value.
- The initial value of a property associated with a specific object.

User-defined behavior:

- The order in which the system determines a slot value.
- The actions to execute whenever a slot value changes.
- The inheritability of a property or value by other objects or classes.
- The order of evaluation of rule conditions and hypotheses.
- The order of inheritance between conflicting properties.
- The inheritance search strategies used to obtain properties, values, and methods.
- The name of a window in which the question for the slot will be asked.

The Meta-Slot Editor

The Meta-Slot editor window lets you create, modify, and display a variety of features for properties that appear in the Object and Class editors (called "slots"). Some of these features (or "meta-slots") modify the default behavior of a given slot that the system tests during inferencing. Other features of the Meta-Slot editor let you define the static attributes of a slot.

Each component of the Meta-Slot editor window includes a label that clearly identifies the component's function. Figure 2-6 shows these components.
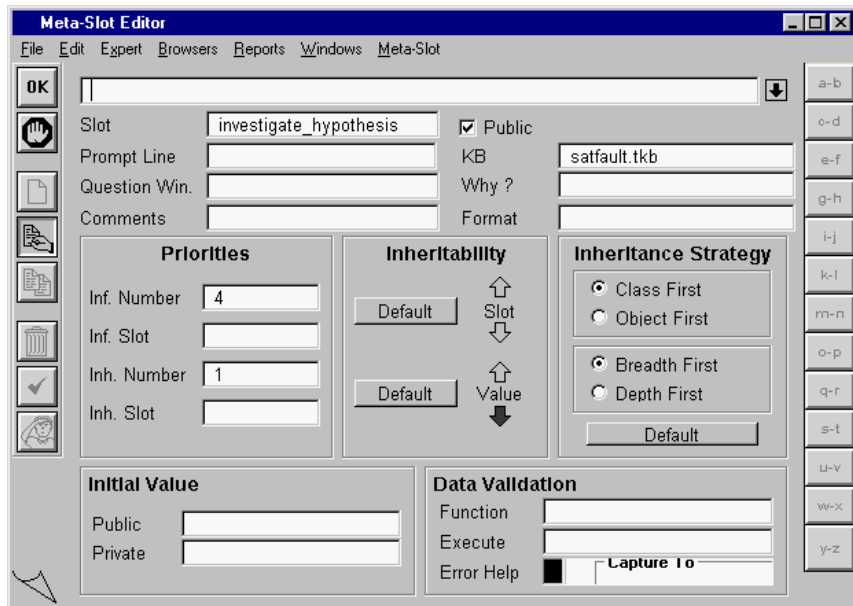
Figure 2-6   Meta-Slot Editor Window

Customizing meta-slots is an optional operation, because the system usually supplies a default behavior or attribute. Use the Meta-Slot editor to override the default behavior only as required for your particular application. Refer to the Intelligent Rules Element Language Reference manual for detailed information about individual meta-slots and their default behavior. Table

2-3 shows the Meta-Slot editor components that you can customize for a given slot.

Component	Description
Public/Private	Lets you determine whether a slot value can be accessed by rules and methods (public) or by a method attached to the slot's associated object only (private).
Why	Lets you provide a reason to the end user when the system solicits a value during inferencing.
Format	Lets you define an acceptable range of values for input solicited from the end user during inferencing.
Prompt Line	Lets you customize the default question the system displays to obtain slot values from the end user.
Question Window	Lets you replace the default question window with your own custom window to solicit input from the end user during inferencing.
Priorities	Lets you predetermine the order that the system evaluates rule LHS conditions, hypotheses, and inheritance pathways.
Inheritability	Lets you enable or disable the inheritance of properties and slot values by other object structures.
Inheritance Strategy	Lets you preselect object network pathways the system uses to inherit properties and slot values.
Initial Value	Lets you give the slot an initialization value for use during inferencing. Because the value may be inherited, it can be specified as either Public (inheritable) or Private (not inheritable).
Data Validation	Lets you define an acceptable range of values for input solicited from the end user during inferencing.

Table 2-3   Meta-Slot System Attributes

The following sections describe procedures for creating custom behavior and attributes for a given slot. Public slots that you customize can appear anywhere in the application, including the conditions and actions of rules and methods; hypotheses are also slots. Private slots that you customize can appear only in the method conditions and actions attached to the slot's object or property. It is important to note that a slot need not be explicitly identified in the rule or method if the condition or action uses pattern matching.

Public/Private

The Public/Private option button in the top portion of the Meta-Slot editor window lets you change the accessibility of the data value stored by the slot

during application processing. This is fundamentally different from assigning an initial slot value that is "private" as described in the Initial Value section. If you make the slot private, the slot name cannot appear in a rule condition or action. This requirement provides data protection since the value of a private slot cannot be affected by rule processing. Note, a public slot that has a private Initial Value remains accessible in rule conditions and actions, but the initial value cannot be inherited.

Slots that you create can have their privacy reversed by selecting the Private slot option button. Changing a public slot's status to one that is private may produce a dependency warning by the Rules Element if the slot name is used improperly.

> **Note:** Private slots that you create can only appear in a method attached to the slot's object or property. Refer to the Language Reference manual for more information about the usage of slots.

Use the following procedure to change a public slot to a private slot.

To change a slot's privacy status:

1. Open the Class or Object editor and browse the editor to display the class or object of the desired slot.

   If the class or object is undefined, you can type the slot name in the Slot field of the Meta-Slot editor; the system automatically creates a new class or object and new property based on the slot name.

2. Click on the arrow button that appears on the same line as the desired property name. The system opens the Meta-Slot editor with the selected slot displayed.

   The arrow button appears as an outline until meta-slot attributes are defined.

3. Click on the **Modify** button. The system automatically highlights the first field.

4. Click on the Public/Private option button and select the desired option.

   Refer to the Intelligent Rules Element Language Reference manual for more information about private and public slots.

5. Click on the **OK** button to verify and compile the system attributes.

Why, Format

The fields in the top portion of the Meta-Slot editor window let you customize several aspects of the session interface. The Why field holds the

rationale you provide to explain why the system needs the value. The Format field holds a string that you enter to determine the slot value format.

Note: Refer to Chapter Six, "Application Documentation" for detailed information about the Prompt Line, Comments, and Why fields.

The Property editor window lets you establish global property formats. See "Format Component" of the Property editor window for more information about formats. Use the following procedure to establish system display attributes for a given slot.

To establish system display properties for a slot:

1. Open the Class or Object editor and browse the editor to display the class or object of the desired slot.

    If the class or object is undefined, you can type the slot name in the Slot field of the Meta-Slot editor; the system automatically creates a new class or object and new property based on the slot name.

2. Click on the arrow button that appears on the same line as the desired property name. The system opens the Meta-Slot editor with the selected slot displayed.

    The arrow button appears as an outline until meta-slot attributes are defined.

3. Click on the **Modify** button. The system automatically highlights the first field.

4. Type the desired why string in the edit line and press return when done. The system automatically highlights the next field.

5. Type the desired format string in the edit line and press return when done. The system automatically highlights the next field. Complete the remaining fields as needed.

    Refer to the Intelligent Rules Element Language Reference manual for more information about format strings.

6. Click on the **OK** button to verify and compile the system attributes.

Prompt Line, Question Window

The Prompt Line field holds your wording of the question that prompts the end user to enter a public slot value. The Question Window field in the top portion of the Meta-Slot editor window lets you replace the system's default question that appears in the main window, with a separate window of your own design. The custom question window is part of your application's

user-interface that you can attach to a slot as a Rules Element resource and open at the appropriate time.

The Resource option on the Browsers menu gives you access to the runtime interface builder. Create the window resource as described in the Open Interface Elements User's Guide. Use the following procedure to attach an existing custom question window resource to a given public slot in the Meta-Slot editor. Prompt lines and question windows may not be attached to private slots since the value obtained from the end user cannot be used to update the private slot.

To specify a custom question window and prompt line for a slot:

1. Open the Class or Object editor and browse the editor to display the class or object of the desired slot.

   If the class or object is undefined, you can type the slot name in the Slot field of the Meta-Slot editor; the system automatically creates a new class or object and new property based on the slot name.

2. Click on the arrow button that appears on the same line as the desired property name. The system opens the Meta-Slot editor with the selected slot displayed.

   The arrow button appears as an outline until meta-slot attributes are defined.

3. Click on the **Modify** button. The system automatically highlights the first field.

4. Click on the Prompt Line field and type the desired question wording in the edit line and press return when done. The system automatically highlights the next field.

   The Prompt Line field accepts variables, refer to Chapter Six, "Application Documentation" for detailed information about variables.

5. Click on the Question Window field and type the full resource name `Modulename.WindowName` in the edit line. Press return when done.

   Creating and naming window resources is described in the Open Interface Elements User's Guide.

6. Click on the **OK** button to verify and compile the system attributes.

Priorities

The Priorities component of the Meta-Slot editor window (see Figure 2-6) has several fields that let you assign salience factors to individual slots. The system uses these saliencies to resolve inferencing and inheritance conflicts. These fields perform the same task as the Inference Priorities fields of the

Rule editor (see "Creating a Rule"), but at a lower level of granularity. For example, saliencies that you assign to slots contained in rule LHS conditions predetermine the order of condition evaluation or pattern matching evaluation. The Inf(erencing) and Inh(eritance) fields each accept a static and dynamic salience factor as follows.

■ The Number fields are the static salience factors. These fields accept an integer number that you assign the slot directly. Valid salience factors range from -32,000 to 32,000. The higher the number, the higher the slot's priority.

■ The Slot fields are the dynamic salience factors. The field accepts a variable name that you assign the slot for evaluation during inferencing. The evaluated slot must still be an integer number in the range from -32,000 to 32,000. And again, the higher the number, the higher the slot's priority. The variable name must not be a private slot; only public slots are valid as dynamic salience factors.

Negative inferencing priority numbers you assign have a specific meaning to the inference engine for hypotheses only. Refer to the Intelligent Rules Element Language Reference manual for detailed information about Priorities. Use the following procedure to establish inferencing and inheritance priorities in the Meta-Slot editor, for a given slot.

To establish inferencing and inheritance priorities for a slot:

1. Open the Class or Object editor and browse the editor to display the class or object of the desired slot.

   If the class or object is undefined, you can type the slot name in the Slot field of the Meta-Slot editor; the system automatically creates a new class or object and new property based on the slot name.

2. Click on the arrow button that appears on the same line as the desired property name. The system opens the Meta-Slot editor with the selected slot displayed.

   The arrow button appears as an outline until meta-slot attributes are defined.

3. Click on the **Modify** button. The system automatically highlights the Slot field.

4. Click on the desired Priorities field and type the number or atom name in the edit line, and press return when done. The system automatically highlights the next field in the Priorities component.

5. Click on the **OK** button to verify and compile the system attributes.

Inheritability

The Inheritability component of the Meta-Slot editor window (see Figure 2-6) has two graphic displays that give you control over the relative priority of inheritance. The default behavior of the system lets objects inherit properties down from classes only. You can modify this behavior, for example, to let classes inherit properties from objects as well. Refer to the Intelligent Rules Element Language Programmer's Guide for detailed information about inheritance.

The Slot display lets you determine whether other object structures can inherit the property of the current object or class. The Value display lets you determine whether other object structures can inherit the property value of the current object or class. The arrows that you select for these displays determine the direction of inheritance as follows.

■　The up arrow means that the parents of the object or class can inherit up from their child.

■　The down arrow means the children of the object or class can inherit down from their parent.

Inheritability either from the slot or into another slot is unaffected by whether the slot is public or private. Use the following procedure to override the default inheritability scheme in the Meta-Slot editor, for a given slot.

To establish the inheritability of a slot by other slots:

1.　Open the Class or Object editor and browse the editor to display the class or object of the desired slot.

　　If the class or object is undefined, you can type the slot name in the Slot field of the Meta-Slot editor; the system automatically creates a new class or object and new property based on the slot name.

2.　Click on the arrow button that appears on the same line as the desired property name. The system opens the Meta-Slot editor with the selected slot displayed.

　　The arrow button appears as an outline until meta-slot attributes are defined.

3.　Click on the **Modify** button. The system automatically highlights the first field.

4.　Click on the Slot display arrows to select or unselect them as desired. The system highlights the arrow to show its selected status.

　　Click on the "Default" boxes to return a slot to its default inheritability behavior.

5. Click on the Value display arrows to select or unselect them as desired. The system highlights the arrow to show its selected status.

   Click on the "Default" boxes to return a slot to its default inheritability behavior.

6. Click on the **OK** button to verify and compile the system attributes.

Inheritance Strategy

The Inheritance Strategies component of the Meta-Slot editor window (see Figure 2-6) gives two sets of checkboxes that determine the pathways through the class and object network. The system follows certain predetermined pathways in order to propagate data validation information, methods, properties, and property values. You can override the system's default search pathways by choosing one of the following strategy checkbox combinations.

■ Class-First, Breadth-First search strategy.
■ Class-First, Depth-First search strategy.
■ Object-First, Breadth-First search strategy.
■ Object-First, Depth-First search strategy.

Note: The Default button returns the search strategy to the global strategy as determined by the Strategy Monitor settings. You can select the Strategy option on the Expert menu to globally change the default inheritance strategy.

In case the slot is attached to a class and an object, the Class and Object checkboxes let you select the search starting point. The Breadth and Depth checkboxes let you determine the search order from the selected parent. Use the following procedure to override the default inheritance strategy in the Meta-Slot editor, for a given slot.

To establish the inheritance search strategy for a slot:

1. Open the Class or Object editor and browse the editor to display the class or object of the desired slot.

   If the class or object is undefined, you can type the slot name in the Slot field of the Meta-Slot editor; the system automatically creates a new class or object and new property based on the slot name.

2. Click on the arrow button that appears on the same line as the desired property name. The system opens the Meta-Slot editor with the selected slot displayed.

   The arrow button appears as an outline until meta-slot attributes are defined.

3. Click on the **Modify** button. The system automatically highlights the first field.

4. Click on the Class or Object checkbox to select the beginning point of the inheritance pathway as desired. The system highlights the checkbox to show its selected status.

5. Click on the Breadth First or Depth First checkbox to determine how the search will proceed through the inheritance pathways as desired. The system highlights the checkbox to show its selected status.

    Click on the Default button to return a slot to the global default inheritance search strategy.

6. Click on the **OK** button to verify and compile the system attributes.

Initial Value

The Initial Value fields along the bottom of the Meta-Slot editor window let you specify a value to begin inferencing for a given slot. The system provides the value to the slot automatically once inferencing is initiated. The value you define can be either public or private. If the value is private, it cannot be inherited by other slots. Use the following procedure to establish an initial value for a given slot in the Meta-Slot editor. Note that the Initial Value field (public or private) has no affect on the slot's public or private status. For instance, a private slot that has a public initial value remains accessible from a method only.

To establish an initial value for a slot:

1. Open the Class or Object editor and browse the editor to display the class or object of the desired slot.

    If the class or object is undefined, you can type the slot name in the Slot field of the Meta-Slot editor; the system automatically creates a new class or object and new property based on the slot name.

2. Click on the arrow button that appears on the same line as the desired property name. The system opens the Meta-Slot editor with the selected slot displayed.

    The arrow button appears as an outline until meta-slot attributes are defined.

3. Click on the **Modify** button. The system automatically highlights the first field.

4. Click on the Initial Value Public or Private field and type the desired value in the edit line and press return when done.

    If the value you type is a string, it must appear between double quotes

("").

5.   Click on the **OK** button to verify and compile the system attributes.

Data Validation

The Data Validation fields along the bottom of the Meta-Slot editor window let you specify an acceptable numeric range, list of strings, or more complex constraint for a slot whose value will be determined during processing. The system checks the end user's input or a computational result against the list and displays an error message if the entry is out of range.

> **Note:** By default data validation is disabled globally. If you specify data validation parameters, and you want the system to use them, you must enable Data Validation in the Strategy Monitor window.

Data validation can be defined at the level of the slot in the Meta-Slot editor or at the level of the property in the Property editor. If the data validation function is not defined at the level of the slot, the system will obtain one from an inheritable property. The three Data Validation fields in either editor are the same:

Function field	Boolean expression that checks the validity of the new value for the slot. The slot must be referenced by SELF. See the Intelligent Rules Element Language Reference Manual for details.
Execute field	An external routine installed through the `NXP_SetHandler` call that specifies data validation requirements. The routine must return `TRUE` or `FALSE`.
	Note: The Execute field routine is processed after the Function field expression when present. If the Execute and the Function results do not agree, the Execute routine result takes priority.
Error Help field	Alert window help string to display when an incorrect value is entered. It can be made dynamic by using the `@V()` and `@SELF` syntax. If no help string is specified, the system displays a default alert window with the options `ABORT`, `ALLOW`, and `RETRY`.

Use the following procedure to define acceptable values for the slot in the Meta-Slot editor.

To establish data validation for a slot:

1.   Open the Class or Object editor and browse the editor to display the class or object of the desired slot.

   If the class or object is undefined, you can type the slot name in the Slot

field of the Meta-Slot editor; the system automatically creates a new class or object and new property based on the slot name.

2. Click on the arrow button that appears on the same line as the desired property name. The system opens the Meta-Slot editor with the selected slot displayed.

    The arrow button appears as an outline until meta-slot attributes are defined.

3. Click on the **Modify** button. The system automatically highlights the first field.

4. Optionally, click on the Function field and type the boolean expression in the edit line and press return when done. The system automatically highlights the next field.

5. Optionally, click on the Execute field and type the name of the routine to execute in the edit line and press return when done. The system automatically highlights the next field.

    The Execute field routine is processed after the Function field expression when present.

6. Optionally, click on the Error Help field and type the message in the edit line and press return when done. The system automatically highlights the next field.

    If no help message is entered, the system supplies a default error message with the options abort, allow, and retry.

7. Click on the **OK** button to verify and compile the system attributes.

The Property Editor

The Property editor window supplements the Rule and Object editor's ability to define properties. The Property editor's most basic use is as a notebook that lets you rapidly browse the data types of properties already declared in the Rule or Object editors and any methods declared in the Method editor. You can also use the Property editor to establish one of seven types of properties before you declare them in a rule. Less frequently performed operations of the Property editor include globally modifying or

deleting properties. Figure 2-7 shows the Property editor fields; it has no
unique popup menus associated with it.
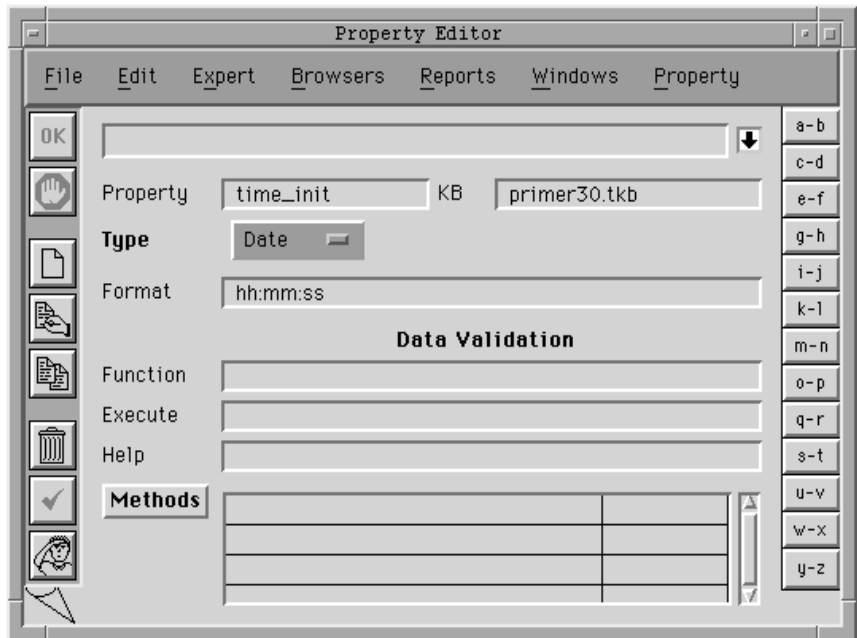
Figure 2-7   Property Editor Window

☞   Operations you perform in the Property editor window affect properties
throughout the application and may have consequences for more than one
object or class.

The Property editor window does not show the value(s) of the property
displayed. This reflects the fact that the Property editor does not distinguish
between the same property associated with different objects and classes.
You can use the Object editor to view the current value of a specific object's
property. The following sections describe ways to make global property
modifications in the Property editor.

Property Name and Type

The first two components of the Property editor window (see Figure 2-7) let
you globally specify the type of a property before you declare it in a rule.
Sometimes this approach can prevent the system from incorrectly assigning
a property type for data you establish in the LHS condition of a rule. Table

2-4 shows the fields you can select to establish one of seven, valid property types.

Field	Description
Boolean	This field lets you store the specific property value TRUE, FALSE, or NOTKNOWN. Corresponds to the operators Yes and No.
Integer	This field lets you store integer numbers (numbers that do not contain a decimal point).
Float	This field lets you store floating point numbers (numbers with a decimal point) with up to 16 digits of precision.
String	This field lets you store alphanumeric property values.
Date	This field lets you manipulating years, months, days, hours, minutes, or seconds. See "Format Component" below.
Time	This field lets you represent the difference between two times or two dates. See "Format Component" below.
Special	The system automatically selects this field for the default property "Value" (when using objects without a specified property for example).

Table 2-4   Property Types

Use the following procedure to establish a property type in the Property editor.

To globally establish property types:

1. Open the Property editor and click on the desired mode button.

2. Type the property name in the edit line and press return when done. The system automatically highlights the next field.

3. Click on the checkbox that corresponds to the data type you want to establish for the currently displayed property. The system highlights the small square to indicate its selected status.

4. Click on the **OK** button to verify and compile the property.

The Property editor also lets you globally modify or delete the currently displayed property throughout the entire application. However, the system first displays a confirmation window to remind you of the global consequences. Property editor deletions are not reversible since the system completely removes the structure from the application. Modifications to a property, on the other hand, are easily undone by the same procedure used to modify the property.

☞ Property editor deletions also remove any structures that depend on the specified property. This can include meta-slots and rule conditions, as well as objects and classes.

Use the following procedure to globally delete properties in the Property editor.

To globally delete properties:

1. Open the Property editor and browse the editor to display the desired property in the Property field.

2. Click on the **Delete** button. The system automatically displays a confirmation window.

3. Click on the **OK** button in the confirmation window to proceed. The system automatically displays a dependencies window for the current property.

   Click on the Cancel button to abort the operation.

4. Click on the **Dependencies** button. The system lists the current property's related slots (slot = object.property or class.property).

   If the property appears in a rule expression or pattern matching condition, only the number of the rule is listed.

5. You can cancel the deletion operation by clicking on the **Abort** button, or you can "**Delete It Anyway**" by clicking on that button.

   Deleting the property will result in the deletion of all related rule and object structures from the application.

Format

The Format field of the Property editor window (see Figure 2-7) serves two purposes related to the global format of the currently displayed property.

■ It lets you specify how the system formats values stored by the property for output to the display, databases, or data files.

■ It lets you specify how the system converts incoming text strings received from end users, databases, data files, or the application programming interface to the data types stored by the property.

> **Note:** You can also format *specific* slots in the Meta-Slot editor. Formats you establish for specific slots take priority over the global property format definition.

Property formatting is an optional operation. If no format appears in the Property editor (or Meta-Slot editor), the system obtains a default format from the `ckbres.format` module in the `nxrun.dat` database file. You

can customize this file to match local conventions (in dates, for example: month before day, versus day before month).

If you want to establish a global format for properties, enter the format information as a string in the Format field of the Property editor. The syntax of these strings depends on the property type selected. Refer to the Intelligent Rules Element Language Reference manual for detailed information about the syntax of property type formats. Use the following procedure to establish global formatting of values stored in a property.

To globally establish property formats:

1. Open the Property editor and complete the Property and Type fields. The system automatically highlights the Format field.

2. Type the desired format string for the selected property type in the edit line and press return when done.

3. Click on the **OK** button to verify and compile the property.

Data Validation

The Data Validation fields of the Property editor window let you specify an acceptable numeric range, list of strings, or more complex constraint for a property which will be inherited by an object during processing and whose value will be determined by the end user. The system checks the end user's input against the list and displays an error message if the entry is out of range.

Note: By default data validation is disabled globally. If you specify data validation parameters, and you want the system to process them, you must enable Data Validation in the Strategy Monitor window.

Data validation can be defined at the level of the property in the Property editor or at the level of the slot in the Meta-Slot editor. If the data validation function is already defined at the level of the slot, the system will not obtain one from the property. The three Data Validation fields in either editor are the same:

Function field	Boolean expression that checks the validity of the value entered for the slot to which the property is a component. The slot must be referenced by SELF. See the Intelligent Rules Element Language Reference manual for details.
Execute field	An external routine installed through the NXP_SetHandler call that specifies data validation requirements. The routine must return TRUE or FALSE.

Note: The Execute field routine is processed after the Function field expression when present. If the Execute and the Function results do not agree, the Execute routine result takes priority.

Error Help field   Alert window help string to display when an incorrect value is entered. It can be made dynamic by using the `@V()` and `@SELF` syntax. If no help string is specified, the system displays a default alert window with the options `ABORT`, `ALLOW`, and `RETRY`.

Use the following procedure to define acceptable values for a property in the Property editor.

To establish data validation for a property of an object:

1. Open the Property editor and browse the editor to display the desired property in the Property field.

2. Click on the **Modify** button. The system automatically highlights the first field.

3. Optionally, click on the Function field and type the boolean expression in the edit line and press return when done. The system automatically highlights the next field.

4. Optionally, click on the Execute field and type the name of the routine to execute in the edit line and press return when done. The system automatically highlights the next field.

   The Execute field routine is processed after the Function field expression when present.

5. Optionally, click on the Error Help field and type the message in the edit line and press return when done. The system automatically highlights the next field.

   If no help message is entered, the system supplies a default error message with the options abort, allow, and retry.

6. Click on the **OK** button to verify and compile the system attributes.

Methods

The Methods component of the Property editor window lets you view the list of methods that have been attached to the currently displayed property. The first column displays the method name and the second column indicates whether the method is public or private and displays an arrow button that lets you open the Method editor with that specific method displayed. The word "Method" for the entire component is also a button that lets you view the Method editor. None of the fields in this component are editable.

> **Note:** To view the methods that have been attached to a slot (and not the property of that slot), you must open the Class and Object editors and view the class or object name of the slot; the corresponding property of that slot will appear in the Methods component.

The Method Editor

The Method editor window lets you create, modify, and display a single method. Like meta-slots, methods specify the behavior of a particular slot, class, or object; they do not exist independently from other object structures. Methods let you define a custom set of actions to be invoked for the object during inferencing. The Method Editor window provides the templates and popup menus required to create user-defined methods for any object in the application. Figure 2-8 shows the Method Editor fields and popup menus.
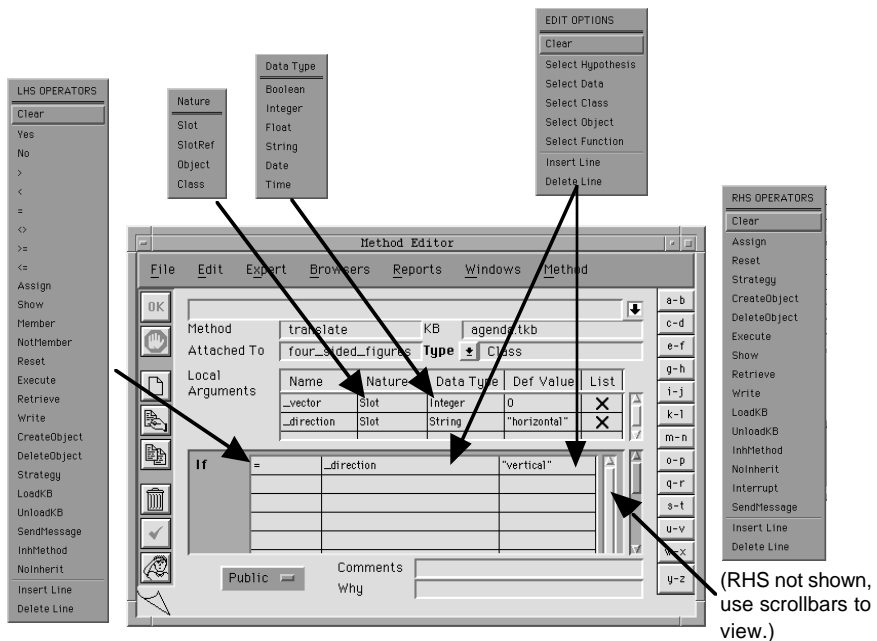

Figure 2-8   Method Editor Window

A method is primarily a list of actions that can modify the behavior of a given slot (public or private), object, class, or property during inferencing. The method can include optional test conditions that the system evaluates before executing the actions. A method can be triggered and its actions

executed by the system under special circumstances, or by the application developer through a rule or other method (using the SendMessage operator). The first case is referred to as a system method and the second case is a user-defined method. The following sections describe the task of creating methods in the Method editor, including either user-defined or system methods.

Conditions and Actions (If, Then, Else)

Methods that you create in the Method editor can be either conditional or non-conditional. This means that once the method is triggered, its list of actions can be executed immediately, or they can be executed subject to a set of test conditions. The Method editor has the following components that let you define method conditions and actions:

IF component	Lets you specify the optional test conditions.
THEN component	Lets you specify a set of actions to be executed 1) when all conditions listed in the IF component are satisfied, or 2) in the event no conditions appear in the IF component.
ELSE component	Lets you specify a set of actions to be executed in the event any one of the conditions listed in the IF component fails during method evaluation.

Data that appears in conditions can be represented by any public slot. The use of a private slot in the method, however, is restricted to the slot to which the method is attached. If the private slot is to be used in any of the IF, THEN, ELSE components, it must be referenced by SELF.propname in order to preserve data encapsulation.

Note: The IF component of the method is completely optional. By omitting the conditions list, you can create a non-conditional list of actions to be executed by the system under the appropriate circumstances.

User-Triggered Methods

The Method editor template lets you create methods that another rule or other method will explicitly trigger during inferencing. This type of method is known as a "user-defined method" because the application developer decides when to trigger it in the application. The user-defined method is always triggered through a corresponding SendMessage operator that appears in a condition or action of a rule or other method. The system executes every action from top to bottom of the method triggered by a SendMessage operator. If the rule or method that contains the operator is not evaluated during inferencing, the method will not be triggered.

When a SendMessage action is executed during inferencing, the system will trigger the method attached to any class, object, or slot named in the operator's list of addressees. If arguments have been specified along with the SendMessage operator, they are passed to the same list of addressees. This requirement makes methods that accept arguments by definition "user-defined" (it is not possible to pass arguments to system methods unless they are explicitly triggered by the SendMessage operator).

The multi-column component in the top portion of the Method editor lets you define how arguments that are passed to the method will be used. The fields of the Local Arguments component act as a template for the arguments to be passed. The arguments themselves are specified along with the SendMessage operator that you add to the conditions or actions of a rule or method. Local popup menus for each field in the Local Arguments component of the Method editor let you select the appropriate information for the argument template. The fields have the following usage:

Name	It must begin with an underscore (_) even within a list.
Nature	Slot - passed by value. The value is copied locally in the local argument and modifying the local argument will not affect the original slot.
	SlotRef - passed by reference. Changes to the value of the matching local argument will affect the value of the original slot.
	Class - always passed by reference. Changes to the value of the matching local argument will affect the value of the original class.
	Object - always passed by reference. Changes to the value of the matching local argument will affect the value of the original object.
Data Type	Value type in the case of a slot: boolean, integer, float, date, time, or string.
Default Value	The value the inference engine will adopt when you don't specify the argument when sending the message. Not valid for lists.
List	Checkmark specifies that the arguments being passed are a list. The list of objects or slots can be passed by reference or not.

When a slot argument is passed by value, the slot named can be either a public or a private slot since the original slot's value is never modified. Private slots passed by value, however, can only be passed from their own Method and the slot name must appear as SELF.propname in the SendMessage argument list. Private slots can never be passed by reference since this would modify the slot's value.

> **Note:** For complete information about the usage of the SendMessage operator and the restrictions of private slots, refer to the Intelligent Rules Element Language Reference manual.

Use the following procedure to establish a user-defined method in the Method editor.

To establish user-defined methods:

1. Open the Method editor and click on the desired mode button. The system automatically highlights the Method field.

2. Type the method name in the edit line and press return when done. The system displays the name in the Method field.

   To view the list of existing method names, choose the Select Method option from the text edit local popup menu.

3. Click on the Attach To field and type the name of the class, object, slot, or property to which the method will be attached in the edit line. Press return when done.

4. Click on the Atom Type menu button and select the type of the structure named in the Attach To field.

   In the case of slots or an existing structure, the system will select the correct type for the structure named in the Attach To field.

5. If you want to pass arguments to the method, click on the Local Arguments Name field and type an underscore (_) and the name of the argument to be passed in the edit line. Press return when done.

   If the argument name you specify designates a list of arguments, select the List field to display the checkmark symbol.

6. To specify how a local argument is passed, display the popup menu for the Nature field and select the desired option.

   Only the Slot option lets you pass an argument that is a copy of the original. All other options (SlotRef, Object, or Class) let you act directly on the argument passed (not allowed in the case of private slots).

7. To specify the data type of a local argument, display the popup menu for the Data Type field and select the desired option.

   If the argument name you specify will actually be a list of arguments, do not select the Data Type field.

8. To specify a default value for a local argument in the event it is not passed, click on the Def Value field and type the value in the edit line. Press return when done.

9. Repeat steps 5 through 8 to specify additional argument templates. Each row of fields in the Local Arguments component defines the template for one argument. Examples of completed local argument definitions appear below.

```
_vector, Slot, Integer, 0_direction, Slot, String, "horizontal"
```
Argument templates are used to identify the individual arguments that you specify along with the SendMessage operator as described in the Intelligent Rules Element Language Reference manual.

10. If you want to create a conditional method, display the popup menu for a blank IF Operator field and select an operator from the list. The system automatically highlights the next field to complete the condition. An example that uses one of the local arguments (_direction) appears below.

```
=       _direction       vertical
```
Test conditions need not appear in the IF component of the Method editor. If absent, the system will automatically execute the actions listed in the THEN fields.

11. Display the popup menu for a blank THEN Operator field and select an action operator from the list. The system automatically highlights the next field to complete the action. An example that uses the other local argument (_vector) appears below.

```
Assign  _vector+SELF.box_origin_y  SELF.box_origin_y
```

12. If the method uses conditions in the IF component and you want to specify "false actions", display the popup menu for a blank ELSE Operator field and select an action operator from the list. The system automatically highlights the next field to complete the action. An example that uses the same local argument (_vector) appears below.

```
Assign vector+SELF.box_origin_x  SELF.box_origin_x
```
ELSE actions are optional for conditional methods. If present, the system will only execute the actions listed in the ELSE fields after all the IF component conditions fail.

13. Complete the remaining Method editor fields and click on the **OK** button to verify and compile the method.

System-Triggered Methods

In general, methods are suspended processes that wait for a particular event to occur. If the event that triggers the method is not a SendMessage operator from a rule or method, then the method is known as a system-triggered method or just "system method." There are two types of system methods

that you can define for a given slot that the system will automatically trigger and execute during inferencing:

Order of Sources
Lets you define and prioritize the sources the system uses to obtain a value for a slot during a session.

If Change
Lets you specify actions that the system initiates whenever the value of the slot changes. Optionally, the If Change method can be triggered when the slot is reset to the value UNKNOWN.

Both the Order of Sources and If Change method are created in the Method editor window (see Figure 2-8). The Method editor popup menus let you choose action operators for the system method named (either "OrderOfSources" or "IfChange"). You select the method operators from the method THEN or ELSE operator field to gain even greater control over inferencing.

■ Order of Sources operators let you specify actions that tell the system how to obtain the slot's value. The order of your actions list is important. The system executes the method THEN or ELSE actions list from top to bottom until it obtains a property value.

■ If Change operators let you specify actions to produce wide ranging effects resulting from the modification of a slot value during inferencing. The order of your actions list is important. The system executes the method THEN or ELSE actions list from top to bottom until it completes the list.

Note: It is also possible to trigger system methods explicitly from a rule or method by using the SendMessage operator. The system will first determine whether the value needs to be determined by the Order of Sources. If the value has not already been supplied, and depending on the current strategy, the system executes the Order of Sources actions list in sequential order until the value of the slot is found and then stops.

Refer to the Intelligent Rules Element Language Reference manual for more information about Order of Sources methods and If Change methods and their operators.

Use the following procedure to establish a system method in the Method editor.

To establish system methods:

1. Open the Method editor and click on the desired mode button. The system automatically highlights the Method field.

2. To specify the system method name, display the local popup menu for the edit line and choose the Select Method option from the list. The system displays a selection dialog.

   Instead of using the selection dialog, you can type the method name in the edit line. The words "OrderOfSources" and "IfChange" must be typed without spaces to correctly specify a system method.

3. Select the system method (*OrderOfSources or *IfChange) from the list that you want to appear in the Method field. Select the OK button to complete the selection.

   The system methods appear in the list with an asterisk (*) in front of the name to distinguish them from user-defined methods.

4. Click on the Attach To field and type the name of the class, object, slot, or property to which the method will be attached in the edit line. Press return when done.

5. Click on the Atom Type menu arrow and select the type of the structure named in the Attach To field.

   In the case of slots or an existing structure, the system will select the correct type for the structure named in the Attach To field.

6. If you want to create a conditional method, display the popup menu for a blank IF Operator field and select an operator from the list. The system automatically highlights the next field to complete the condition.

   Test conditions need not appear in the IF component of the Method editor. If absent, the system will automatically execute the actions listed in the THEN fields.

7. Display the popup menu for a blank THEN Operator field and select an action operator from the list. The system automatically highlights the next field to complete the action.

8. If the method uses conditions in the IF component and you want to specify "false actions", display the popup menu for a blank ELSE Operator field and select an action operator from the list. The system automatically highlights the next field to complete the action.

   ELSE actions are optional for conditional methods. If present, the system will only execute the actions listed in the ELSE fields after all the IF component conditions fail.

9. Complete the remaining Method editor fields and click on the **OK** button to verify and compile the method.

Inheritability

It is important to note that slots will inherit user-defined methods, Order of Sources methods, or If Change methods from a parent slot by default when needed. To override this default behavior you can make individual methods not inheritable by selecting the Private option in the Method editor. By default the Inheritability option is enabled (Public).

Comments and Why

The Comments and Why fields in the bottom portion of the Method editor window let you customize several aspects of the session interface. The Comments field holds your comments about the method triggered. The Why field holds the rationale you provide to explain why the system needs to execute the method. Refer also to Chapter Six, "Application Documentation" for more information about the Comments and Why fields.

3 *Application Editing*

This chapter shows how to start editing the application. It gives instructions for loading and saving knowledge base files. This chapter also provides a limited description of the mechanisms for viewing rule and object structures.

Introduction

In much the same way a text editor knows which file is being edited, the Intelligent Rules Element keeps track of the different knowledge bases in memory. This capability lets you perform more sophisticated editing operations using multiple knowledge bases. This chapter describes these operations. It also gives instructions for working with multiple knowledge bases when you want to view the rule and object structures.

If you have already begun implementing your application as described in Chapter Two, "Application Implementation" then your knowledge base is the default UNTITLED.KB knowledge base. Before you proceed further in this chapter read "Saving Rules and Objects" for details about renaming your knowledge base and saving it to disk.

Setting Up the Editing Environment

This section describes how to use the working memory of the Rules Element shell to control the application editing environment. The operations described in this section become especially important when you want to load several files together. The topics include:

- Loading and clearing knowledge base files
- Designating the "current" file to store new rule and object structures
- Controlling the automatic creation of objects
- Controlling the automatic deletion of related rule and object structures
- Controlling the global numeric data type designation.

Note: The current knowledge base file designation is significant when adding new rules and objects since the system always assigns new structures to the current knowledge base. The Set Knowledge Base command on the Expert menu controls this designation as described below.

Load Knowledge Base

It is possible to load one or more knowledge base files into the working memory of the editing environment. These files must have been previously saved using the Save Knowledge Base dialog window (see Figure 3-6). All knowledge base files that you load will have either a plain text format or a machine-dependent, compiled format. If the naming convention specified for these formats was followed, you can identify file formats by their file name extension. We suggest the .tkb extension for the text format file and the .ckb extension for the compiled format file.

> **Note:** Do not attempt to load a compiled format file on another hardware platform, these files are machine dependent. For details about the text and compiled formats, refer to the "Saving Rule and Object Structures" section later in this chapter.

Of special significance when loading multiple knowledge base files is the way the Rules Element keeps track of rule and object structures. To you the structures will appear as a single file, with rule numbers automatically assigned by the system. This is particularly evident when using the network windows as described in "Viewing Rule and Object Structures" later in this chapter. The system assumes that the knowledge base files you load will work together and renumbers the rules accordingly. If you do not want to view the structures of a particular file, you must clear the file as described in the next section.

Use the following procedure to load one or more knowledge base files.

To load files:

1. Select the Load Knowledge Base option on the Expert menu. The system displays a dialog window that gives you access to system files.

   The Load KB option is also available on the local popup menu that you display in the file list area of the main window.

2. Browse the dialog window directories to locate the desired knowledge base files and click on the file name. The system highlights the file name in the dialog window.

3. Click on the **Load** button in the dialog window. The system recompiles the file and loads it into the working memory of the editing environment.

   Double clicking on the file name has the same effect as selecting the Load button. If the file was originally saved in the compiled format, the system does not recompile and the file loads more quickly.

4. Repeat the procedure to load additional knowledge base files. The system automatically designates the last file loaded as the default current knowledge base.

   To change the current knowledge base file, use the Set Knowledge Base command as described in the following section.

The Non Autocreate option on the Load Knowledge Base dialog window lets you manually verify compilation of the object, class, property, and slot definitions. This feature is generally useful only when you want to structure knowledge base files with rules separated from their object-related definitions. In this way you can precisely control the presence of unwanted object-related structures upon loading. Use the following procedure to load knowledge base files and manually verify compilation.

To verify object compilation with non autocreate:

1. Display the Load Knowledge Base dialog window and select the Non Autocreate option on the window. The system highlights the square to indicate its selected status. The default is unselected.

2. Load the desired file from the dialog window. The system displays a confirmation window that prompts you for the assignment of each object, class, property, and slot in the system.

3. Select the desired option from the confirmation window and click on the **OK** button to complete the compilation of these object structures. When done, the system finishes loading the file into the working memory of the editing environment.

Clear Knowledge Base

After loading knowledge base files into the working memory of the editing environment, you can at any time clear these files and their structures from memory. The Clear Knowledge Base command on the Expert menu or main window local popup menu gives you this option. Use the following procedure to clear the rule and object structures of a particular knowledge base from memory.

To clear a previously loaded file:

1. Select the Clear Knowledge Base option on the Expert menu. The system displays a dialog window that lists all loaded knowledge base files. The current knowledge base appears highlighted.

   The Clear KB option is also available on the local popup menu that you display for an individual file name in the main window.

2. Click on the **Clear** button. The system automatically displays a confirmation dialog window.

3. Alternately, click on the **Clear All** button to clear all files at once.

4. Click on the **OK** button in the confirmation window. The system clears the selected file and designates a new current knowledge base from the ones remaining.

Set Knowledge Base

Especially important to setting up the editing environment is the ability to change the current knowledge base designation. If you have loaded multiple files into memory, the system makes the last file loaded the default current knowledge base. This means the system will automatically store all new rule and object structures that you create into this file. If you do not want this to occur, you must change the current file designation through the Set Knowledge Base command available on the Expert menu or main window local popup menu.

> Note: The Set Knowledge Base command has no affect on modifications to existing structures. Modifications are automatically stored in their original knowledge base files.

Use the following procedure to change the current knowledge base file designation.

To make a file the current file:

1. Select the Set Knowledge Base option on the Expert menu. The system displays a dialog window that lists all loaded knowledge base files. The current knowledge base file appears highlighted.

    The Set KB option is also available on the local popup menu that you display for an individual file name in the main window.

2. Click on the name of the file that you want to store all new rule and object structures. The system highlights the selection.

3. Click on the **OK** button. The system designates the selected file as the current knowledge base file and closes the dialog window.

Set Up Environment Dialog Window

The Set Up Environment option on the File menu displays a dialog window that you can use to control several aspects of the editors. The two pertinent areas of the dialog window are Editor Settings and Default Numeric Data Type. For information about the other dialog window options, refer to

Chapter Five, "Application Testing." The following sections describe the editing options.

> **Note:** To reuse your Set Up Environment settings in later sessions, select the Save Environment option on the File menu. The system updates the `nxrun.dat` file with the new settings.

Automatic Creation

During the implementation stage of application development, you probably let the system automatically create the objects, classes, and properties when you edited rules or meta-slots and assign unique names. This is the system's default mode of operation, but you can control it through the Automatic Creation option in the Set Up Environment dialog window. In the default mode the system prompts you only when it cannot determine by itself the nature (object or class) or the data type (boolean, integer, float, date, or string) of a new object structure.

The default setting is very convenient when you begin implementation. As your application develops, however, your object and class base becomes more stable and you will want to monitor more precisely the creation of object structures. Unselecting the Automatic Creation option guarantees that only desired items will be created. This approach can, for example, help avoid introducing typographical errors during editing.

> **Note:** No name checking is performed when the Automatic Creation option is disabled except on Rule names. Use unique names to identify knowledge base atoms or the system will assume the knowledge base is referencing the same atom.

Use the following procedure to force the system to prompt you for the creation of a new object structure.

To prompt for object data types:

1. Select the Set Up Environment option on the File menu. The system displays the options dialog window.

2. Click on the checkbox under the Editor Settings section that corresponds to the Automatic Creation option. The system unhighlights the square to show its unselected status. The default is selected.

3. Click on the **OK** button from the dialog window. The system records the new settings for the current session.

Automatic Deletion

The Automatic Deletion option on the Set Up Environment dialog window determines whether the system will attempt a cleanup operation when you delete a rule or meta-slot. If you enable the Automatic Deletion option, the system will delete all object-related structures (classes, objects, slots, and properties) that were referenced in the deleted rule or meta-slot and which are no longer referenced elsewhere in the knowledge base. In the default mode you must manually delete these items. Use the following procedure to force the system to clean-up after deleting rules and meta-slots.

To delete object-related structures automatically:

1. Select the Set Up Environment option on the File menu. The system displays the options dialog window.

2. Click on the checkbox under the Editor Settings section that corresponds to the Automatic Deletion option. The system highlights the square to show its selected status. The default is unselected.

3. Click on the **OK** button from the dialog window. The system records the new settings for the current session.

Default Numeric Data Type

The Default Numeric Data Type option on the Set Up Environment dialog window lets you choose the default data type for numeric operations. In the default mode the system always uses the floating point data type (the one available in initial versions of the software). This particular setting keeps the interaction during editing very simple since the system does not prompt continuously. If you want to use additional data types such as integer, date, and time you must select the Prompt User option. Then the Rules Element will prompt you more often, but you will be able to correctly edit expressions using these data types. Use the following procedure to obtain access to the full range of system data types for numeric operations.

To prompt for numeric data types:

1. Select the Set Up Environment option on the File menu. The system displays the options dialog window.

2. Click on the checkbox under the Default Numeric Data Type section that corresponds to the Prompt User option. The system highlights the square to show its selected status.

   This is the only option that lets you decide during editing whether to assign the float, integer, date, or time data type.

3. Click on the **OK** button from the dialog window. The system records the new settings for the current session.

Viewing Rule and Object Structures

The Rules Element helps you to maintain a fully-functioning application during editing by "incrementally compiling" rules and objects. Incremental compiling means each time you click on the editor window's **OK** button, the system automatically adds the new or modified structure to the application. This capability of the Rules Element lets you display and process your application with a minimum of effort. There are several tools that take advantage of this capability to let you immediately visualize the new rule or object structure's relationship to the overall application. The following paragraphs describe a few features of these tools.

> **Note:** This section only introduces the graphic network windows. Refer to Chapter Five, "Application Testing" for advanced operations that allow you to visualize dynamic relationships.

The Rule Network Window

The Rule Network window gives you a "snapshot" view of your growing rule base. This graphic-display window shows your rules in a hierarchical network. The resulting network diagram represents the "reasoning pathways" formed when you created the rule in the Rule editor. Each time you create or modify a rule you can display this window to visualize the rule's relationship to the current knowledge base. Figure 3-1 shows a fully-expanded, rule network diagram in the Rule Network window.

Note: Refer to Appendix C, "Network Icons" for a description of the functions the icons provide in the network window.
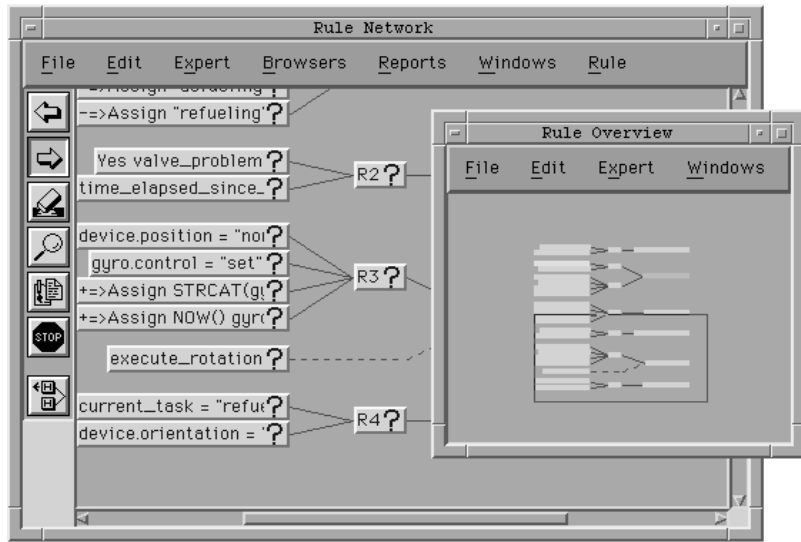

Figure 3-1 Typical Rule Network Diagram: Fully Expanded

The first thing one notices about the network diagram are the lines that connect rules. These lines are the links that form the reasoning pathways between rules. Specifically, the system makes a link when a rule's hypothesis serves as the condition of one or more other rules. This type of link, called a "strong link," is represented by a solid line connecting two or more rules. The network diagram also displays "weak links" or rules whose hypotheses you join through the Context editor. Weak links are represented in the network by dashed lines connecting two or more rules. The majority of links formed in a typical application will be the strong link type. (For more information about strong and weak links, refer to the Intelligent Rules Element Programmer's Guide.)

Rule nodes terminate each connecting line in the network diagram. The textual information displayed in these nodes corresponds with the rules you created in the Rule editor. For instance, each rule node contains the LHS conditions, hypothesis, RHS actions (optional), and name that belong to that particular rule. On the rule RHS, the actions may be either positive evaluation or negative evaluation actions: where the +=> symbol represents

THEN Do actions and the -=> symbol represents THEN Else actions. Figure
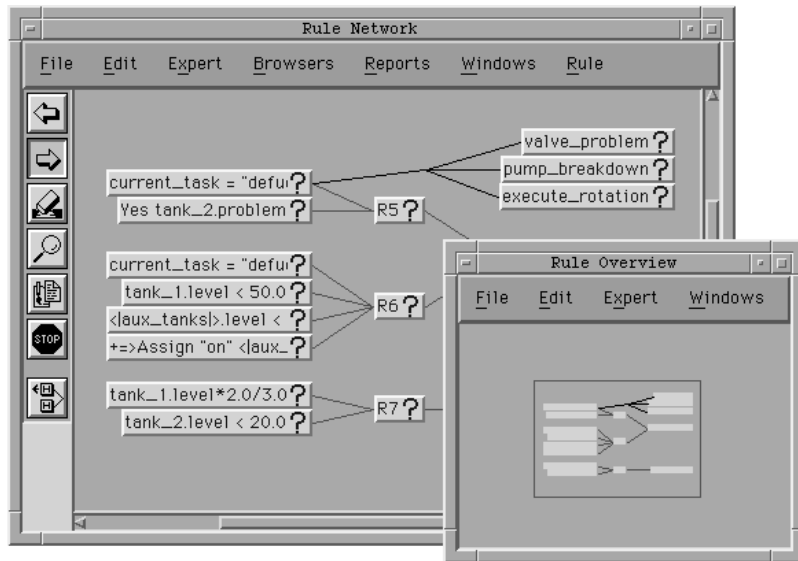3-2 shows these components for two strongly linked rules.

Figure 3-2 Typical Rule Network Diagram: Partially Expanded

Note: You can customize the appearance of the network diagram
using the popup menu Change Settings option as explained in
Appendix D, "Customizing the Environment."

Accessing the Rule Network Window

The windowing environment provides several ways to access the Rule
Network window. The complete set of options are as follows.

- From the Browsers menu on the menu bar
- From the Rule editor popup menu
- From the List of Rules window popup menu
- From the Object Network window popup menu
- From the main window session control panel popup menu
- From the Cross Reference window popup menu.

Initially you can access the Rule Network from the Browsers menu. As you
become more comfortable with the windowing environment, you may
prefer to use the Rule editor, List of Rules window, or Cross Reference
window. These additional approaches provide shortcuts to focus the
network on the currently displayed rule structure.

Full Rule Network Expansion

One approach to visualizing the growing application, is to display the network diagram already fully-expanded. This lets you see the full backward extension from each terminal hypothesis in the application. The more rules you create, the more complex the network diagram will appear. Use this approach when your application is still fairly small, and you can resize the window to show the entire network diagram. Use the following procedure to display the full rule network diagram in the Rule Network window.

To view the fully-expanded, rule network diagram:

1. Select the Rule option on the Browsers menu. The system opens the network window.

2. Click on the Expansion Mode icon to constrain the full display. See Appendix C, "Network Icons" for a description of its function.

3. Display the global popup menu and select the Display All option from the list. A dialog window asks if you want to proceed.

4. Click on the **OK** button. The system displays the rule network diagram with most rules in the current knowledge base.

   Rules in the knowledge base that are not branches of a terminal hypothesis or which contain a Reset operator for their own hypothesis appear at the end of the display.

5. Resize the window or reposition the network diagram if necessary.

   For a description of the overview window and its operation, refer to Chapter One, "The Windowing Environment."

Selective Rule Network Expansion

As your application grows, it will become more difficult to locate a single new rule structure in a fully-expanded, network diagram. The network you display with the Display All option could become a vast tangle of rules, and the overview window may offer little help in separating the rule structures. In this case, you need a different visualization approach. Use the following procedure to expand the network diagram from a single rule in the Rule Network window.

To selectively expand the rule network diagram:

1. Create or display the rule in the Rule editor.

   The rule must be compiled to proceed, click on the **OK** button in the Rule editor to verify and compile the rule.

2. Display the Rule editor global popup menu and select the Focus on Rule option from the list. The system opens the Rule Network window with the previously edited rule displayed.

   You can accomplish the same thing by selecting the Focus on Hypothesis option on the network window global popup menu. A selection window lists the hypotheses currently available.

3. To display the forward chaining links from the currently displayed hypothesis, click on the Right Arrow icon shown in the network window palette. The cursor changes to the shape of a right arrow.

4. Position the cursor over a LHS condition in the group and click. If other rules contain relevant data, the network diagram expands to show the set of hypotheses that share the selected condition's data.

5. Position the cursor over a THEN Do action (represented by the +=> symbol) or a THEN Else action (represented by the -=> symbol) and click. If other rules contain relevant data, the network diagram expands to show the set of hypotheses that share the selected action's data.

   The rule diagram has limitations when trying to show all possible forward chaining links. It does not show links for data contained in methods, nor does it show links for data when it is part of an expression. Refer to the section on the Cross Reference Network.

6. In order to expand the network further, click on the Left Arrow icon (the counterpart to the Right Arrow icon). Move the cursor over the newly displayed set of "related" hypotheses and click. The network diagram expands to reveal the next rule group in the backward chaining link.

   A dashed line between two hypotheses indicates the Context editor was used to establish a weak link.

7. Reposition the network diagram as necessary and continue the procedure by expanding the network diagram using the left and right arrow cursors.

The Object Network Window

The Object Network window gives you a "snapshot" view of your growing object base. This graphic-display window shows your object structures in a hierarchical network. The resulting network diagram represents the "inheritance pathways" formed when you created the object in the editor windows. Each time you create or modify an object, you can display the network window to visualize the object's relationship to the current

knowledge base. Figure 3-3 shows a fully-expanded, object network diagram in the Object Network window.

Note: Refer to Appendix C, "Network Icons" for a description of the network window icon functions.
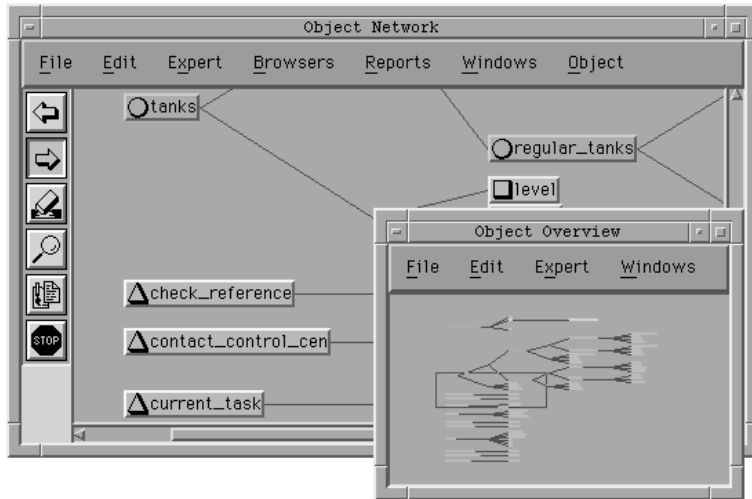

Figure 3-3 Typical Object Network Diagram: Fully Expanded

The first thing one notices about the object network diagram are the lines that connect the object structures. These lines are the links that represent the inheritance pathways between classes, subclasses, objects, and subobjects. Specifically, the system makes a link when you establish a parent/child relationship between two object structures in the editor windows. For instance, the parent/child structures of an object could include classes (or subclasses) and subobjects. Figure 3-4 focuses on this particular relationship

for a single object. The network diagram also shows the properties propagated over these inheritance pathways.
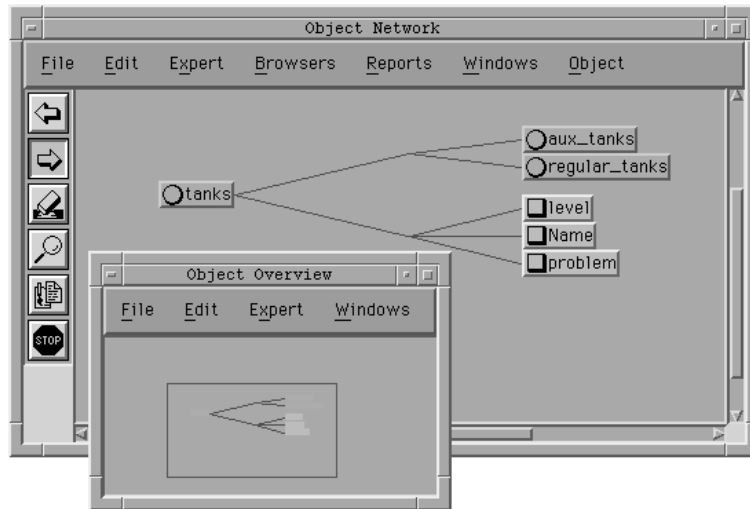

Figure 3-4   Typical Object Network Diagram: Partially Expanded

> **Note:** You can customize the appearance of the network diagram using the popup menu Change Settings option as explained in Appendix D, "Customizing the Environment."

Accessing the Object Network Window

The windowing environment provides several ways to access the Object Network window. The complete set of options are as follows.

- From the Browsers menu on the menu bar
- From the Method editor window popup menu
- From the Object editor window popup menu
- From the List of Objects window popup menu
- From the Rule Network window popup menu
- From the Cross-Reference window popup menu
- From the main window session control panel popup menu.

Initially you can access the Object Network window from the Browsers menu. As you become more comfortable with the windowing environment, you may prefer to use the Object editor, List of Objects window, or Cross Reference window. These additional options provide shortcuts to focus the network window on the currently displayed object structure.

Full Object Network Expansion

One approach to visualizing the growing application, is to display the network diagram already fully-expanded. This lets you see the hierarchy of object structures. The more parent/child structures you create, the more complex the network diagram will appear. Use this approach when your application is still fairly small, and you can resize the window to show the entire network diagram. Use the following procedure to display the fully-expanded, object network diagram in the Object Network window.

To view the fully-expanded, object network diagram:

1. Select the Object option on the Browsers menu. The system opens the network window.

2. Click on the Extension Mode icon to constrain the full display. See Appendix C, "Network Icons" for a description of its function.

3. Display the global popup menu and select the Display All option from the list. A dialog window asks if you want to proceed.

4. Click on the **OK** button. The system displays the object network diagram with every object in the current knowledge base.

5. Resize the window or reposition the network diagram if necessary.

   For a description of the overview window and its operation, refer to Chapter One, "The Windowing Environment."

Selective Object Network Expansion

As your application grows, it will become more difficult to locate a single new object structure in a fully-expanded, network diagram (although a color monitor ease the task a great deal). The network you display with the Display All option could become a vast hierarchy of objects, and the overview window may offer little help in separating the object structures. Unless you have a color monitor, you probably need a different visualization approach.

> **Note:** The Options... command on the Object Network window-specific menu displays a small dialog window that lets you simplify the network diagram by choosing whether or not to display classes, objects, properties, validation functions, and methods when expanding the object network.

Use the following procedure to expand the network diagram from a single object in the Object Network window.

To selectively expand the object network diagram:

1. Create or display the object in the Object editor window.

   The object must be compiled to proceed. Click on the **OK** button in the Rule editor to verify and compile the rule.

2. Display the Object editor global popup menu and select the Focus on Object option from the list. The system opens the Object Network window with the previously edited object and related structures displayed.

   You can accomplish the same thing by selecting the Focus on Object option on the network window global popup menu. A selection window lists the objects currently available.

3. Select the Options... command on the Object menu from the menu bar. The system displays a small dialog window. Click on the All checkbox to include all structures in the network expansion.

4. To identify possible candidates for an inheritance down to the object, click on the Left Arrow icon shown in the network window palette. The cursor changes to the shape of a left arrow.

5. Position the cursor over a property of the object and click. If the property is used in another object or class and could inherit its value down from these sources, the network diagram expands to show the set of slots that include the selected property.

   The related structures are given in the form `object.property` because they are slots of existing objects or classes and not independent properties.

6. Click on the originally displayed object with the left arrow cursor. If the object belongs to another object or class, the network diagram expands to show these inheritance pathways. The objects and classes identified in the previous property expansion will show up in the object expansion.

7. In order to expand the network further, click on the Right Arrow icon (the counterpart to the Left Arrow icon). Position the cursor over the slots displayed from the property and click. The network diagram expands to reveal the methods and meta-slots that may 1) affect inheritance, 2) determine the value of the slot, and 3) be inherited by other object structures when value inheritance fails.

   Meta-slots are identified as darkened squares and methods look like diamonds. Those methods with a P inside the diamond have been defined as not inheritable (private) in the Method editor window.

8. Reposition the network diagram as necessary and continue the procedure by expanding the network diagram using the left and right arrow cursors.

You can continue the investigation from the originally displayed properties or you can initiate a new branch from the left arrow object expansion by using the right arrow cursor.

The Cross Reference

This Cross Reference window provides an excellent way to explore the relationships between the rule and object structures of your application. The Cross Reference window displays these structures in a network format. When you display an item in the Cross Reference window, a right-hand expansion shows those structures that reference the selected item in the currently loaded knowledge base files. Figure 3-5 shows the network diagram resulting from cross referencing two items with the structures in the knowledge base files.
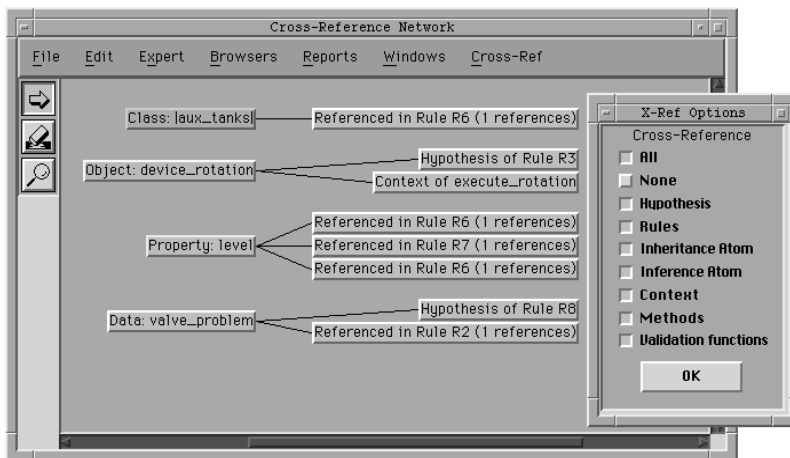
Figure 3-5   Cross Reference Window with Referenced Items

The item name shown in the cross reference network on the left is the item in question. The expansions to the right of the item name identify by cross reference those rule and object structures that share that item. All rule and object structures shown on the right can be said to be related by the shared item on the left. The following sections describe how to use the Cross Reference window to identify related structures in your application.

Viewing Cross References

The Cross Reference window lets you isolate every structure that references a given item in the application. This capability complements the Rule Network or Object Network windows because these windows cannot display relationships between rule and object structures at this level of detail. For example, during a rule network expansion, strong links between data that appear in expressions cannot be visualized. In this case use the Cross Reference window to identify all rules that utilize the data item and thereby locate possible strong links. Table 3-1 lists the items that you can display in the Cross Reference window and their cross reference options:

Item Displayed	Cross References Available
Class	Displays any rules, hypotheses, inference atoms, inheritance atoms, and methods that use the class.
Object	Displays any rules, hypotheses, inference atoms, inheritance atoms, and methods that use the class.
Property	Displays any methods that use the property.
Data	Displays any rules, inference atoms, inheritance atoms, and methods that use the class.
Hypothesis	Displays any rules and context links that use the class. Note: Hypotheses must be selected from the Data or Object selection dialog window.

Table 3-1   Cross Reference Relationships

Note: The Options... command on the Cross Reference window-specific menu displays a small dialog window that lets you determine whether or not to include hypotheses, rules, data, inheritance atoms, inference atoms, context links, and methods among the cross referenced structures.

Use the following procedure to identify cross references for a given item in the Cross Reference window.

To cross reference rule and object structures:

1. Select the Cross Reference option on the Browsers menu. The system opens the Cross Reference window.

2. Select the Options... command on the Cross Reference menu from the menu bar. The system displays a small dialog window. Click on the checkboxes of those structures that you want to cross reference.

3. Display the Cross Reference window local popup menu and select the type of item to display from the list. The system displays a selection dialog window.

    If a hypothesis is desired, select the object item to view a list of all objects including hypotheses.

4. Click on the desired item and click the **OK** button. The system closes the dialog window and displays the item with its cross references in the Cross Reference window.

5. Click on the desired item in the Cross Reference window to visualize the references. Resize the window or reposition the network diagram if necessary.

6. Display the Cross Reference window local popup menu and select the View Line option and the Overview option from the list. The system displays the view line and the overview windows.

Displaying the Rule or Object Network

Once you have isolated a particular item and its cross references in the Cross Reference window, you can view the item and its cross references in the Rule or Object Network window. Use the following procedure to augment rule or object network expansion.

To view cross references in a network window:

1. Select the Cross Reference option on the Browsers menu and display the desired cross references.

2. Display the local popup menu for the desired reference name in the Cross Reference window and select the Focus on Rule Network option. The system opens the Rule Network window with the item and its cross reference hypotheses displayed.

    The referenced item can be viewed in the object network by selecting the Focus on Object Network option from the local popup menu.

3. Use the Rule Network window to identify the full structure of the rule hypothesis, click on the Left Arrow icon shown in the network window palette. The cursor changes to the shape of a left arrow.

4. Position the cursor over the hypothesis name and click. The network diagram expands to show the conditions and actions that include the data item originally displayed in the Cross Reference window.

    Refer also to the Rule Network Expansion section of this chapter for further details.

Modifying Rule and Object Structures

The editor windows that you use to create your rule and object structures are the same ones you use to edit them. You can return to a particular editor window at any point in the application development process to modify some aspect of the current knowledge base. The Rules Element does not require you to first close an active window or to enter a special mode. In fact windows that you use to view rule and object structures make it easy to display the appropriate editor window by providing special buttons and local popup menu Edit options. The following paragraphs explain the options for accessing editor windows with each type of window.

Note: Modifications that you make to a structure take affect on the application immediately, even during runtime. Modifications will not, however, affect a question already displayed in the main window session control panel.

From the Editors

Occasionally, you will want to edit another aspect of the current editor's rule or object structure. However, before you attempt to use another editor window, you must complete the editing operation of the current window. In this way the system keeps track of the related structure by permitting editing of one rule or object structure at a time.

The global popup menu of certain editor windows provides convenient access to related editor windows. Table 3-2 summarizes the relationship between the various editor windows.

Editor Displayed	Global Popup Menu: Editor Options
Rule editor	Context editor
Context editor	none, use Edit menu
Class editor	Object editor, Method editor, Meta-Slot editor
Object editor	Class editor, Method editor, Meta-Slot editor
Property editor	Method editor
Meta-Slot editor	none, use Edit menu
Method editor	none, use Edit menu

Table 3-2 Editor Window Relationships

Additionally, several editor windows provide special buttons that you can select to access a related editor window. When you select one of these

buttons, the system opens the corresponding editor window in view mode. The complete list of these buttons and their functions follows.

- The "Classes" button in the Object editor opens the Class editor.
- The "Properties" button in the Class and Object editors opens the Property editor.
- The "Methods" button in the Class, Object, and Property editors opens the Method editor.
- The arrow button in the third method field of the Class, Object, and Property editors opens the Method editor with the specific class, object, property, or slot displayed (depending on how the method is attached).
- The arrow button in the third property field of the Class and Object editors opens the Meta-Slot editor with the specific slot displayed.

If the editor window does not provide the desired editor option, you can select the desired editor from the Edit menu. In this case you may have to use the window's browsing mechanism to display the desired rule or object structure.

From List Windows

While viewing the various list windows, you may occasionally want to edit the current list window's rule or object structure. The local popup menus of each list window provide this option. You can display the popup menu for specific items in the window and select an editor option that in turn opens the editor window for the selected item. Table 3-3 summarizes the relationship between the list and the editor windows.

List Window	Local Popup Menu: Editor Options
List of Rules	Rule, Context editors
List of Hypotheses	Context editor
List of Data	Meta-Slot editor
List of Object	Object editor
List of Class	Class editor
List of Property	Property editor
List of Methods	Method editor

Table 3-3   List and Editor Window Relationships

The List of Rules window has a close association with the Rule editor. It displays a local popup menu that gives many editing options:

- Create a new rule in the Rule editor

- Edit the current rule in the Rule editor
- Create a copy of the current rule in the Rule editor
- Delete the current rule from the knowledge base.

The current rule is defined as the top page of the List of Rules window. To change the current rule, use the lateral index or page flipping icon as described in the Chapter One, "The Windowing Environment." If no rules exist yet in your current session, the local popup menu command "New" lets you create a new rule in the Rule editor.

From Network Windows

When you start to use the network windows, you may prefer to select the editor windows from their local popup menus. The popup menus that you display for individual items of the network provide shortcuts to modify specific items in the knowledge base. The local popup menu contains the Edit option in the case where a single editor applies to the selected item, otherwise the editor options specify the editor type (for example, "Edit Object" is the option listed for the Object editor). Table 3-4 summarizes the

relationship between the rule and object network windows and the editor windows.

	Selected Network Items	Local Popup Menu: Editor Options
Rule Network	Rule Name	Rule editor
	Hypotheses:	
	terminal hypothesis	Object, Meta-Slot, Context editors
	hypothesis as condition	Object, Meta-Slot editors
	LHS Conditions:	
	class slot	Class, Meta-Slot editor
	data and object slot	Object, Meta-Slot editors
	RHS Actions	none, use the Rule Name local popup menu to display the Rule editor
Object Network	Class, Subclass	Class editor
	Object, Subobject	Object editor
	Property	Class or Object editor dependent on property usage, Meta-Slot editor
	Meta-Slot	Meta-Slot editor
	Method	Method editor

Table 3-4   Rule and Object Network and Editor Window Relationships

> **Note:** Selecting an editing option from the Rule or Object Network window may clear the window. Use the window's global popup menu to redisplay the desired structure(s).

From Cross Reference Window

When you use the Cross Reference window to isolate a particular item and its cross references, you can view each cross reference in its corresponding editor window. Use the following procedure to view the editor in which the item and its cross reference belong.

To edit cross referenced rule or object structures:

1. Select the Cross Reference option on the Browsers menu and display the desired cross references.

2. Display the local popup menu for one of the cross referenced names displayed for the item in the Cross Reference window and select the

Edit... option. The system opens the editor window corresponding to the item and its cross reference.

Saving Rule and Object Structures

When you start the Rules Element and do not load a knowledge base file the system automatically opens a new file in memory called UNTITLED.KB. Then, as you use the Rules Element, the system compiles the rule and object structures you define and records them into this file. When you finish your session and select Quit the Rules Element, the system gives you the option to rename the file before saving it to disk. If you do not want to save the knowledge base in memory, you need only click on the No button in response to the "Save Knowledge Base(s) before Exit?" query.

Most often you will want to save your newly defined rule and object structures to a permanent file. You can either wait until you are ready to exit the Rules Element and let the system prompt you to save the file, or you can select the Save Knowledge Base option from the Expert Menu or local popup menu in the main window. Both approaches display the same dialog window for saving and renaming your files. Figure 3-6 shows the Save Knowledge Base dialog window with the temporary file UNTITLED.KB in memory.
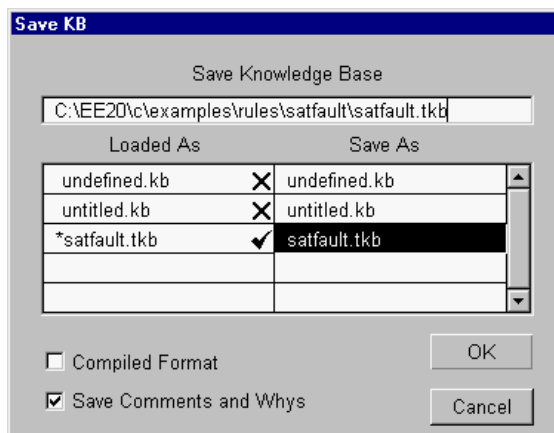
Figure 3-6   Save Knowledge Base Dialog Window

The Save Knowledge Base dialog window lets you perform the following tasks with respect to files currently in memory.
- View names of knowledge base files currently in memory.
- Determine which files have been modified.

- Rename files and specify their directory path.
- Save files to disk in either a text or compiled format.
- Merge two or more files currently in memory.
- Remove all knowledge base comments from files.

Rename and Save

The Loaded As and Save As columns (see Figure 3-6) let you view all knowledge base files currently in memory. Files named in these columns are not limited to files that you specifically loaded (using the Load KB option). These columns also show files loaded by the system during application processing and temporary files created by the system, such as the UNTITLED.KB file previously mentioned. (For more details about temporary files, refer to Chapter Four, "Application Processing.") As the following sections explain, this portion of the Save Knowledge Base dialog window lets you rename and save your knowledge base files.

Note: The system automatically creates a backup copy of the original knowledge base file the next time it is saved. On most machines the symbol "$$$" identifies backup files on disk. On the Mac, the backup file has the extension .BAK appended to the knowledge base file name.

Loaded As

The Loaded As column identifies the file's current name, without its directory path attached. This is known as the logical name. Logical names for knowledge bases can be anything you want to use within the bounds of what is legal for your operating system. The system attaches two symbols to file names that appear in the Loaded As column. An asterisk (*) in front of the file name indicates that someone modified the file since loading. And the checkmark or X to the right of the file name appears to indicate whether or not the file is to be saved. The system automatically places a checkmark once the file is modified during the current session. If you do not want the system to save individual files, you can click on the checkmark to get an X. Conversely, you can click on the X to get a checkmark in order to force the system to save files.

Save As

The Save As column identifies the file's name and location in the system by its directory path. This is known as the physical name. Use this column to rename your file and specify its directory path. The edit line across the top of the dialog window lets you edit any part of the physical name. For instance, you might assign the file name a dot extension of three characters

to distinguish it from other files that would otherwise have the same name. File name extensions are discussed in the "Compiled Format" section. Use the following procedure to modify the physical or logical name and save your knowledge base files.

To rename and save knowledge base files:

1. Select the Save Knowledge Base option on the Expert menu. The system opens the save dialog window with the currently loaded files displayed.

2. Unselect the checkmarks shown next to those files in the Loaded As column that you do not want to save to disk. Only the files that have checkmarks will be saved to disk.

    The system automatically places a checkmark next to the file names of files that were modified since loading. However, you may not always want to save the changes. The asterisk symbol remains displayed regardless of the checkmark's presence or absence.

3. Click on the desired file name displayed in the Save As column. The system displays the file name and full directory path in the edit line and highlights the file name displayed in the Save As column.

4. Use the edit line across the top of the dialog window to modify the file name or its directory path as desired.

    Remember to use file name extensions to help identify the format of files that share the same name. The extensions `.tkb` and `.ckb` are recommended for text and compiled format respectively.

5. Click on any file name in the Save As column to transfer the edited name back. The system unhighlights the changed file name in the Save As column and automatically places a checkmark in the Loaded As column.

    The small checkmark may have already been displayed if the file was modified since loading.

6. Click on the **OK** button in the dialog window. The system saves all files to disk that have a checkmark in the Loaded As column and closes the dialog window.

    The system does not overwrite existing files of the same name. Previous file versions are identified by the symbol $$$ prefixed to the file name.

Another operation that you can perform with the Rename and Save columns is merging two or more knowledge base files together. This procedure requires that the files exist together permanently. If you are not sure, make backup copies since this procedure cannot be reversed. Use the following

procedure to merge the rules and objects of two or more knowledge base files into a single file.

To merge knowledge base files:

1. Select the Save Knowledge Base option on the Expert menu. The system opens the save dialog window with the currently loaded files displayed.

2. Ensure that each file that you want to merge has a checkmark displayed in the Loaded As column. The default is unselected, if the files were not modified during the session.

3. Decide on a single name for the combined files' and use the edit line to rename the files. Transfer the new name back to the Save As column for each file you want to merge.

4. Click on the **OK** button in the dialog window. The system saves all files to disk that have a checkmark in the Loaded As column and closes the dialog window.

Compiled Format

The Save Knowledge Base dialog window (see Figure 3-6) gives you a choice of file formats for knowledge bases. The Compiled Format option lets you easily select this special format instead of the regular text format. The default mode of saving files uses the text format.

If the Compiled Format option is unselected, the system automatically saves knowledge base files in text format. The text format is readable and can therefore be edited with any standard text editor. Additionally, the text format is the only format which lets you transfer files across different hardware platforms. You should always keep a copy of your knowledge base files in the text format; they will remain fully upward compatible with future releases of the Rules Element.

If the Compiled Format option is selected, the system saves knowledge base files in this special format. The compiled file format is unreadable and is stored in binary mode. The advantage of using the compiled format is that it saves time during loading since the system does not have to recompile the rules and objects first. This format is strictly platform dependent. If you plan to transfer files across different hardware platforms, use the text file format for delivery.

To identify the same knowledge base files stored in both formats, we suggest giving each file a three letter file name extension: `.TKB` for text format and `.CKB` for compiled format. For example, `MYFILE.TKB` and `MYFILE.CKB` are the same knowledge bases that you specifically saved in different

formats. These extensions are not automatically assigned by the system. You must use the renaming procedure to add the appropriate extension when saving files.

Use the following procedure to rename and save your knowledge base files in compiled format.

To compile and save knowledge base files:

1. Select the Save Knowledge Base option on the Expert menu. The system opens the save dialog window with the currently loaded files displayed.

2. Ensure each file that you want to save has a checkmark displayed in the Loaded As column. The default is an X, if the files were not modified during the session.

3. If necessary, use the edit line to rename the files and transfer the new name back to the Save As column.

    Remember to use file name extensions to help identify the format of files that share the same name. The extensions .tkb and .ckb are recommended for text and compiled format respectively.

4. Select the Compiled Format option from the dialog window. The system places an X in the square to indicate its selected status. The default is unselected.

    Compiled format is platform dependent and cannot not be transferred across hardware platforms.

5. Click on the **OK** button in the dialog window. The system saves all files to disk that have a checkmark in the Loaded As column and closes the dialog window.

    The next time you save the compiled format file, remember to select the Compiled Format option on the dialog window.

Save Comments and Why

The Save Knowledge Base dialog window (see Figure 3-6) lets you easily delete all comments from your knowledge base files by unselecting the Save Comments and Why option. In most cases you will want to save comments and why explanations with your knowledge base files, therefore the Save Comments and Why option's default is selected. Upon delivering the application, you can remove comments if desired.

☞   If you unselect the Save Comments and Why option in the dialog window, and exit the Rules Element or clear the knowledge base file, all of your comments will be lost. Make certain that this option's usage is intentional.

Use the following procedure to save your knowledge base files without comments and why text.

To delete comments and why text:

1. Select the Save Knowledge Base option on the Expert menu. The system opens the save dialog window with the currently loaded files displayed.

2. Ensure each file that you want to save has a checkmark displayed in the Loaded As column. The default is an X, if the files were not modified during the session.

3. If necessary, use the edit line to rename the files and transfer the new name back to the Save As column.

4. Unselect the Save Comments and Why option from the dialog window. The system unhighlights the square to indicate its unselected status. The default is selected.

5. Click on the **OK** button in the dialog window. The system saves all files to disk that have a checkmark in the Loaded As column and closes the dialog window.

   If you need to recover comments that were unintentionally lost, get the backup copy of the file created by the system. On most machines the symbol "$$$" identifies backup files on disk. On the Mac, the backup file has the extension .BAK appended to the knowledge base file name.

Changing File Ownership of Rule and Object Structures

If you are creating the knowledge base for the first time, the system automatically opens a new file in memory called UNTITLED.KB. This file stores all new rule and object structures until you rename and save it to disk. As your application develops, you might want to create several additional knowledge base files. For example, many developers choose to place objects and rules in separate but compatible files. Eventually, you could even load and manipulate several knowledge base files at once. To facilitate these tasks the system loads each files' rule and object structures into working memory (for editing), but still keeps track of which structures belong to which files. The system also provides commands that let you move structures between loaded files.

To better understand how you can manipulate rule and object structures, you need to know how the system stores them in the knowledge base file. Assume you just created the rule R1: "If Yes A.Prop then H," where A.Prop has an Order of Sources method defined as well as a meta-slot for data validation, and H has a context link. The system stores the rule R1, the

objects A and H, the property Prop, and the slots A.Prop and H.Value in the current knowledge base file as follows.

R1:   with its contents of LHS, RHS, and hypothesis name
A: the fact that it has a Prop
H: the fact that it has the property Value with type boolean
Prop: the fact that it is of type boolean
Value: special property type for hypotheses only
A.Prop: the fact that it has a method (OrderOfSources)
A.Prop: the fact that it has a meta-slot (data validation)
H.Value: the fact that it has a context link.

Once you save the application, each one of these structures exists independently in the file. You can load the file to change and manipulate them through their corresponding editor windows as desired. The following sections describe these operations and their commands.

Change KB

Working with several knowledge base files loaded into memory is possible because the system keeps track of the files' rule and object structures. To help you identify the knowledge base to which the structure belongs, each editor displays the corresponding knowledge base file name. You can select the Change KB command from the window's global popup menu to change ownership of the rule or object structure to another knowledge base file listed in the dialog window.

The Change KB command completely changes ownership of the displayed item. Structures that you select will not appear in the original file unless you first duplicate the item in the current editor window. Use the following procedure to change the knowledge base of the rule or object structure currently displayed in the editor window.

To change knowledge base ownership of existing items:

1. Display the desired rule or object structure in its corresponding editor window and select the Edit button.

2. Display the editor window global popup menu and select the Change KB option from the list. The system displays a dialog window with the displayed rule or object structure's current file highlighted.

   Knowledge base files that were modified since loading show an asterisk (*) to the left of their file name.

3. Click on the name of the file in which you want to store the existing rule or object structure. The system highlights the selection.

4. Alternately, click on the **New** button to create a new knowledge base file and type the file name.

    Do not type the name of a file that already exists or the system will overwrite the old file with the new file. If this occurs, get the backup copy of the file created by the system.

5. Click on the **OK** button. The system changes ownership of the displayed rule or object structure to the selected file and closes the dialog window.

    The structure's new knowledge base file appears in the KB Name field of the editor window.

Set Knowledge Base

The Set Knowledge Base command on the Expert menu or the Set Current KB command on the main window local popup menu lets you designate any loaded knowledge base as the current one. In the main window, the current knowledge base designation is indicated by an arrow next to the knowledge base file name and is by default the last file loaded. This designation is important when dealing with multiple files because it lets the system decide where to store all *new* rule and object structures you create. Essentially, changing the current knowledge base file designation lets you change ownership of all new structures to the file of your choice.

> Note: The Set Knowledge Base command has no affect on modifications to existing structures. Modifications are automatically stored in their original knowledge base files.

Use the following procedure to change the current knowledge base file designation in order to add *new* rule and object structures through the editor windows.

To change knowledge base ownership of new items:

1. Select the Set Knowledge Base option on the Expert menu. The system displays a dialog window that lists all loaded knowledge base files. The current knowledge base file appears highlighted.

    Knowledge base files that were modified since loading appear in italic in the main window.

2. Click on the name of the file in which you want to store all new rule or object structures. The system highlights the selection.

3. Alternately, click on the **New** button to create a new knowledge base file and type the file name.

    Do not type the name of a file that already exists or the system will

overwrite the old file with the new file. If this occurs, get the backup copy of the file created by the system.

4. Click on the **OK** button. The system closes the dialog window and places an arrow next to the selected file.

5. Display the desired editor windows and create the new structures for the newly designated current knowledge base.

If you change the current knowledge base designation, you still have access to the rule and object structures of all loaded files through the editor windows. This capability of the system lets you easily duplicate structures from one knowledge base into another. In most cases a slight modification of the structure is required (the only exception is for rules). The system does not otherwise allow two copies of the same structure to exist.

Use the following procedure to quickly copy structures from any number of knowledge bases to the one you designate as the current knowledge base.

To change knowledge base ownership of duplicate items:

1. Select the Set Knowledge Base option on the Expert menu and specify the file in which you want to store the duplicate structures. The system designates the selected file as the current knowledge base file.

2. Display the desired rule or object structure in its corresponding editor window and click on the **Copy** button in the editor window. The system displays the duplicate structure.

3. Modify the structure to make it unique within the currently loaded knowledge base files.

    The system does not allow two exact copies of the same structure to coexist unless it is a rule.

4. Click on the **OK** button in the editor window. The system automatically stores the new structure in the current knowledge base file.

5. Repeat the copy procedure for all other structures. Each time the system automatically stores the duplicate in the knowledge base file you previously specified in the Set Knowledge Base dialog window.

4 *Application Processing*

Application developers should read this chapter to run their applications using the Intelligent Rules Element shell.

Introduction

This chapter gives useful information about running the Rules Element applications. It describes the function of a variety of windows you use to start and interact with the running application. The windows that comprise the processing environment include the following.

- Suggest | Volunteer | SendMessage dialog window
- List of Hypotheses and List of Data windows
- Session Control Panel (displayed in the main window).

Although the Rules Element shell may not be the final runtime environment, the concepts expressed in this chapter are still useful for understanding the requirements of running the Rules Element applications. The Rules Element shell simply brings together your application and the processing algorithms supplied by the inference engine in a single environment. The principles of starting the process to dynamically direct or control the system, remain the same for all runtime environments. This activity of conducting an application processing session is sometimes called "inference engine processing" or just processing.

Note: Refer to the Open Interface User's Guide for specific information about running applications in other environments.

You can initiate processing at virtually any stage of the application development process. The system requires no minimum number of rules to conduct an application processing session in the Rules Element environment.

Setting Up the Processing Environment

This section describes system requirements for initiating processing. The topics include.

- Loading knowledge base files

■ Setting the search path.

Load Knowledge Base

If you are not currently editing a knowledge base that you want to initiate processing on, you must first load the desired knowledge base into the system's working memory. Processing can take place only on currently loaded knowledge base files.

Your application may include one or more knowledge base files depending on how you have structured the application. If multiple files are involved it may be necessary to load them all before initiating processing. The number of files you can load at one time is primarily limited by the system's memory capacity. However, knowledge base files are typically small and memory should not be a factor.

You can use the Load Knowledge Base option on the Expert menu or main window local popup menu to load the file previously stored in either the text or the compiled format. The text format file uses the `.tkb` extension and the compiled format file uses the `.ckb` extension. Files in the compiled format load much faster since the system does not have to recompile the rules and objects first. Use the following procedure to load your knowledge base files prior to processing.

To load knowledge base files:

1. Select the Load Knowledge Base option on the Expert menu. The system opens the load knowledge base dialog window.

2. Change the current path if necessary and click on the desired knowledge base file to load. The system highlights the filename.

3. Ensure the Non Auto Create checkbox has an unselected status. The default compiles the rules and automatically creates knowledge base objects.

   Refer to Chapter Three, "Application Editing" for information about using the Non Auto Create checkbox.

4. Click on the **Load** button from the dialog window. The system loads the file and closes the loading window.

   Double-clicking rapidly on the filename loads the file directly.

Set Search Paths

If your application utilizes files other than knowledge base files, you must identify where the Rules Element can locate them in your system's directory.

For example, files of this type could include databases files and Apropos files. One of the following paragraphs identifies the way to set the search path for your hardware platform.

VAX/UNIX

If you are running the Rules Element on a VAX or UNIX platform, you must customize the logical name or environment variable by specifying the directory name or list of directories to search. Only then will the Rules Element be able to locate files such as the `nxrun.dat` file or other knowledge base files used by your applications.

Under VMS the command to specify a directory or list of directories is:

```
$ define nd$path dirlist
```

Under the UNIX Bourne shell the command is:

```
$ NXP_PATH=dirlist; export NXP_PATH
```

Under the UNIX C shell the command is:

```
% setenv NXP_PATH dirlist
```

Macintosh

If you are running the Rules Element on a Macintosh, you must edit the string resource for directory path names.

Starting the Session

The Rules Element inference engine is flexible enough to investigate the consequences of situations and/or facts that you represent in the knowledge-based application as rules and objects. To initiate processing or "knowcess" on these structures you need only specify a starting point within the application. Table 4-1 elaborates on the possible processing modes that your starting point can initiate.

Mode	Description
`Suggest Hypotheses`	The term "suggest" refers to the placement of hypotheses on the system's agenda. The supplied hypotheses represent the user's best guess about the type of problem. The system uses the hypothesis as a goal that it proves or disproves in a backward chaining fashion by evaluating the evidence necessary to access the initial hypothesis. The system automatically expands the search by investigating other potential solutions.

`Volunteer Data`	The term "volunteer" refers to the input of data values into the system. While the system starts with all data values in the UNKNOWN state, you can change any of the data by supplying actual values when known. The data supplied represents known symptoms, changes of values, or other incidental information about the problem. The system uses the data as evidence to prove or disprove hypotheses in a forward chaining fashion. The system automatically propagates the initial facts to related hypotheses in order to investigate their full consequence.
`Mixed-Mode`	This approach combines the suggest and volunteer approaches to start processing. Use this approach when correlations between situations and facts are known in advance. The system uses forward chaining and backward chaining together. It starts by putting the hypotheses of those rules that contain the data on the agenda for evaluation and then begins backward chaining.
`Dynamic Binding of Methods`	This term "dynamic binding" refers to the triggering of a method at runtime for a specific object name and arguments. Use this approach when you want to emphasize the object domain of your application. Methods that you trigger take precedence over forward chaining and backward chaining that you may initiate. The system starts by putting the methods that you select in the current evaluation stack of the agenda for evaluation and triggers them one by one.

Table 4-1   Inference Engine Processing Modes

Note: For a complete description of the Rules Element inferencing behavior, refer to the Intelligent Rules Element Language Programmer's Guide.

The following sections describe these ways to initiate processing using various windows in the Rules Element environment.

Suggest | Volunteer Dialog Window

The Suggest | Volunteer... option on the Expert menu displays a dialog window that you can use to initiate processing. The Suggest | Volunteer | SendMsg dialog window (see Figure 4-1) offers the most direct way to initiate processing in the Rules Element environment. The window gives you easy access to all object structures that can serve as a starting point. As its name implies, you can initiate the following possible processing modes.

- Suggest hypotheses
- Volunteer data
- Mixed-mode: volunteer data and suggest hypotheses
- Send messages to trigger methods.

Figure 4-1 shows the Suggest | Volunteer | SendMsg dialog window as it would appear with a knowledge base loaded. The figure shows the window's default mode: the Hypotheses button selected with the hypotheses listed in the left side column and no starting point yet specified.
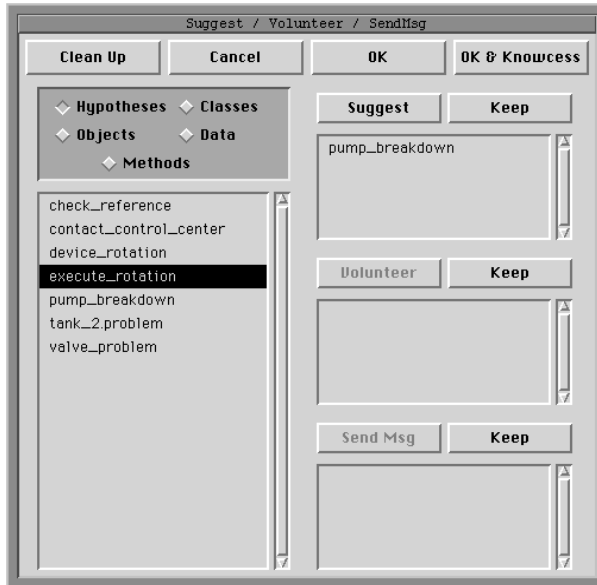
Figure 4-1 Suggest | Volunteer | Send Message Dialog Window

To initiate processing with this window, the desired rule and object structure(s) must appear in one of the right side columns. You select these structures from the ones listed in the left side column. The five function buttons (Hypotheses, Classes, Objects, Data, and Methods) grouped directly above the left side column determine the list of selectable structures. Table

4-1 shows the other function buttons that determine how the system uses structures you select.

Button	Description
Ok & Knowcess	This button lets you begin processing using the currently selected items.
Ok	This button lets you close the window and suggest or volunteer selections without initiating processing. The system retains the Suggest	Volunteer	Send Message status until you start the session.
Cancel	This button lets you close the window without saving the status of the current Suggest	Volunteer	Send Message selections.
Clean Up	This button lets you clear the right side column selections and begin over.
Suggest	This button lets you select a hypothesis for display in the right side column for inferencing.
Volunteer	This button lets you select public data, objects, or classes for display in the right side column for inferencing.
Send Message	This button lets you select methods for display in the right side column for processing.
Keep	This button lets you reuse the same Suggest	Volunteer	Send Message selections in future application processing sessions.

Table 4-1 Suggest | Volunteer | Send Message Dialog Window Buttons

The following sections describe the task of initiating processing using the Suggest | Volunteer | Send Message dialog window.

Suggest Hypotheses

When you open the Suggest | Volunteer | SendMsg dialog window the system displays the window by default in the suggest hypotheses mode. The left side column automatically holds the list of hypotheses from the currently loaded knowledge base files. You can specify one or more hypotheses from the list as starting points for the inference engine. Hypotheses that you select appear in the right side suggest column.

Use the following procedure to select hypotheses and initiate processing from the Suggest | Volunteer dialog window.

To suggest hypotheses:

1. Select the Suggest | Volunteer... option on the Expert menu. The system opens the Suggest | Volunteer dialog window with all hypotheses displayed in the left side column.

   If another list appears in the left side column, click on the **Hypotheses** button to redisplay hypotheses.

2. Scroll the list to display the desired hypothesis and click on the hypothesis name in the list. The system highlights the hypothesis name.

3. Click on the **Suggest** button over the right side column. The system displays the hypothesis in the right side suggest column. You can place as many hypotheses in the right side suggest column as desired.

   Double-clicking rapidly on the hypothesis name displays the hypothesis in the right side suggest column directly.

4. If you want to initiate processing immediately, click on the **OK & Knowcess** button. The system initiates application processing using the suggested hypotheses.

5. If you prefer not to initiate processing yet want to save the current selections, click on the **OK** button. The system closes the dialog window.

   To close the Suggest | Volunteer dialog window and initiate processing, select Start With... Knowledge Base on the Expert menu.

At the end of a session, the Restart Session option on the Expert menu normally resets the status of all suggested hypotheses to their original UNSELECTED state. Also, the system automatically clears the Suggest | Volunteer dialog window's suggested hypotheses each time you close and reopen the window. It is possible, however, to give right side column selections a permanently selected status. Use the following procedure to reuse the currently selected hypotheses in other application processing sessions.

To reuse the same hypotheses:

1. Open the Suggest | Volunteer dialog window and place hypotheses that you know you will reuse on the right side suggest column.

2. Click on the **Keep** button above the right side suggest column. The system gives these hypotheses a permanently selected status.

   Additional hypotheses you add to the list after clicking the **Keep** button will not have a permanently selected status.

3. Click on the **OK** button from the dialog window. The system closes the dialog window without initiating processing.

4. Open the List of Hypotheses window and browse the list to display the hypotheses previously selected in the dialog window. The system highlights the previously selected hypotheses.

   This operation is identical to using the local popup menu in the List of Hypotheses window to select hypotheses.

5. Optionally, select the Save Knowledge Base option on the Expert menu to reuse the same hypotheses even after exiting the Rules Element.

   If you do not save the knowledge base before exiting the Rules Element, the system restores any previously saved hypotheses the next time you load the knowledge base.

If you no longer want hypotheses to appear permanently on the right side columns, you can unselect individual hypotheses. Use the following procedure to unselect permanently selected hypotheses from the right side suggest column.

To unselect reused hypotheses:

1. Open the Suggest | Volunteer dialog window to view the list of hypotheses with a permanently selected status.

   Closing and reopening the dialog window clears hypotheses from the list that were not permanently selected.

2. If you want to unselect specific hypotheses, double click on the hypothesis name in the right side suggest column that you want to clear.

   If you want to unselect all hypotheses, click on the **Clean Up** button from the dialog window. The system clears the right side Suggest, Send Message *and* Volunteer columns.

3. Click on the **Keep** button above the right side suggest column. The system "keeps" remaining hypotheses if any.

4. Click on the **OK** button from the dialog window. The system closes the dialog window without initiating processing.

5. Open the List of Hypotheses window and browse the list to display the hypotheses previously selected in the dialog window. The system displays the previously unselected hypotheses normally (not highlighted).

   This operation is identical to using the local popup menu in the List of Hypotheses window to unselect hypotheses.

6. Optionally, select the Save Knowledge Base option on the Expert menu to retain the right side column's current status even after exiting the Rules Element.

    If you do not save the knowledge base before exiting the Rules Element, the system restores the previously saved hypotheses the next time you load the knowledge base.

Volunteer Data

The left side column displays three function buttons that you can use to select object structures to initiate data-driven processing. Table 4-2 shows these function buttons.

Button	Description
Data	This button lets you display any hypotheses and objects that appear in rule conditions in the left side column. Private data (from private slots) is excluded from this list since it cannot be volunteered.
Classes	This button lets you display all structures specifically defined as classes in the left side column.
Objects	This button lets you display all structures that are either objects or hypotheses in the left side column.

Table 4-2   Left Side Column Volunteer Buttons

The Suggest | Volunteer dialog window offers these three buttons because you may want to focus on a specific type of data. Data can be simple data (the generic property Value) or more specific slots of any object or class. Additionally, data can include subgoal hypotheses for which you can specify a value of TRUE, FALSE, or NOTKNOWN.

Use the following procedure to select data and initiate processing from the Suggest | Volunteer dialog window.

To volunteer data:

1. Select the Suggest | Volunteer... option on the Expert menu. The system opens the Suggest | Volunteer dialog window with all hypotheses displayed in the left side column.

2. Click on the **Data**, **Classes**, or **Objects** button to display the desired list.

3. Scroll the list to display the desired data and click on the data name in the list. The system highlights the data name.

4. Click on the **Volunteer** button over the right side column. The system displays the data in the right side volunteer column. You can place as much data in the right side volunteer column as desired.

   Double-clicking rapidly on the data name displays the data in the right side volunteer column directly.

5. If you want to initiate processing immediately, click on the **OK & Knowcess** button. The system displays a selection dialog window.

   If the data belongs to an object or class, the system also displays a dialog window to obtain the correct property.

6. If you prefer not to initiate processing yet want to save the current selections, click on the **OK** button. The system displays a selection dialog window.

   If the data belongs to an object or class, the system also displays a dialog window to obtain the correct property.

7. Click on the desired value shown in the dialog window and click on the **OK** button. The system volunteers the data you supply.

   If the selected data is a subgoal hypothesis (one tested in a rule condition), the system lets you volunteer the values TRUE, FALSE, or NOTKNOWN.

8. Repeat the selection procedure for each selection dialog window the system displays. The system either initiates application processing using the volunteered data, or the system merely closes the Suggest | Volunteer dialog window. The action taken depends on which button invoked the selection dialog window.

   To initiate processing after closing the Suggest | Volunteer dialog window, select Start With... Knowledge Base on the Expert menu.

At the end of a session, the Restart Session option on the Expert menu normally resets the status of all volunteered data to their original UNSELECTED state. Also, the system automatically clears the Suggest | Volunteer dialog window's volunteered data each time you close and reopen the window. It is possible, however, to give right side column selections a permanently selected status. Use the following procedure to reuse the currently selected data in other application processing sessions.

> **Note:** The Restart Session option resets the value of the data to UNKNOWN, you must therefore supply new values.

To reuse the same data:

1. Open the Suggest | Volunteer dialog window and place data that you know you will reuse on the right side volunteer column.

2. Click on the **Keep** button above the right side volunteer column. The system gives these data a permanently selected status.

   Additional data you add to the list after clicking the **Keep** button will not have a permanently selected status. Click on the **Keep** button each time to retain new data.

3. Click on the **OK** button from the dialog window and supply the desired data in the selection dialog window. The system closes the Suggest | Volunteer dialog window without initiating processing.

4. Open the List of Data window and browse the list to display the data previously selected in the dialog window. The system displays the selected data highlighted.

   This operation is identical to using the local popup menu in the List of Data window to select data.

5. Optionally, select the Save Knowledge Base option on the Expert menu to reuse the same data even after exiting the Rules Element.

   If you do not save the knowledge base before exiting the Rules Element, the system restores any previously saved data the next time you load the knowledge base.

If you no longer want data to appear permanently on the right side columns, you can unselect individual data. Use the following procedure to unselect permanently selected data from the right side volunteer column.

To unselect reused data:

1. Open the Suggest | Volunteer dialog window to view the list of data with a permanently selected status.

   Closing and reopening the dialog window clears data from the list that was not permanently selected.

2. If you want to unselect specific data, double click on the data name in the right side volunteer column that you want to clear.

   If you want to unselect all data, click on the **Clean Up** button from the dialog window. The system clears the right side Volunteer, Suggest *and* Send Message columns.

3. Click on the **Keep** button above the right side volunteer column. The system "keeps" remaining selections if any.

4. Click on the **OK** button from the dialog window and supply the desired data in the selection dialog window. The system closes the Suggest | Volunteer dialog window without initiating processing.

5. Open the List of Data window and browse the list to display the data previously unselected in the dialog window. The system displays the unselected data normally (not highlighted).

   This operation is identical to using the local popup menu in the List of Data window to unselect data.

6. Optionally, select the Save Knowledge Base option on the Expert menu to retain the right side column's current status even after exiting the Rules Element.

   If you do not save the knowledge base before exiting the Rules Element, the system restores the previously saved data the next time you load the knowledge base.

Send Message

The left side column of the Suggest | Volunteer | SendMsg dialog window displays the Methods function button that you can use to select methods to trigger at runtime. The dialog window offers the Methods button because you may want to begin processing your application entirely within the object domain. If you want your application to pass messages between objects and use the Methods button to initiate this type of processing, you can eliminate the rule that you might otherwise need to execute the first SendMessage operator.

> **Note:** If your application relies on a combination of rules and objects, you can still use the dialog window to volunteer data and suggest hypotheses, however, methods that you select are given priority by the inference engine and are executed before other processing is initiated.

Methods that you select from the Suggest | Volunteer | SendMsg dialog window appear in the right side Send Message column. The order that the methods appear in the Send Message column is important since the system begins processing with the first method in the list and works its way to the bottom until each method has been triggered.

Once you are satisfied with the method selections and you select the OK or OK & Knowcess button from the Suggest | Volunteer | SendMsg dialog window, the system prompts you for information it will need to initiate processing with the message dialog shown in Figure 4-2. Triggering a method at runtime, also called message passing, requires that you specify

the name of the object structure to trigger the method for and optionally provide arguments to pass to the method.

Note: Currently, the arguments you specify for methods that you select from the Suggest | Volunteer | SendMsg dialog window have a few limitations. The arguments cannot accept lists of any kind (either slots separated by commas or slots specified by the pattern matching syntax), and the arguments cannot name a slot unless it has an initial value specified for processing. If a slot name is specified as an argument, but no initial value appears in the Meta-Slot editor, the system will use the default value specified in the Method editor arguments template.
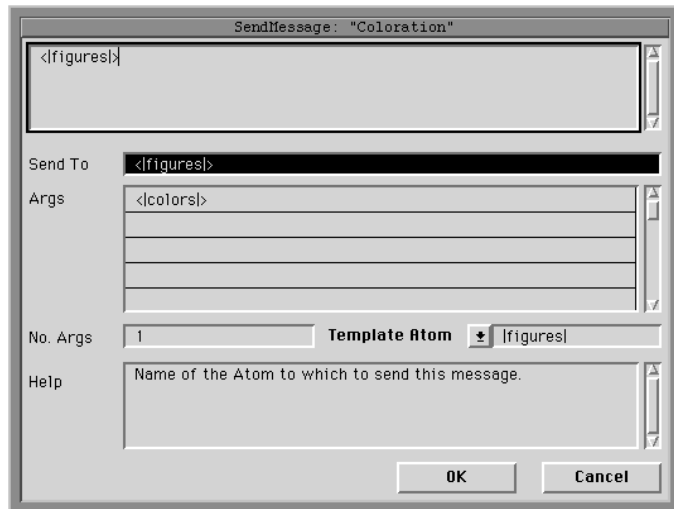
Figure 4-2   SendMessage Dialog Window

Use the following procedure to select methods that you will trigger with a corresponding message name and initiate processing from the Suggest | Volunteer | SendMsg dialog window.

To send messages:

1. Select the Suggest | Volunteer... option on the Expert menu. The system opens the Suggest | Volunteer | SendMsg dialog window with all hypotheses displayed in the left side column.

2. Click on the **Methods** button to display the list of methods that appear in the currently loaded knowledge base files.

3. Scroll the list to display the desired method and click on the method name in the list. The system highlights the method name.

4. Click on the **Send Msg** button over the right side column. The system displays the method in the right side Send Message column. You can place as many methods in the right side column as desired.

   Double-clicking rapidly on the method name displays the method in the right side Send Message column directly.

5. If you want to initiate processing immediately, click on the **OK & Knowcess** button. The system displays the SendMessage dialog window for the first method in the right side column.

6. If you prefer not to initiate processing yet want to save the current selections, click on the **OK** button. The system displays the SendMessage dialog window.

7. Click on the **Send To** field and type the names of the addressees to receive the message in the text edit field and press the Return key. Each addressee must be separated by a comma.

   An addressee list cannot be specified by pattern matching to initiate processing, although pattern matching is valid from a SendMessage operator that appears in a rule or method. See the Intelligent Rules Element Language Reference manual for information about the SendMessage operator.

8. Click on the **Template Atom** button and view the list of object structures that have the method selection attached. The menu can display class, object, slot, and property names. Select the object structure from the list that you want to use as a prototype for the argument definitions. The system displays the selected atom in the menu button.

   It is assumed that all addressees in your Send To list have the same argument definitions. Sending messages to addressees with arguments of different types requires separate SendMessage definitions.

9. To specify the first argument to pass to the method, click in the first row of the **Arguments** list box. The system displays the argument parameters from the Method editor's argument template in the Help

box. Type the argument in the text edit and press the Return key. Arguments can be a value you supply or a slot name.

If a slot name is specified as an argument, but no initial value appears in the Meta-Slot editor, the system will use the default value specified in the Method editor arguments template.

10. Repeat step 9 for each additional row of the Arguments list box (each row matches one argument). The Help box displays the argument parameters again to remind you of its data type. Click on the **OK** button from the SendMessage dialog window when you are satisfied with the message passing information. The system closes the dialog window and displays another arguments dialog window for the next method in the Send Message list.

11. Repeat steps 7 through 10 for each SendMessage dialog window the system displays. The system either initiates application processing using the methods to trigger, or the system merely closes the Suggest | Volunteer | SendMsg dialog window. The action taken depends on which button invoked the selection dialog window.

To initiate processing after closing the Suggest | Volunteer | SendMsg dialog window, select the Start With... Knowledge Base option on the Expert menu.

The system automatically clears the Suggest | Volunteer | SendMsg dialog window's selected methods each time you close and reopen the window. It is possible, however, to give right side column selections a permanently selected status. Use the following procedure to reuse the currently selected methods in other application processing sessions.

To reuse the same methods:

1. Open the Suggest | Volunteer | SendMsg dialog window and place methods that you know you will reuse on the right side Send Message column.

2. Click on the **Keep** button above the right side Send Message column. The system gives these methods a permanently selected status.

   Additional methods you add to the list after clicking the **Keep** button will not have a permanently selected status. Click on the **Keep** button each time to retain new methods.

3. Click on the **OK** button from the dialog window and supply the desired message passing information for each SendMessage dialog window.

The system closes the Suggest | Volunteer dialog window without initiating processing.

See the previous procedure to complete the SendMessage window fields. See also the Intelligent Rules Element Language Reference manual SendMessage operator topic for details about triggering methods from a rule or method.

4. Optionally, select the Save Knowledge Base option on the Expert menu to reuse the same methods even after exiting the Rules Element.

If you do not save the knowledge base before exiting the Rules Element, the system restores any previously saved methods the next time you load the knowledge base.

If you no longer want methods to appear permanently on the right side Send Message column, you can unselect individual methods. Use the following procedure to unselect permanently selected methods from the right side Send Message column.

To unselect reused methods:

1. Open the Suggest | Volunteer | SendMsg dialog window to view the list of methods with a permanently selected status.

Closing and reopening the dialog window clears methods from the list that were not permanently selected.

2. If you want to unselect specific methods, double click on the method name in the right side Send Message column that you want to clear.

If you want to unselect all methods, click on the **Clean Up** button from the dialog window. The system clears the right side Send Message, Suggest *and* Volunteer columns.

3. Click on the **Keep** button above the right side Send Message column. The system "keeps" remaining selections if any.

4. Click on the **OK** button from the dialog window. The system closes the Suggest | Volunteer | SendMsg dialog window without initiating processing.

5. Optionally, select the Save Knowledge Base option on the Expert menu to retain the right side column's current status even after exiting the Rules Element.

If you do not save the knowledge base before exiting the Rules Element, the system restores the previously saved methods the next time you load the knowledge base.

Mixed-Mode

The Rules Element environment lets you suggest hypotheses and volunteer data for processing simultaneously. The Suggest | Volunteer | SendMsg dialog window is ideally suited for this purpose because it groups all valid processing structures in a single window. The window's left side columns display the list of selectable structures. Its right side columns display the selected structures thus letting you weight the necessity of each structure before initiating processing. Figure 4-3 shows a Suggest | Volunteer | SendMsg dialog window that uses mixed-mode.

The inference engine begins by backward chaining from the initial hypotheses ranked according to their inferencing priority numbers. Volunteered data receives a lower priority of evaluation. For a complete description of inferencing priorities, refer to the Intelligent Rules Element Language Programmer's Guide.
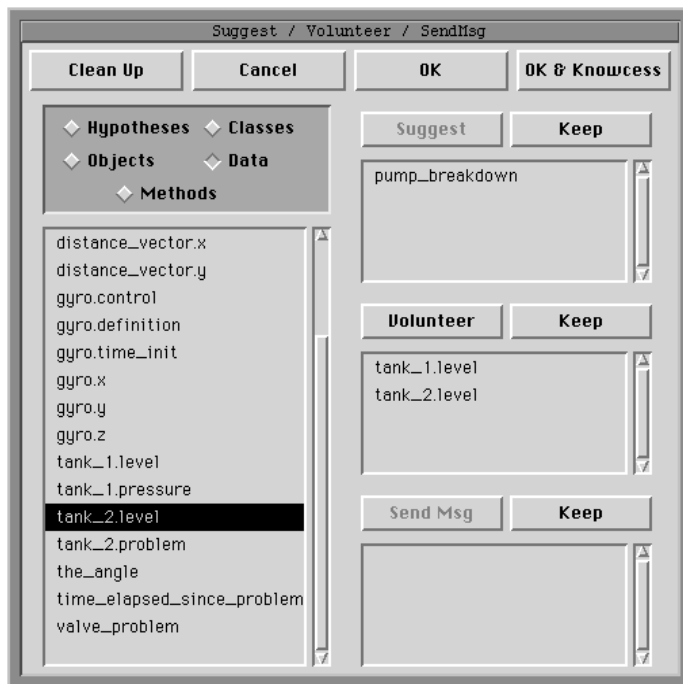
Figure 4-3   Mixed-Mode Processing

You can select as many object structures from the left side column lists as desired. Use the following procedure to initiate processing from the Suggest

| Volunteer | SendMsg dialog window using a combination of hypotheses and data.

To use mixed-mode:

1. Open the Suggest | Volunteer | SendMsg dialog window and place hypotheses that you want to suggest on the right side suggest column.

2. Click on the **Data**, **Classes**, or **Objects** button to display the desired list and place the data that you want to volunteer on the right side volunteer column.

3. If you want to initiate processing immediately, click on the **OK & Knowcess** button. The system displays a selection dialog window to obtain values for selected data.

4. If you prefer not to initiate processing yet want to save the current selections, click on the **OK** button. The system displays a selection dialog window to obtain values for selected data.

   If the data belongs to an object or class, the system also displays a dialog window to obtain the correct property.

5. Click on the desired value shown in the dialog window and click on the **OK** button. The system volunteers the data you supply.

   If the selected data is a subgoal hypothesis (one tested in a rule condition), the system lets you volunteer the values TRUE, FALSE, or NOTKNOWN.

6. Repeat the data selection procedure for each selection dialog window the system displays. The system either initiates application processing using the suggested hypotheses and volunteered data, or the system merely closes the Suggest | Volunteer dialog window. The action taken depends on which button invoked the selection dialog window.

   To close the Suggest | Volunteer dialog window and initiate processing, select the Start With... Knowledge Base option on the Expert menu.

List Windows

The list window options on the Browsers menu let you display categorized lists of rule and object structures. The lists they display are identical to the ones described for the left side column of the Suggest | Volunteer dialog window. Popup menus that you display for individual rule and object structures in the list windows let you initiate processing. And like the Suggest | Volunteer dialog window, the list windows provide an option to save selections for multiple application processing sessions.

Additionally, several list windows are useful for viewing data values. Refer to "Continuing the Session" for details about viewing these windows during processing. Table 4-3 describes the processing capabilities of each list window.

List Window	Description
Rules	Lets you suggest hypotheses. The boldface type identifies the hypothesis name from other rule components shown in the List of Rules window's textual format.
Hypotheses	Lets you suggest all hypotheses from a list of knowledge base hypotheses. An asterisk in front of hypothesis names distinguishes a terminal hypothesis from a subgoal hypothesis.
Data	Lets you volunteer data for any hypotheses and objects that appear in rule conditions. If a hypothesis is volunteered, the system accepts a value of TRUE, FALSE, or NOTKNOWN. Private data (from private slots) is displayed in this list but it cannot be volunteered.
Objects	Lets you volunteer data for structures that are either objects or hypotheses. The system classifies hypotheses as objects with the special property value (unless specifically assigned otherwise).
Classes	Lets you volunteer data for structures specifically defined as classes.
Properties	Lets you view the properties for all objects and classes, but does not allow processing.
Methods	Lets you specify methods to trigger with a given message. If a method is selected, the system prompts you for the name of an object to trigger the method and optionally arguments to pass to the method.

Table 4-3 List Window Descriptions

The following sections describe the task of initiating processing using list windows.

Suggest Hypotheses

The List of Hypotheses window lets you view the complete list of hypotheses contained in the currently loaded knowledge base files. The List of Hypotheses window displays both types of hypotheses as follows.

■ Terminal hypotheses (identified with an asterisk mark)

■ Subgoal hypotheses (hypotheses used in rule conditions).

Unlike the Suggest | Volunteer dialog window or the Rule Network window, the List of Hypotheses window does not let you volunteer a value

for subgoal hypotheses. To initiate processing in the List of Hypotheses window you must use the Suggest approach. Figure 4-4 shows the List of Hypotheses window with a popup menu that displays the Suggest option.
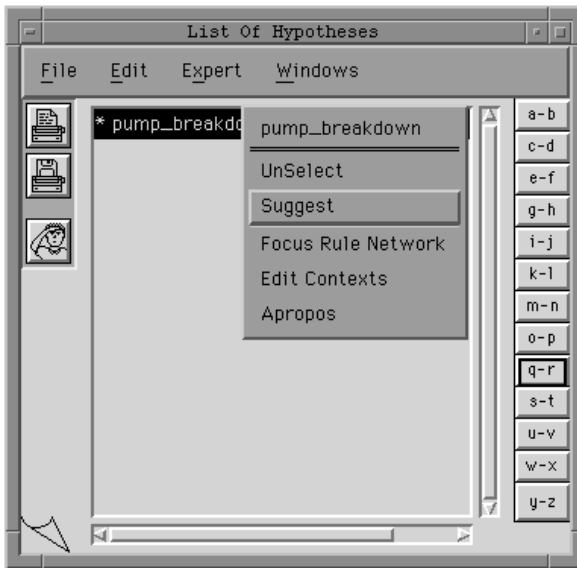
Figure 4-4   List of Hypotheses Window

Use the following procedure to select hypotheses and initiate processing from the List of Hypotheses window.

To suggest hypotheses from the List of Hypotheses:

1. Open the List of Hypotheses window and browse the window to display the desired hypothesis name.

   The list window highlights any hypotheses already assigned a permanently selected status in the Suggest | Volunteer dialog window.

2. Display the popup menu for the desired hypothesis name and select the Suggest option from the list. The system suggests the hypothesis and closes the popup menu.

   To unsuggest a hypothesis, redisplay the popup menu for that hypothesis and select the Unsuggest option from the list.

3. If you know you will reuse the suggested hypothesis, redisplay the popup menu and select the Select option from the list. The system highlights the hypothesis name.

   To unselect a hypothesis, redisplay the popup menu for that hypothesis

and select the Unselect option from the

4. If you want to initiate processing immediately, display the Windows popup menu and select the Knowcess option from the list. The system initiates application processing using the suggested hypotheses.

   This action simultaneously initiates processing using currently volunteered data if any.

5. If you prefer not to initiate processing yet want to keep the current selections for later use, close the list window.

   To close the list window and initiate processing, select the Start With... Knowledge Base option on the Expert menu. This action uses selected hypotheses and data.

The List of Rules window has the same Suggest option as the List of Hypotheses window, but the procedure for initiating processing is somewhat different. Figure 4-5 shows the List of Rules window with a popup menu that displays the Suggest option. The hypothesis name appears immediately following the keywords THEN or ELSE for the currently displayed rule.

Figure 4-5   List of Rules Window

Use the following procedure to select hypotheses and initiate processing from the List of Rules window.

To suggest hypotheses from the List of Rules:

1. Open the List of Rules window and browse the window to display the desired hypothesis name shown in boldface.

2. Display the popup menu for the current rule and select the Suggest Hypothesis option from the list. The system suggests the hypothesis and closes the popup menu.

    To unsuggest a hypothesis, redisplay the popup menu for the rule and select the Unsuggest Hypothesis option from the list.

3. If you want to initiate processing immediately, display the Windows popup menu and select the Knowcess option from the list. The system initiates application processing using the suggested hypotheses.

    This action simultaneously initiates processing using currently volunteered data if any.

4. If you prefer not to initiate processing yet want to keep the current selections for later use, close the list window.

    To close the list window and initiate processing, select the Start With... Knowledge Base option on the Expert menu. This action uses selected hypotheses and data.

Volunteer Data

Three list windows let you initiate processing using the volunteer data approach: List of Data, Object, and Class windows (see Table 4-3). The object structure lists that appear in these three list windows are identical to the ones found in the Suggest | Volunteer dialog window. However, the list windows do not compact the lists, and you must therefore browse the list windows to display the desired structures. Figure 4-6 shows the List of Data

window with a popup menu that displays the Volunteer option for data beginning with the letter C.
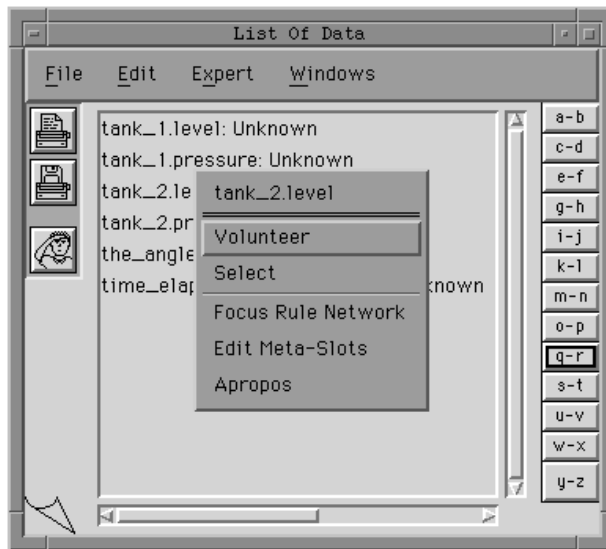
Figure 4-6 List of Data Window

The List of Data window shows both public and private slot data, but only public data can be volunteered. Private slot data can only be set by triggering a method for the slot. Use the following procedure to select public data and initiate processing from the List of Data, Object, or Class windows.

To volunteer data from the List of Data, Object, or Class windows:

1. Open the desired list window and browse the window to display the desired data name.

    The list window highlights any data already assigned a permanently selected status in the Suggest | Volunteer dialog window.

2. Display the popup menu for the desired data name and select the Volunteer option from the list. The system displays a selection dialog window to obtain values for selected data.

    If the data belongs to an object or class, the system also displays a dialog window to obtain the correct property.

3. Click on the desired value shown in the dialog window and click on the **OK** button. The system volunteers the data you supply.

    If the selected data is a subgoal hypothesis (one tested in a rule condition), the system lets you volunteer the values TRUE, FALSE, or

NOTKNOWN. To unvolunteer data, redisplay the popup menu for that data and select the Unvolunteer option from the list.

4. If you know you will reuse the volunteered data, redisplay the popup menu and select the Select option from the list. The system highlights the data name.

   To unselect data, redisplay the popup menu for that data and select the Unselect option from the list.

5. If you want to initiate processing immediately, display the Windows popup menu and select the Knowcess option from the list. The system initiates application processing using the volunteered data.

   This action simultaneously initiates processing using currently suggested hypotheses if any.

6. If you prefer not to initiate processing yet want to keep the current selections for later use, close the list window.

   To close the list window and initiate processing, select the Start With... Knowledge Base option on the Expert menu.

Network Windows

The network window options on the Browsers menu let you display the network diagram for rules and objects. Popup menus that you display for individual rule and object structures in either network diagram let you initiate processing. Unlike the Suggest | Volunteer dialog window or the list windows, the network diagrams do not provide an option to save selections for multiple application processing sessions.

The network windows are primarily useful when you are already working with the network diagram and want to quickly initiate processing. This capability also serves to reduce the number of windows needed to test your application, as described in the next chapter. The popup menu you display for selected rule and object structures in either network diagram will have

one of four possible Suggest | Volunteer combinations. Table 4-4 shows these popup menu options and their corresponding structures.

Menu Option	Structure
Suggest	A terminal hypothesis can only initiate processing if it is suggested. It cannot be volunteered since it is the final goal that you want to prove or disprove. The system determines terminal hypotheses' values through inferencing mechanisms (unless you specify otherwise through the Order of Sources).
Volunteer	Data that is the public slot of an object or class can only initiate processing if volunteered. The system accepts values that are either numeric or string types. Private slot data cannot be volunteered. The object network displays the properties of private slots as squares with the letter P inside, while the property of a public slot is represented by a square without the P.
Both (Suggest and Volunteer)	A hypothesis that is a subgoal can initiate processing using suggest or volunteer. If volunteered, the system accepts a value of TRUE, FALSE, or NOTKNOWN. If suggested, the system treats the sub-goal as a terminal hypothesis and determines its value through inferencing mechanisms.
not present	Data contained in rule condition expressions cannot initiate processing in the network window. Use the List of Data window to volunteer these specific data items for public slots.

Table 4-4   Network Window Processing

For information about displaying the network diagrams, refer to Chapter Three, "Application Editing." The following sections describe how to initiate processing using either network window

Suggest Hypotheses

The Rule Network window lets you easily identify the hypotheses of your application when you display the rule network diagram. The rule network diagram shows both types of hypotheses that you can suggest.

■ Terminal hypotheses that have one or more branching rules

■ Subgoal hypotheses that serve as data in rule conditions and can also have one or more branching rules.

The Object Network window also shows these same hypotheses that you can suggest because the system automatically creates objects for each hypothesis named in the Rule editor. Typically, hypotheses appear as standalone objects, unlinked to other classes or objects. Additionally, if you do not assign a specific property to the hypothesis, it possesses the default

property `Value`, although other objects may also use this property if none is specifically assigned. If you are not sure, you can always identify hypotheses in the object network diagram by displaying the popup menu for the property of the object. Only hypotheses will display the Suggest option.

Use the following procedure to select hypotheses and initiate processing from either the Rule or Object Network window.

To suggest hypotheses from the network diagrams:

1. Open the desired network window and browse the network diagram to display the desired hypothesis.

2. Display the popup menu for the hypothesis item and select the Suggest option from the list. The system suggests the hypothesis and closes the popup menu.

    On the object network diagram, the selection must be made on the property of the hypothesis. To unsuggest a hypothesis, redisplay the popup menu for that hypothesis and select the Unsuggest option from the list.

3. If you want to initiate processing immediately, display the Windows popup menu and select the Knowcess option from the list. The system initiates application processing using the suggested hypotheses.

    This action simultaneously initiates processing using currently volunteered data if any.

4. If you prefer not to initiate processing yet want to keep the current selections for later use, close the network window.

    To close the network window and initiate processing, select the Start With... Knowledge Base option on the Expert menu.

Volunteer Data

The Rule and Object Network windows also let you volunteer data that appears in the left-hand side conditions or right-hand side actions of a rule. Private data, visualized in the Object Network window, cannot be volunteered since it is never used in the conditions and actions of rules. The object network displays private slots with properties that have the letter P inside the square, while public slots have properties with an empty square. Additionally, you can easily identify whether the data in question can initiate processing by displaying the popup menu for the particular item. Only data that the system can use in the volunteer data approach actually display the Volunteer option.

> **Note:** The system is unable to distinguish individual data used in rule expressions; use the List of Data window to volunteer data that belongs to an expression.

Use the following procedure to select data and initiate processing from either the Rule or Object Network windows.

To volunteer data from the network diagrams:

1. Open the desired network window and browse the network diagram to display the desired data.

   When using the object network diagram display the property of the object structure to select.

2. Display the popup menu for the data item and select the Volunteer option from the list. The system displays a selection dialog window to obtain values for selected data.

   If the data belongs to an object or class, the system also displays a dialog window to obtain the correct property.

3. Click on the desired value shown in the dialog window and click on the **OK** button. The system volunteers the data you supply.

   If the selected data is a subgoal hypothesis (one tested in a rule condition), the system lets you volunteer the values TRUE, FALSE, or NOTKNOWN. To unvolunteer data, redisplay the popup menu for that data and select the Unvolunteer option from the list.

4. If you want to initiate processing immediately, display the Windows popup menu and select the Knowcess option from the list. The system initiates application processing using the volunteered data.

   This action simultaneously initiates processing using currently suggested hypotheses if any.

5. If you prefer not to initiate processing yet want to keep the current selections for later use, close the network window.

   To close the network window and initiate processing, select the Start With... Knowledge Base option on the Expert menu.

Continuing the Session

The following section describes ways to continue the application processing session. Due to the interactive nature of the Rules Element environment, all editing windows are available during inferencing. The implications of the

Rules Element's interactive nature are fully explored in Chapter Five, "Application Testing."

Supplying Data Input

After you initiate processing, the system always begins a backward search from the starting hypotheses. Along the way, the inference engine notices it has starting data to fire additional rules. It then shifts to a forward chaining strategy. Rules that the inference engine finds have insufficient data may result in a request for additional information from the user. First the system attempts to locate the data from your application. If the search fails to fetch the needed data, the system uses the session control panel on the main window to prompt the user for input. Figure 4-7 shows the format used to display questions to the end user in the main window.
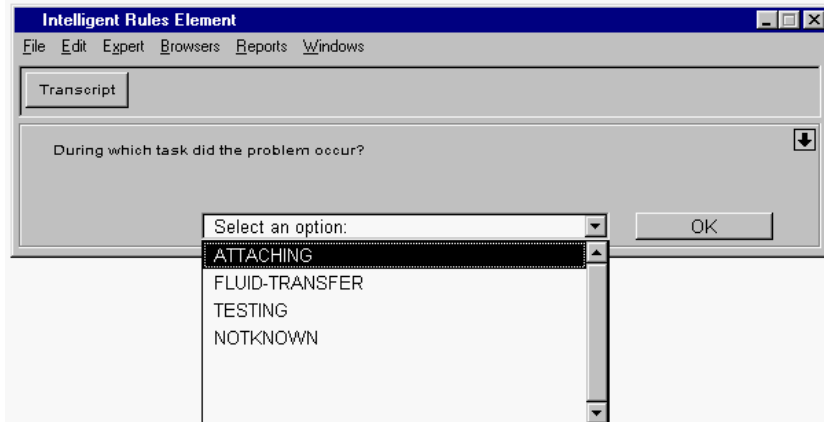
Figure 4-7   Session Control Panel with Question Displayed

Note: Figure 4-7 shows the Main Window after it has been reduced to display only the session control panel. The up arrow displayed on the right side of the panel provides this option. Click on the arrow again to expand the window to its original size.

Use the following procedure to input a value to the session control panel during application processing.

To supply a value:

1. Initiate processing using the suggest, volunteer, or send message options.

2. The system pauses to obtain a value for a slot it cannot determine. The main window displays the question in the session control panel.

   The input field displayed with the question may contain a value if you initiate the session using a journal file. See Chapter Six, "Application Documentation."

3. Click on the desired value from the session control panel. The system inserts the value in the input field.

   Double-clicking on the value selects and volunteers the value.

4. Click on the **OK** button from the session control panel. The system continues processing using the volunteered data.

Before the application processing session ends and the system displays the "Done" message in the session control panel of the main window, you can interrupt the session and open other windows in the Rules Element environment. This capability of the Rules Element inference engine to interrupt its state means you can process your application interactively. The implications of interactive processing are the subject of Chapter Five, "Application Testing." Use the following procedure to interrupt the current application processing session and view the explanation facility, for example.

To interrupt application processing:

1. Click on the **Interrupt** button shown in the session control panel. The system pauses the inference engine and displays the **Continue** button.

   You can also use the following keyboard commands to interrupt the session.
   a. IBM PC and compatible users press the Ctrl+Alt keys.
   b. Macintosh users press the Command+period (.)keys.
   c. Workstation users press the CTRL+backslash (\) keys.
   Alternately, wait until the system displays a question and do not supply a value.

2. Display the session control panel popup menu and select the **Why** option from the list. The system displays the explanation dialog window.

3. Click on the **OK** button from the explanation dialog window. The system closes the explanation dialog window.

4. Select the **Continue** button on the session control panel. The system releases the inference engine and continues processing.

Making Revisions

The Rules Element inference engine's event-driven architecture enables you to express the application naturally, using symmetrical rules. This is particularly useful when the application user cannot state the goal in advance. Instead, the user can offer observations about various facts from which the system draws preliminary conclusions. Then the system can use the same rules to pinpoint their causes and provide solutions by backward chaining, and the user can further refine the goal using the established facts and conclusions by continuing application processing.

When you place yourself in the role of the user and offer new data for processing, the system automatically considers the consequences of the new information to related rules. Modifications you make may trigger enough revisions to continue processing. It could also trigger the investigation of a previously unused portion of the knowledge base. This operation is commonly referred to as a "What-If operation."

List of Data Window Revisions

The List of Data window is ideally suited for making the desired revisions. Not only does it list all the data contained in the currently loaded knowledge base files, it also lists their resulting values. Use the following procedure to trigger revisions using modified data at the end of application processing.

To modify data:

1. Open the List of Data window and display the popup menu for the desired data. The list window shows the current value of the data.

2. Select the Modify option from the list. The system displays a selection dialog window to obtain values for selected data.

3. Click on the desired value shown in the dialog window and click on the **OK** button. The system volunteers the data you supplied.

4. If you want to initiate processing immediately, display the Windows popup menu and select the Knowcess option from the list. The system initiates application processing using the modified data.

    This action simultaneously initiates processing using currently suggested hypotheses if any.

5. If you prefer not to initiate processing yet want to save the current selections, close the list window.

    To initiate processing after closing the network window, select the Start With... Knowledge Base option on the Expert menu.

It is also possible to modify data interactively, during application processing. Again, the List of Data window permits this operation because the system dynamically updates the list of values. Use the following procedure to interrupt the session and trigger revisions using modified data.

To interrupt the session and modify data:

1. Click on the **Interrupt** button shown in the session control panel. The system pauses the inference engine and displays the **Continue** button.

    You can also use the following keyboard commands to interrupt the session.
    a. IBM PC and compatible users press the Ctrl+Alt keys.
    b. Macintosh users press the Command+period keys.
    c. Workstation users press Ctrl+backslash keys.
    Alternately, wait until the system displays the next question, the inference engine pauses automatically.

2. Open the List of Data window and display the popup menu for the desired data.

3. Select the Modify option from the list. The system displays a selection dialog window to obtain values for selected data.

4. Click on the desired value shown in the dialog window and click on the **OK** button. The system volunteers the data you supply.

5. Close the list window and select the **Continue** button on the session control panel. The system releases the inference engine and continues processing using the modified data.

Network Window Revisions

The Rule and Object Network windows can also be used to make limited revisions. These windows provide a Reset option on the local popup menu of certain items that lets you set the item to its initial value of UNKNOWN.

In the Object Network, the Reset option appears on the local popup menu of a slot. Selecting the Reset in this case resets the value of the slot to UNKNOWN.

In the Rule Network, the Reset option appears on the local popup menu of a hypothesis. When Reset is selected for a terminal or sub-goal hypothesis, the selected hypothesis and all the hypotheses leading to it will all be reset to UNKNOWN. You cannot reset an individual sub-goal hypothesis since all

other sub-goals of this hypothesis will also be reset in the Rule Network window.

Restarting the Session

At the conclusion of application processing the system displays the "Done" message in the session control panel on the main window. If desired, you can open the List of Data window and revise the previous session's results to continue processing (as described in the previous section). If you do not want to reuse the previous session's results, you can return the application to its original state where the values of all data and hypotheses are UNKNOWN. Figure 4-**8** shows the session control panel as it appears at the end of the processing session.
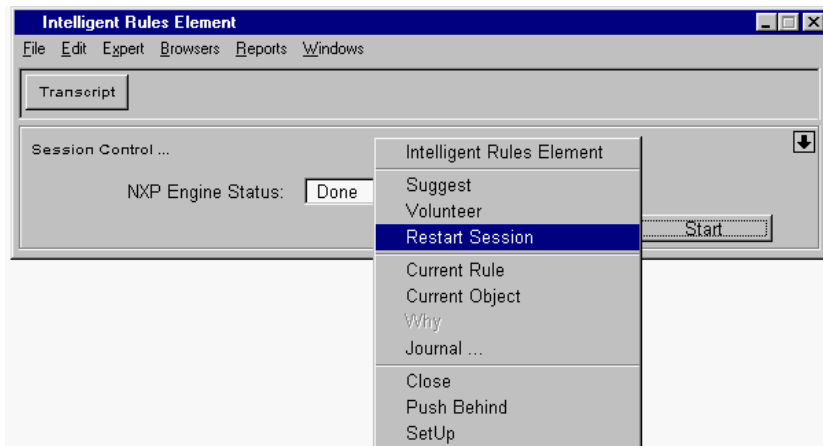
Figure 4-8 Restart Session Message

The Restart Session option on the Expert menu lets you clear the results of the previous application processing session. Figure 4-**8** shows the same Restart Session option is available on the session control panel popup menu. The Restart Session command has the following specific effects on the application.

■ Clears the contents of all monitoring windows
■ Clears the suggest or volunteer selections
■ Clears knowledge base values obtained during the session
■ Deletes dynamic objects and their links created during the session
■ Restores object links temporarily deleted during the session
■ Restores current strategy settings to default settings defined in the

Strategy Monitor window.

You can automate the restarting process by selecting the Restart Session checkbox in the Set Up Environment dialog window. If this option is selected, the system automatically restarts the application after you initiate processing. For additional information about automatic session restarts, refer to Chapter Five, "Application Testing."

Use the following procedure to manually restart the session and begin processing using the original UNKNOWN states for data and hypotheses.

To restart the session:

1. Select the Restart Session option on the session control panel popup menu. The system returns the application to its original state and clears dynamic object structures.

    If the session is not yet concluded and you want to restart the session, use the same action. You can also select the Restart Session option on the Expert menu.

2. Suggest new hypotheses or volunteer new data as desired and initiate processing.

5 *Application Testing*

Application developers should read this chapter to test their applications using the Intelligent Rules Element shell and its monitoring facilities.

Introduction

This chapter gives useful information about testing the Rules Element applications. It describes the function of a variety of windows you use to monitor and interact with the running application. The windows that comprise the application testing environment include the following.

■ Current Rule, Hypothesis, and Conclusion windows

■ Case Status and Full Report windows

■ Transcript window

■ Explanation facility

■ Strategy Monitor window

■ Rule and Object Network windows

■ Cross Reference window

■ Agenda Monitor window.

Even if you do not use the Rules Element shell to deliver your final application, you can run the growing application for testing purposes. The Rules Element shell facilitates testing an application because it combines the processing algorithms, editing facilities, and specialized monitoring facilities in a single environment. Under this environment you can add new rule and object structures and determine their effect on the existing design. This activity is referred to as "application testing."

You can initiate testing at virtually any stage of the application development process. The system requires no minimum number of rules to conduct an inferencing session in the Rules Element environment. This allows you to add rule and object structures incrementally and test their effect on the existing application.

Setting Up the Testing Environment

This section describes optional features you can use to customize the testing environment. The topics include.

■ Displaying monitoring windows automatically
■ Starting up a testing session automatically
■ Specifying inference strategies for the entire system
■ Specifying inheritability scheme for the entire system
■ Specifying inheritance search strategy for the entire system
■ Saving environment settings between sessions.

Set Up Environment Dialog Window

The Set Up Environment option on the File menu displays a dialog window that you can use to control the behavior of monitoring windows. Figure 5-1 shows the Set Up Environment dialog window with only the options that pertain to application testing selected.
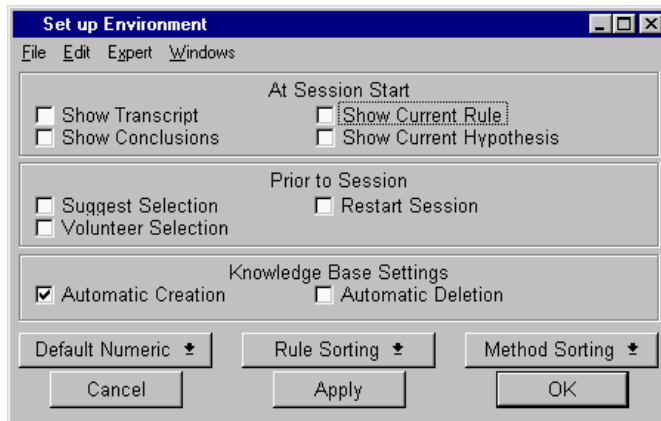
Figure 5-1   Set Up Environment Dialog Window

Note: To reuse your Set Up Environment settings in later sessions, select the Save Environment option on the File menu. The system updates the ndset.dat file with the new settings.

Refer to Chapter Three, "Application Editing" for information about the unselected options shown in Figure 5-1. The following sections describe the application testing options.

When Starting a Session

The Set Up Environment dialog window lets you select options that determine whether the Rules Element monitoring windows automatically appear with the onset of inference engine processing. Normally, you would

display the following windows individually by selecting the corresponding option in the Reports menu.

■ Current Rule window

■ Current Hypothesis window

■ Conclusions window

■ Transcript window.

If you know that you will want to monitor the inference engine events that take place during processing, you can cause any or all of these windows to appear automatically. Use the following procedure to permit the system to display individual monitoring windows automatically.

To cause monitoring windows to display automatically:

1. Select the Set Up Environment option on the File menu. The system displays the options dialog window.

2. Click on the checkboxes under the At Session Start section that correspond to the monitoring windows you want to keep. The system highlights each selected checkbox.

3. Click on the **OK** button from the dialog window. The system records the new settings for the current session.

    The system automatically write enables each monitoring window that you select in the Set Up Environment window. To disable these windows, display each window from the Reports menu and select the Disable Write option from its popup menu. You can close the disabled window if desired.

4. Initiate processing. The system displays the main window session control panel and any monitoring windows you selected.

Prior to the Session Startup

Another set of options that you can select from the Set Up Environment dialog window lets you completely automate the inferencing startup process. These particular options are grouped under "Prior to Session" and include the following.

■ Suggest Selection

■ Volunteer Selection

■ Restart Session.

Normally, you might re-select the startup parameters (data and/or hypotheses) in the Suggest | Volunteer dialog window. If you want to reuse the selected parameters, you can select the Suggest Selection and Volunteer Selection options. The Restart Session option lets you automatically clear

data and conclusions from the system after completing an inferencing session.

Whether you select one or all of these options determines the degree of automation, as well as the effect on the next inferencing session. The selection status of the Restart Session option is especially important because it determines whether the system initiates processing with known data.

Note: Refer to Chapter Four, "Application Processing" for complete details about the Restart command's effect on the application.

The following procedure shows how to automate a session so you can recycle the same starting parameters. You can also try different combinations as desired. Use the following procedure to automatically restart the just concluded inferencing session.

To permit fast inferencing session restarts:

1. Select the Set Up Environment option on the File menu. The system displays the options dialog window.

2. Click on the three checkboxes under the Prior to Session section that correspond to the options: Suggest, Volunteer, and Restart. The system highlights the square to show its selected status.

    Restarting the session returns the knowledge base to its original state and clears dynamic object structures.

3. Click on the **OK** button from the dialog window. The system records the new settings for the current session.

4. Select the Start With... Knowledge Base command on the Expert menu to initiate processing.

Save Environment

The Save Environment option on the File menu lets you reuse the environment settings currently in affect for the next session. The system stores the current settings in the `ndset.dat` file and automatically reads the file the next time you launch the Rules Element. Saving environment settings has the following effects.

■ It saves the Set Up Environment dialog window selections.

■ It saves the write status of the monitoring windows.

■ It saves the location and size of currently open windows.

■ It saves the network window settings.

■ It saves the color and font definitions specified for windows.

> **Note:** Appendix D, Customizing the Environment describes the customization options for individual windows that you can save.

For example, if you want to continue testing the application but have to quit your current session, you might want to select the Save Environment option. Use the following procedure to save the settings you specify for the Set Up Environment dialog window and the write status of monitoring windows.

To save the current environment settings:

1. Select the Save Environment option on the File menu. The system displays the confirmation dialog window.

2. Click on the **OK** button from the dialog window. The system updates the `ndset.dat` file with the new settings.

   The file's location on your platform is determined by the search path specified for all files. For PC, VMS, and Unix the path is the default `ND_DATA`.

Strategy Monitor Window

The Strategy option on the Expert menu displays a monitor window that you can use to control the global behavior of the inference engine. The Strategy Monitor window is divided into several distinct components related to inference engine processing as follows.

- The inheritability settings
- The inheritance search strategy settings
- The inference strategy settings (with a separate set of Forward Action Effects)
- The system method evaluation settings
- The data validation settings.

The Strategy Monitor window lets you alter these strategies in two modes: default or current. The default mode shows the settings you want the system to use at the outset of processing. The current mode reflects changes to the default strategies that occur as the system evaluates the rules and meta-slots you define. The mode displayed by the Strategy Monitor

window depends on your selection of the Default and Current buttons. Figure 5-2 shows the window with the Default option selected.
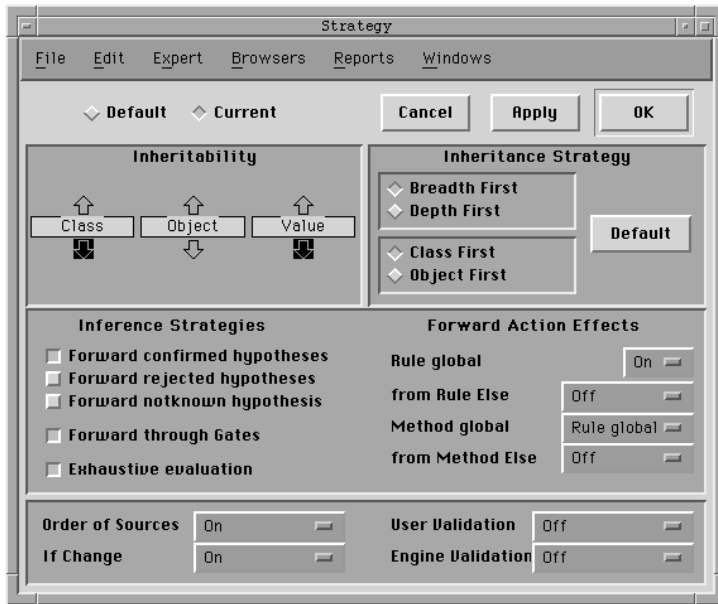
Figure 5-2   Strategy Monitor Window

The following sections show how to modify the default settings to initialize an application testing session. The system does not require you to alter these settings, but the operations described may aid in detecting unwanted side-effects. If you choose not to alter the system's default configuration, your application starts with the full range of inference and inheritance strategies as shown in Figure 5-2.

Note: Changing Strategy settings is an advanced operation and requires familiarity with the concepts expressed in the Intelligent Rules Element Language Programmer's Guide.

Inheritability Strategies

The Inheritability component of the Strategy Monitor window (see Figure 5-2) has three graphic displays that give you control over the way inheritance occurs throughout the system. The unmodified global behavior of the system lets objects inherit properties and values down from classes only.

You can modify this behavior, for example, to let classes inherit properties from objects as well. The arrows that you select for these displays determine the direction of inheritance as follows.

■   The up arrow means that the parents of the object or class can inherit up from their child.

■   The down arrow means the children of the object or class can inherit down from their parent.

Once you determine the default global behavior of the application, the inheritability setting is seldom changed during application testing. However, you may want to change to specify a global setting before testing that is consistent with your application. Use the following procedure to modify the default inheritability scheme in the Strategy Monitor window, for all slots (either public or private) in the application.

To modify the global inheritability scheme:

1.   Select the Strategy option from the Expert menu. The system opens the Strategy Monitor.

2.   Click on the **Default** button from the Strategy Monitor. The system displays the default settings.

3.   Click on the Class, Object, or Value display arrows to select or unselect them as desired. The system highlights the arrow to show its selected status.

4.   Click on the **OK** button from the Strategy Monitor to compile the new global default settings.

Inheritance Strategies

The Inheritance Strategy component of the Strategy Monitor window (see Figure 5-2) is two sets of checkboxes that together determine the pathways through the class and object network. The system follows certain predetermined pathways in order to propagate methods, properties, and property values. You can modify the system's default search pathways by choosing one of the following strategy checkbox combinations:

■   Class-First, Breadth-First search strategy.

■   Class-First, Depth-First search strategy.

■   Object-First, Breadth-First search strategy.

■   Object-First, Depth-First search strategy.

In case the search can begin with either a class or an object, the Class and Object checkboxes let you select the search starting point. The Breadth and Depth checkboxes let you determine the search order from the selected

parent. Use the following procedure to modify the global inheritance strategy in the Strategy Monitor window, for all slots in the application.

To modify the global inheritance search strategy:

1. Select the Strategy option from the Expert menu. The system opens the Strategy Monitor.

2. Click on the **Default** button from the Strategy Monitor. The system displays the default settings.

3. Click on the Class or Object checkbox to select the beginning point of the inheritance pathway as desired. The system highlights the checkbox to show its selected status.

4. Click on the Breadth First or Depth First checkbox to determine how the search will proceed through the inheritance pathways as desired. The system highlights the checkbox to show its selected status.

5. Click on the **OK** button from the Strategy Monitor to compile the new global default settings.

    Click on the Apply button to approve the changes without exiting the Strategy Monitor.

General Inference Strategies

Table 5-1 shows the Strategy Monitor window global inference strategies default mode of operation.

Strategy	Default	Description
Forward Confirmed Hypotheses	ON	Rule whose hypothesis evaluates to TRUE propagates the inference process to weakly linked hypotheses via the context mechanism.
Forward Rejected Hypotheses	OFF	Rule whose hypothesis evaluates to FALSE propagates the inference process to weakly linked hypotheses via the context mechanism.
Forward NotKnown Hypotheses	OFF	Rule whose hypothesis evaluates to NOTKNOWN propagates the inference process to weakly linked hypotheses via the context mechanism.
Forward Thru Gates	ON	Rule whose data makes the LHS conditions of another rule TRUE propagates the inference process to that rule's hypothesis via the gates mechanism.
Exhaustive Evaluation	ON	All rules leading to a suggested hypothesis will always be evaluated, even after the value of the hypothesis has already been determined to be TRUE by a previous rule.

Table 5-1 Inference Strategies

Re-configuring the inference strategies is an optional operation. If you do not make adjustments to the default behavior, the system conducts processing with most of the strategies enabled. However, you can disable combinations of these strategies to simplify the behavior of the inference engine during application testing. For instance, by disabling the three strategies related to contexts, you can focus the inference engine on a single knowledge island. Or, by disabling all but one strategy you can isolate its effect on the application. The Strategy operator used in a rule or method provides local control of selected strategies in order to override global settings as needed.

Use the following procedure to configure the inferencing session for its most basic mode of operation.

To initialize default inference strategies:

1. Select the Strategy option from the Expert menu. The system opens the Strategy Monitor.

2. Click on the **Default** button from the Strategy Monitor. The system displays the default settings.

3. Select the checkbox options that correspond to as many of the inference strategies as you can effectively disable. You may want to disable all strategies to start. The system unhighlights the small square.

    During the actual application testing session, you enable the previously disabled strategies one at a time in order to view its effect on the application.

4. Click on the **OK** button from the Strategy Monitor to compile the new global default settings.

    Click on the Apply button to approve the changes without exiting the Strategy Monitor.

Forward Action Effects Strategies

Table 5-2 shows the Strategy Monitor window forward action effects default mode of operation.

Strategy	Default	Description
Rule Global	ON	Rules whose actions (from LHS or RHS) change the value of shared data of another rule's LHS conditions propagates the inference process to that rule's hypothesis via the action effects mechanism. Note: Slot values can be changed by the Assign, Retrieve, or Execute operators.
from Rule Else	OFF	Rules whose Else actions change the value of shared data of another rule's LHS conditions propagates the inference process to that rule's hypothesis via the action effects mechanism.
Strategy	Default	Description
Method Global	GLOBAL Uses Rule Global (ON)	Methods whose actions (from LHS or RHS) change the value of shared data of a rule's LHS conditions propagates the inference process to that rule's hypothesis via the action effects mechanism.
from Method Else	OFF	Methods whose Else actions change the value of shared data of a rule's LHS conditions propagates the inference process to that rule's hypothesis via the action effects mechanism.

Table 5-2   Forward Action Effects Strategies

Re-configuring the inference strategies is an optional operation. If you do not make adjustments to the default behavior, the system conducts processing with the forward action effects strategy for rules and methods enabled (except from the Else side of a rule or method which is disabled by default). However, you can disable this strategy to simplify the behavior of

the inference engine during application testing. The forward action effects strategy options include:

ON   Enables the strategy for the entire system.

OFF   Disables the strategy for the entire system.

GLOBAL   This option is used to synchronize control of the three Forward Action Effects strategies with the setting of Rule Global Forward Action Effects. If you select the GLOBAL option the strategy behaves exactly as the Rule Global setting which specifies the LHS and Then action behavior.

The Strategy operator used in a rule or method provides local control of selected strategies in order to override global settings as needed. Use the following procedure to configure the inferencing session for its most basic mode of operation.

To initialize default forward action effects strategies:

1. Select the Strategy option from the Expert menu. The system opens the Strategy Monitor.

2. Click on the **Default** button from the Strategy Monitor. The system displays the default settings.

3. Click on the Rule Global menu button and select the option OFF from the list. The system displays the option OFF in the menu button.

    The Method Global strategy follows the current Rule Global setting when its GLOBAL option has been selected.

4. Click on the **OK** button from the Strategy Monitor to compile the new global default settings.

    During the actual application testing session, you enable the previously disabled strategies one at a time in order to view its effect on the application.

Method Evaluation Strategies

Table 5-3 shows the Strategy Monitor window global method evaluation strategies default mode of operation.

Method	Default	Description
Order of Sources	ON	Order of Sources methods are in effect and will be executed when appropriate.
If Change	ON	If Change methods are in effect and will be executed when appropriate.

Table 5-3 Method Evaluation Strategies

Re-configuring the method evaluation strategies is an optional operation. If you do not make adjustments to the default behavior, the system conducts processing with system methods (Order of Sources and If Change) enabled. However, you can disable system methods to simplify the behavior of the inference engine during application testing. The available options include:

ON	Enables the method (either Order of Sources or If Change) for the entire system. (Default)
OFF	Disables the method (either Order of Sources or If Change) for the entire system.
CONTINUE	Enables the Order of Sources method and forces the system to execute every action in the actions list from top to bottom. (When "ON" is selected the system stops after the value is found.)
UNKNOWN	Enables the If Change method and forces the system to trigger an If Change method for a slot whose value is reset to the value Unknown. (When "ON" is selected resetting a slot does not trigger the If Change method.)

The Strategy operator used in a rule or method provides local control of selected strategies in order to override global settings as needed. Use the following procedure to configure the inferencing session for its most basic mode of operation.

To initialize default method evaluation strategies:

1. Select the Strategy option from the Expert menu. The system opens the Strategy Monitor.

2. Click on the **Default** button from the Strategy Monitor. The system displays the default settings.

3. Click on the system method menu button (Order of Sources or If Change) and select the option OFF from the list. The system displays the option OFF in the method menu button.

   During the actual application testing session, you enable the previously disabled strategies one at a time in order to view its effect on the application.

4. Click on the **OK** button from the Strategy Monitor to compile the new global default settings.

   Click on the Apply button to approve the changes without exiting the Strategy Monitor.

Data Validation Settings

Table 5-4 shows the Strategy Monitor window global data validation strategies default mode of operation.

Strategy	Default	Description
User Validation	OFF	New values provided by the end user during processing are tested by the system against the data validation functions defined in the Meta-Slot editor or the Property editor.
Engine Validation	OFF	New values provided by the system during processing are tested against the data validation functions defined in the Meta-Slot editor or the Property editor.

Table 5-4   Data Validation Strategies

If you do not make adjustments to the default data validation behavior, the system conducts processing without data validation (end user and system) disabled. However, you can enable data validation to ensure the data validation functions defined in the Meta-Slot editor and Property editor operate as intended. The data validation strategy options include:

ON	Enables data validation (either User or Engine) for the entire system.
OFF	Disables data validation (either User or Engine) for the entire system. (Default)
ON/ACCEPT	Enables data validation (either User or Engine) for the entire system, but accepts the value entered when the data validation expression contains a slot not yet evaluated.
OFF/REJECT	Enables data validation (either User or Engine) for the entire system, but rejects the value entered when the data validation expression contains a slot not yet evaluated. In the case of a question, system re-asks the question. In the case of an engine change, the system will set the slot and action to NOTKNOWN.

The Strategy operator used in a rule or method provides local control of selected strategies in order to override global settings as needed. Use the following procedure to enable data validation.

To enable data validation:

1. Select the Strategy option from the Expert menu. The system opens the Strategy Monitor.

2. Click on the **Default** button from the Strategy Monitor. The system displays the default settings.

3. Click on the validation menu button (User or Engine) and select the option ON from the list. The system displays the option ON in the data validation menu button.

4. Click on the **OK** button from the Strategy Monitor to compile the new global default settings.

    Click on the Apply button to approve the changes without exiting the Strategy Monitor.

Journal Facility

Application testing implies introducing small changes to the rule and object structures and observing the effect through many iterations of the inferencing session. In one application this could mean supplying data through the main window session control panel, whereas another application may derive its data entirely from a database. If your application requires you to manually input data, the task of re-keying the same data each time you initiate processing would quickly become tedious. To ease this task the Rules Element shell lets you record and replay data entry steps.

A window appears when you select the Journal command that lets you record and replay a session. The Replay panel of the window has several options that you can use to control the replay mode. The following sections describe the Journal facility used for application testing.

Record Start and Stop

The Recording Start and Stop buttons let you "tape" the events of an inferencing session that control processing. The specific events the system records include volunteered data, suggested hypotheses, modified data, and end user inputs. The system stores these events in an extended NXP file format that you can view and modify with any text editor. Each line in the journal file has the following format.

```
\slotname\="value"@X
```

where X is:

S	for a suggested event
V	for a volunteered event
Q	for a question event

Note: The Journal facility does not record interactions with external functions, processes, or files. This includes actions initiated by the Execute, Write, and Retrieve operators.

You can invoke the Journal facility either before or during an inferencing session. The record function continues even after the end of the session so you can continue inferencing using the modified data or different hypotheses. Use the following procedure to record your data entry steps in a journal file.

To record a journal file:

1. Select the Journal option on the Reports menu. The system opens the Journal window.

    Select the Journal option on the session control panel popup menu once the session begins.

2. Select the **Record** button. The system displays the Record panel options.

3. Select the **Start** button from the Record panel. The system positions the cursor in the text edit line.

4. Edit the file name as necessary and change the search path if desired.

5. Click on the **OK** button to enable the file currently shown in the text edit line for recording.

    Select the Apply button to keep the Journal window displayed.

6. Begin the inferencing session, the system automatically records the inferencing events in the journal file.

7. To discontinue the journal, redisplay the Journal window and select the "Record" **Stop** button. Click on the **OK** button to confirm.

    If you do not manually discontinue the journal, the system continues journalling until you select the Restart Session option (on the Expert menu, for example). This has the same effect as selecting the Stop button.

Replay Start and Stop

The Replay Start and Stop buttons work with the replay option settings to determine how to replay previously recorded journal files. There are essentially two ways of replaying a recorded session, interactively and non-interactively. The replay options you select from the Journal window

depend on the type of application testing you want to perform. Table 5-5 shows the options and explains their role in testing.

Replay Option	Status	Description
Step by Step	Selected (the default)	Replays the questions in the same order as the original session. Used when interactively replaying and modifying data are desired.
Step by Step	Unselected	Prevents the session control panel from displaying and assumes the values are correct as recorded. Used when replaying non-interactively, often with the Skip Show Statements option selected.
Don't Scan the File for Values	Selected	Makes the order of data entry an important criteria for inferencing from the journal file. Used when testing specifically whether application modifications will cause a failure for the data given.
Don't Scan the File for Values	Unselected (the default)	Uses the data available to continue inferencing from the journal file regardless of order. This allows the greatest flexibility for testing.
Skip Show Statements	Selected	Prevents files originally invoked by the Show operator from appearing. Used when replaying non-interactively.
Skip Show Statements	Unselected (the default)	Allows files originally invoked by the Show operator to appear. Used when replaying interactively and additional inferencing breakpoints are desired.

Table 5-5   Journal Debugging Replay Options

The following sections describe the two ways to replay previously recorded journal files.

Non-Interactive Data Replay

The recorded journal file lets you return quickly to the location in the application where your modifications occur. This way of replaying involves recording an already debugged portion of the application. If you plan to modify certain rule and object structures, record the interactions you know will not change and avoid making modifications to the recorded portion of the application. Use the following procedure to capture your keystrokes and then replay without pauses in order to begin testing.

To replay a journal file non-interactively:

1. Load the original knowledge base file(s) if necessary.

2. Select the Journal option on the Reports menu. The system opens the Journal window.

   Select the Journal option on the session control panel popup menu once the session begins.

3. Select the **Record** button. The system displays the Record panel options.

4. Select the **Start** button from the Record panel. The system positions the cursor in the text edit line.

5. Enter the file name of the desired journal file and change the search path if necessary.

6. Click on the **Step By Step** option. The system unhighlights the square to indicate its unselected status. (The default is selected.) Click on the **Skip Show Statements**. The system highlights the square to indicate its selected status.

7. Click on the **OK** button to enable the file currently shown in the text edit line for replay. The system automatically replays the entire journal file.

   Select the Apply button to keep the Journal window displayed.

8. The application testing session resumes with the first unrecorded question automatically displayed by the system.

Interactive Data Replay

Another capability of the Journal facility lets you conduct application testing iterations using different data. This task is related more to the "Tracking Inferencing Events" section because it does not involve editing the rule and object structures. For instance, you can replay a session after modifying data in the text format journal file, or you can interactively modify the data through the main window session control panel. Use the following procedure to modify journal file data through the session control panel.

To replay a journal file interactively:

1. Load the original knowledge base file(s) if necessary.

   Modifications you make to a recorded application may prevent the journal file from working properly.

2. Select the Journal option on the Reports menu. The system opens the Journal window.

   Select the Journal option on the session control panel popup menu once

the session begins.

3. Select the **Replay** button. The system displays the Replay panel options.

4. Select the **Start** button from the Replay panel. The system positions the cursor in the text edit line.

   The default option settings work best for interactive replay. Ensure the **Step By Step** option is selected.

5. Enter the file name of the desired journal file and change the search path if necessary.

6. Click on the **OK** button to enable the file currently shown in the text edit line for replay. The system displays the question in the session control panel with the previously recorded value displayed in the input field.

   Select the Apply button to keep the Journal window displayed.

7. Answer the question as desired and click on the **OK** button to continue replaying the journal file with the new value.

   If you edited the actual journal file itself (with a text editor), the system will automatically show the new value in the session control panel; you need only select the OK button to continue replaying.

8. Continue processing values until the journal file finishes replaying.

   To manually discontinue the journal, redisplay the Journal window and select the "Replay" Stop button. Click on the OK button to confirm.

Tracking Inferencing Events

After you initiate processing, the system searches your application using the inferencing mechanisms described in the Intelligent Rules Element Language Programmer's Guide. The path the inference engine follows is not determined in advance; it depends on the goal you set for the system and the data provided. Naturally you expect the system to reach certain conclusions for a particular set of data. But if you were unable to follow the reasoning pathways each time you changed the goal or volunteered new data, the end result might seem bewildering. Fortunately, the Rules Element shell supplies facilities that let you monitor the inferencing session as it

occurs. Table 5-6 shows the set of special function buttons these windows let you select.

Button	Description
	Prints the contents of the current facility to a file.
	Sends the contents of the current facility to the printer.
	When this icon is displayed, it means the write to window option is disabled. This icon toggles with the write enable icon shown below.
	When this icon is displayed, it means the write to window option is enabled. This icon toggles with the write disable icon shown above.

Table 5-6   Common Function Buttons

The following sections describe the facilities that let you track the active reasoning pathways of your application. These particular facilities let you determine when and where the system diverged from the anticipated pathway. Use these facilities as a first level of application testing; they help you identify specific areas to investigate in detail.

Focus of Attention

The inferencing mechanisms of the Rules Element are based more on the notion of "focus of attention" than algorithmic approaches. Hence facilities exist in the windowing environment that let you observe the system as it changes the focus of attention through the application search space. These facilities give you the most basic level of information needed to observe an active inferencing session. You can enable these facilities for use either before initiating processing or during a session.

Note: "Write enabling" the following facilities is not required if each facility's Show option is selected in the Set Up Environment dialog window.

Current Hypothesis

The Rules Element inference engine is geared to evaluate rule conditions in order to find the value of their hypotheses. In the process of evaluating the conditions that lead to one hypothesis, the system can change the focus of attention to evaluate related hypotheses. To view the changing focus of

attention, you can display the Current Hypothesis window. Figure 5-3 shows the window as it might look during an inferencing session. It contains the name of the hypothesis the system is currently evaluating.
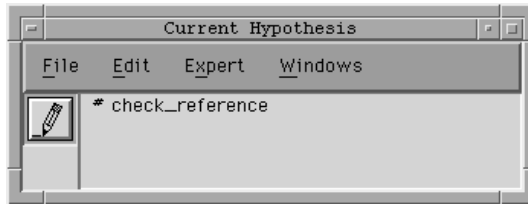
Figure 5-3   Sample Current Hypothesis Window

The Current Hypothesis window is available from the Reports menu either before initiating processing or during a session. Use the following procedure to view a dynamic trace of the system's changing focus of attention.

To enable the Current Hypothesis window:

1. Select the Show Current Hypothesis option on the Reports menu. The system opens the Current Hypothesis window.

2. Click on the write enable icon. The system enables the window.

    If the window is already enabled, from the Set Up Environment window for example, the write disabled icon appears in place of the write enabled icon.

3. Initiate processing and reposition the windows as desired. The system shows the name of the hypothesis currently under evaluation in the Current Hypothesis window.

    If the window updates too quickly to read, you can place breakpoints on the network windows to slow the inference engine.

Current Rule

Similar to the focus of attention trace the Current Hypothesis window provides, the Current Rule window lets you dynamically follow rules the system processes. In addition to the hypothesis name, the Current Rule window shows the rule's conditions. The dynamic trace of the Current Rule window usually changes more frequently than the Current Hypothesis window. This occurs when several rules share the same hypothesis. In this case the Current Hypothesis window maintains the same display even as the system evaluates successive rules. Only the Current Rule window gives you this additional information.

Figure 5-4 shows the window as it might look during an inferencing session. It is comprised of an IF part which enumerates the conditions and a THEN part which shows the hypothesis name. Right-hand side actions are not depicted in the Current Rule window.
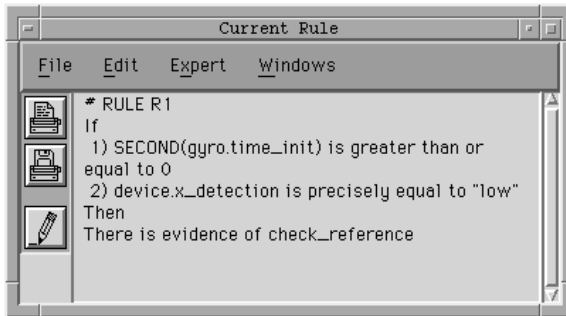
Figure 5-4   Sample Current Rule Window

Note: Conducting an inferencing session with the Current Rule window enabled may result in a slower processing speed.

The Current Rule window is available from the Reports menu either before initiating processing or during a session. Use the following procedure to view a dynamic trace of the system's changing focus of attention.

To enable the Current Rule window:

1.   Select the Show Current Rule option on the Reports menu. The system opens the Current Rule window.

2.   Click on the write enable icon. The system enables the window.

   If the window is already enabled, from the Set Up Environment window for example, the write disabled icon appears in place of write enabled icon.
   If the window is already enabled, from the Set Up Environment window for example, the Disable Write option appears in place of Enable Write on the local popup menu.

3.   Initiate processing and reposition the windows as desired. The system shows the rule currently under evaluation in the Current Rule window.

   If the window updates too quickly to read, you can place breakpoints on the network windows to slow the inference engine.

Another approach you can use to identify the current rule is available through the main window session control panel. When the system interrupts the session to solicit end user input, you can select the Current Rule (or Current Object) option from the window's popup menu. Once

selected, the system displays the network window and focuses on the rule (or object) currently under investigation. The advantage of this option is that you can browse the network diagram to investigate the past and anticipate the future. Dynamic investigation of the network windows is described in the next section of this chapter.

Conclusions

When the system's focus of attention turns to a new hypothesis, it seeks the data needed to complete the evaluation process. To view the effect the known data (specified in rule conditions) has on its hypothesis, you can display the Conclusions window. It shows the value the inference engine determines for the current hypothesis. In this sense, the window gives a trace of the inductions and deductions as they occur. One of the following messages results with respect to hypothesis H.

H is confirmed.	Indicates that the system found at least one true rule after evaluating all the rules leading to the shared hypothesis.
H is rejected.	Indicates that the system found no true rules after evaluating all the rules leading to the shared hypothesis.

In order for one of these messages to appear in the Conclusions window, the system must have completed the evaluation of all rules leading to the shared hypothesis and completed any right-hand side (RHS) actions. Either of these situations may alter the final outcome of the hypothesis. Therefore to help you track the inference engine during the evaluation process, the window reports several interim messages as follows.

Rule x is fired.	Indicates the conditions of H were all true. The system is about to initiate any RHS actions defined in the rule, but has not yet determined the hypothesis' value.
H is already known as true.	
	Indicates H will be true if the evaluation of the remaining rules of the shared hypothesis do not change the outcome and the RHS actions of the current rule do not change the outcome.

Figure 5-5 shows the window as it might look during an inferencing session. The messages that appear at the top of the window are the most recent.
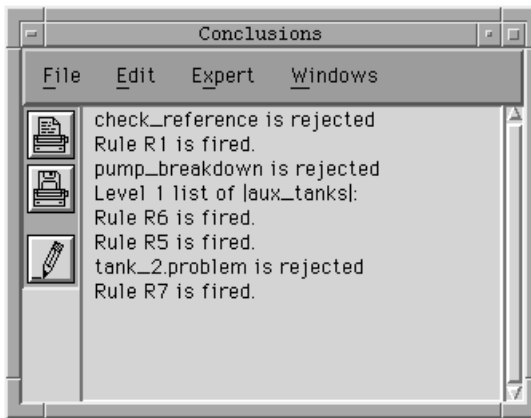
Figure 5-5   Sample Conclusions Window

The Conclusions window is available from the Reports menu either before initiating processing or during a session. Use the following procedure to obtain a dynamic trace of the system's inductions and deductions.

To enable the Conclusions window:

1.   Select the Show Conclusions option on the Reports menu. The system opens the Conclusions window.

2.   Click on the write enable icon. The system enables the window.

     If the window is already enabled, from the Set Up Environment window for example, the write disabled icon appears in place of write enabled icon.

3.   Initiate processing and reposition the windows as desired. The system shows the evaluation conclusions in the Conclusions window.

     If the window updates too quickly to read, you can place breakpoints on the network windows to slow the inference engine.

How and Why

When your application requires input from the end user the system temporarily ceases inferencing and displays a question in the main window. The session control panel lets the end user provide data for the application when the system has no other way to obtain the input. The session control panel, therefore, reflects the system's current focus of attention. To aid in understanding the rationale for the requested data, you (or the end user) can

display the on-line explanation facility by selecting the Why option on the session control panel popup menu.

The explanation facility consists of a dialog window that lets you browse the backward chaining links that lead to the current focus of attention. The dialog window lets you see both "how" the system arrived at its current focus and "why" the data is need for the current focus. Two buttons in the dialog window correspond to these options. When you select the Why button, the explanation changes to reflect the hypothesis and conditions of the next rule in the backward chaining links. When you select the How button, the explanation changes to reflect the hypothesis and conditions of the previous rule in the backward chaining links.

> **Note:** The text that normally appears in the dialog window follows certain syntactic conventions. If desired, you can customize the explanation facility as described in Chapter Six, "Application Documentation."

When the system cannot determine a value for processing it displays the question in the session control panel. Use the following procedure to view the rationale for the system's current focus of attention from the session control panel.

To display and browse the explanation facility:

1. Position the cursor in the session control panel to obtain the ▤ cursor and press the mouse button. The system displays the panel's popup menu.

2. Select the Why option from the list. The system displays the explanation dialog window. The top box gives information about the hypothesis and the bottom box gives information about the LHS conditions.

3. The explanation dialog window shows two buttons that let you browse the backward chaining links from the current rule. Select the **How** button to browse the links in the backward direction. Select the **Why** button to browse the links in the forward direction.

4. Click on the **OK** button from the explanation dialog window. The system closes the explanation dialog window.

Inferencing Event Logs

In the process of finding the value of hypotheses, the Rules Element initiates a wide range of inferencing events. These events contribute to the determination of the hypotheses' status. To help you understand the

system's conclusions, the Rules Element lets you record inferencing events in several windows.

The following sections describe these facilities that record inferencing events. These facilities differ from the ones described in the Focus of Attention section because they capture and retain the information rather than show only current data. This makes the following facilities particularly useful as inferencing session documentation. Refer to Chapter Six, "Application Documentation" for more information about documentation issues.

Transcript

The Transcript window is a useful window for application testing because it combines the focus of attention information described in the previous section with details about the inferencing events. The Transcript window keeps a record of the entire consultation including volunteered data, suggested hypotheses, subgoal hypotheses, and input and modified data. This window displays the following messages.

Data x is set to value y.
> Indicates that the value shown for the data item will be used by the system for processing. The system can fetch the data from within your application, through inheritance, direct assignment, etc. or the system can obtain the value from an external source such as a database, external program, or end user input.

Condition xyz in rule #. (True/False/NotKnown)
> Indicates that the system evaluated the condition using the value of its data and assigned the value `True`, `False`, or `NotKnown` to the condition.

Condition there is/is no evidence of hypo in rule #.
> (True/False/NotKnown)
> Indicates that the system evaluated the conditions of a hypothesis that is itself a condition of a rule (subgoal hypothesis) and assigned the value `True`, `False`, or `NotKnown` to the subgoal hypothesis.

Rule # is set to true/false/notknown.
> Indicates that the system evaluated the rule's conditions and assigned the value `True`, `False`, or `NotKnown` to the rule.

RHS: action xyz in rule #.
> Indicates that the system initiated the right-hand side action of a particular rule.

Hypo is set to True/False/NotKnown.
> Indicates that the system evaluated rules leading to the terminal hypothesis and assigned the value `True`, `False`, or `NotKnown` to the hypothesis.

Figure 5-6 shows the window as it might look during an inferencing session. The messages that appear at the bottom of the window are the most recent.
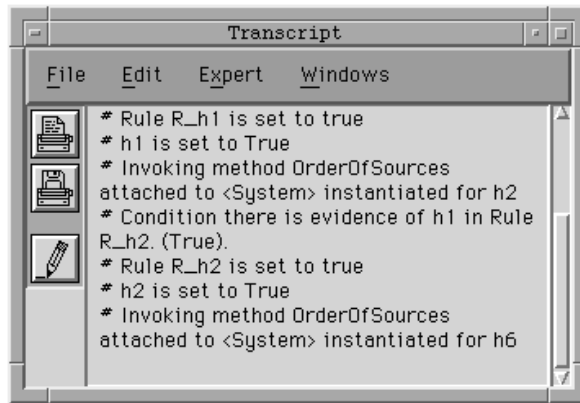
Figure 5-6   Sample Transcript Window

The Transcript window is available from the Reports menu either before initiating processing or during a session. Use the following procedure to obtain a dynamic trace of all inference engine events.

To enable the Transcript window:

1. Select the Show Transcript option on the Reports menu. The system opens the Transcript window.

2. Click on the write enable icon. The system enables the window.

    If the window is already enabled, from the Set Up Environment window for example, the write disabled icon appears in place of write enabled icon.

3. Initiate processing and reposition the windows as desired. The system records the inferencing events in the Transcript window.

    If the transcript is too large to be useful you can place breakpoints on two rules in the network diagram and obtain a limited transcript for the area between the breakpoints. At the first pause (initial breakpoint), write enable the Transcript window and then disable the Transcript window at the second pause (final breakpoint).

Full Report

The Full Report window lets you understand how the system assigned a value to each evaluated hypothesis. Unlike the List of Hypotheses window, the Full Report window keeps a record of which rule conditions were true,

false, or notknown thus providing evidence for the outcome of evaluated hypotheses.

Many rules share the same hypothesis in the application. If a hypothesis has multiple rules leading to it, each rule is joined by an "OR." The system evaluates hypotheses according to the following Full Report message descriptions with regard to hypothesis H.

Hypothesis H was established.
> Indicates that the system evaluated the rules leading to the hypothesis and assigned the value `True` to at least one of the rules. (It disregards any other relevant rule values.)

Hypothesis H was rejected.
> Indicates that the system evaluated the rules leading to the hypothesis and assigned the value `False` to each one.

Hypothesis H is undetermined.
> Indicates that the system evaluated the rules leading to the hypothesis and assigned the value `NotKnown` to at least one rule and `False` to all others (none were true).

Following each of these hypothesis messages, the Full Report window presents the evidence for the conclusion. The window organizes the evidence into two types: Counter Arguments for false hypotheses and Suggestive Evidence for true or notknown hypotheses. Figure 5-7 shows the Full Report window as it might look during an inferencing session.

The system numbers the Full Report window (in the title bar) because it is a "snapshot" in time of the internal state of all evaluated hypotheses. You can display as many of these snapshots as desired and thus simultaneously view multiple versions of the same window. This situation may occur, for example, when you want to modify data and continue processing. You might first display the Full Report window to show the current state, modify the session, and display another Full Report window after initiating

processing. Use the following procedure to obtain a record of the system's justification for all deductions and inductions.
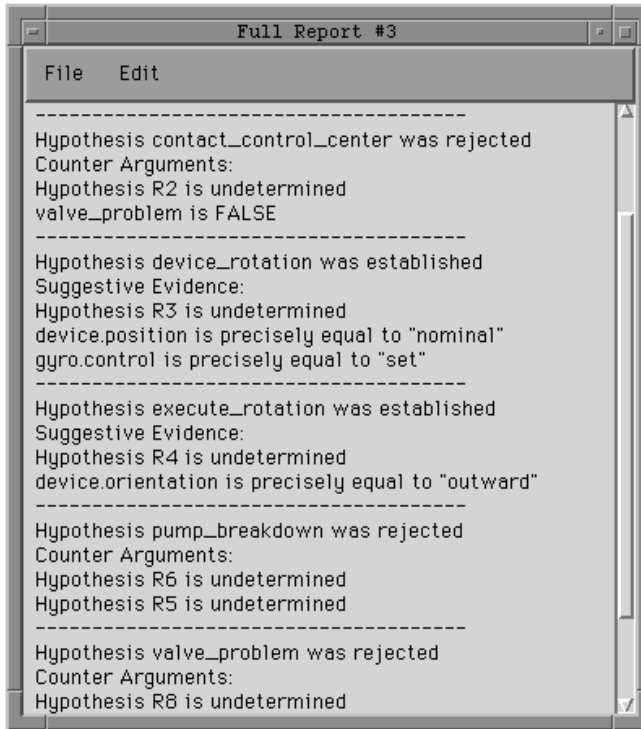
Figure 5-7 Sample Full Report Window

To display the Full Report window:

1. Initiate processing.

2. Select the Full Report option on the Reports menu. The system opens the Full Report window that shows the current state of all evaluated hypotheses.

3. If you want to view the window while continuing processing, reposition the window as desired and continue. Otherwise you can close the window.

   To write the contents of the Full Report window to a file, click on the write to file icon.

4. After continuing processing you can view an updated version of the Full Report window. Select the Full Report option on the Reports menu. The system opens another Full Report window.

To distinguish between different versions of the Full Report window you view, the system numbers them consecutively.

Case Status

During an inferencing session, you may want to review data you've supplied for processing. The Case Status window lets you conveniently view known data and the conclusions the system has drawn from them. Unlike the List of Data and List of Hypotheses windows, the Case Status window only shows known information. The window organizes the information into two groups: Relevant Data and Relevant Hypotheses. Figure 5-8 shows the Case Status window as it might look during an inferencing session.


```
                        Case Status #1

    File    Edit

   RELEVANT DATA
   Data                     Value
   auxiliary_tank_1.Name   auxiliary_tank_1
   auxiliary_tank_2.Name   auxiliary_tank_2
   current_task            defueling
   device.x_detection      low
   device.position         nominal
   device.orientation       outward
   gyro.z                  10:12:10
   gyro.y                  20:30:12
   gyro.x                  10:10:10
   gyro.time_init          15:25:47
   gyro.definition         10:10:1020:30:1210:12:10
   gyro.control            set
   tank_1.pressure         35.0
   tank_1.Name             tank_1
   tank_2.Name             tank_2
   valve_problem           False


   RELEVANT HYPOTHESES AND CONCLUSIONS
   Hypothesis               Status
   check_reference          True
   contact_control_center   False
   device_rotation          True
   execute_rotation         True
   pump_breakdown           False
   valve_problem            False
```

Figure 5-8   Sample Case Status Window

The system numbers the Case Status window (in the title bar) because it is a "snapshot" in time of the internal state of all known data. You can display as many of these snapshots as desired and thus simultaneously view multiple versions of the same window. This situation may occur, for

example, when you want to modify data and continue processing. You might first display the Case Status window to show the current state, modify the session, and display another Case Status window after initiating processing. Use the following procedure to obtain a record of all known data.

To display the Case Status window:

1. Initiate processing.

2. Select the Case Status option on the Reports menu. The system opens the Case Status window that shows the current state of all known data.

3. If you want to view the window while continuing processing, reposition the window as desired and continue. Otherwise you can close the window.

   To write the contents of the Case Status window to a file, click on the write to file icon.

4. After continuing processing you can view an updated version of the Case Status window. Select the Case Status option on the Reports menu. The system opens another Case Status window.

   To distinguish between different versions of the Case Status window you view, the system numbers them consecutively.

Inferencing Event Investigation

While monitoring your application's reasoning pathways during a session (as described in the previous section), you may have encountered situations where the application failed to reach the expected conclusions. The source of this type of behavior is three-fold:

■ Unforeseen agenda control mechanisms

■ Insufficient information to reach the goal

■ Conflicting rule and object structures.

To help you identify the cause of unexpected behavior, the Rules Element shell supplies facilities that let you investigate processing interactively. These facilities, described in the following sections, work together to let you "debug" your application. After identifying possible sources of the unexpected behavior, you can make alterations using the editor windows

and immediately repeat application testing. Figure 5-9 shows the interactive nature of the Rules Element shell when developing applications.

```
┌──────────┐     ┌──────────┐     ┌──────────┐     ┌──────────┐
│  Design  │ ──▶ │Implement │ ──▶ │   Test   │ ──▶ │   Edit   │
└──────────┘     └──────────┘     └──────────┘     └──────────┘
```
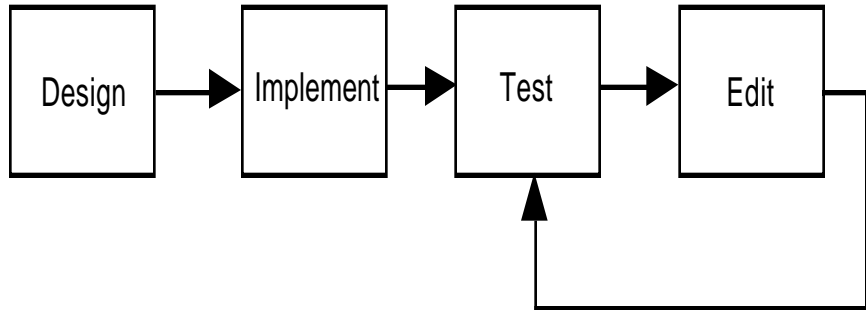
Figure 5-9   Application Testing Cycle

Perhaps the most versatile feature of the Rules Element shell is the graphic network windows – one supplied for the rule network and one for the object network. If you are not already familiar with these windows, refer first to Chapter Three, "Application Editing" for a functional description. The following sections build on that information by introducing the concept of "dynamic investigation." The task of expanding the network remains the same as described in Chapter Three, but now the investigation takes place during processing.

The dynamic investigation of the network windows is possible because the system updates the status of these windows during the session automatically. Information that you can obtain interactively from the network diagrams includes the following.

■   The hypotheses currently under investigation (Rule Network)

■   The evaluation status of individual conditions, actions, rules, and hypotheses (Rule Network)

■   The interpreted value of the Priority Slot attached to individual rules (Rule Network)

■   The data values of individual slots (Object Network)

■   The interpreted value of the Priority Slot attached to individual slots (Object Network)

■   The creation of dynamic classes, objects, and properties (Object Network)

■   The list of each method's conditions and actions (Object Network).

The information listed above is only available during a processing session. It lets you, for example, easily determine the outcome of hypothesis evaluation as the system progresses through the reasoning pathways. In

general the listed features expand your level of interaction with the system so you can now investigate the unexpected behavior of your application. The following sections give more detailed information about using the windows interactively.

Network Breakpoints

In the "Tracking Inferencing Events" section you were able to predict the inferencing behavior of your application through the monitoring facilities. Maybe you identified an area of the application that did not perform as you expected it would, and now you want to determine the inferencing links that are responsible.

In order to conduct a dynamic investigation of the rule network, the first requirement is to be able to stop and start the inference engine when desired. The Rules Element gives you several ways to interrupt the inferencing engine as follows.

■ Click on the Interrupt/Continue button on the main window session control panel.

■ Place "breakpoints" in the rule network to control the inference engine.

■ Use keyboard interrupt commands, for example, Ctrl-Alt on the PC orc Cmnd–. (flower and period keys together) on the Macintosh.

The best approach is the second, placing breakpoints directly on the network diagram where you want the inference engine to pause.

The breakpoint icon (●) resembles a stop sign and is selected by clicking on its corresponding window icon. Figure 5-11 shows a breakpoint on the first condition of rule number 6. Actually the system lets you place breakpoints on classes, objects, properties, slots, rule conditions/actions, rule name, hypotheses, method conditions/actions, and method names – in short, anywhere in the network diagrams.

The drawback of using breakpoints is that they require some advanced planning because they stop the inference engine after evaluation. In this sense the other approaches give you a more spontaneous way of interrupting the session. Of course, there is no reason you could not place breakpoints on every rule in the network and then initiate processing. Though to avoid having to do this, use the inferencing tracking facilities as described in the "Tracking Inferencing Events" section. The goal is to isolate the problem and begin the dynamic investigation around that area of the network diagram.

Rule Network Breakpoints

You can place breakpoints in the rule network diagram on any item desired including conditions, actions, rule name, and hypotheses. However, the position of the breakpoint in a rule node (a node is the complete rule) is important because it determines when the evaluation of the rule is interrupted and therefore what data is known. The inference engine pauses for breakpoints that you place on the rule network as follows.

■ On a condition or action, *after* the system evaluates it.

■ On a rule name, *after* the system fires the rule.

■ On a hypothesis, *after* the system evaluates it.

Use the following procedure to place breakpoints on the rule network diagram.

To insert breakpoints:

1. Click on the breakpoint button shown in the Rule Network window palette. The cursor changes to .

2. Position the cursor over the rule network item that you want to place the breakpoint on and click. The system displays the icon on the selected item.

Use the following procedure to clear a breakpoint that has been set in the rule network diagram.

To remove breakpoints:

1. Click on the breakpoint button shown in the Rule Network window palette. The cursor changes to .

2. Move the cursor over the breakpoint you want to remove and click. The system removes the icon from the network diagram.

3. If you want to remove *all* breakpoints from the rule network diagram, double click on the breakpoint button. The system automatically displays a confirmation dialog window. Click on the **OK** button in the confirmation window to proceed. The system redisplays the rule network diagram with all breakpoints removed.

   Click on the Cancel button to abort the operation.

Object Network Breakpoints

You can place breakpoints in the object network diagram on any item desired including classes, objects, properties, slots, method names, and

individual conditions or actions of methods. However, the position of the breakpoint in an object network is important because it interrupts the evaluation of any rule that contains the object structure. The inference engine pauses for breakpoints that you place on the object network as follows.

■ On a class or object, *after* it is used.
■ On a property, *after* the value of any of the slots changes, for all slots related to the property.
■ On a slot, *after* the value of the slot changes.
■ On a method name, *after* the system completes the entire method.
■ On a method condition, *after* the system completes the condition.
■ On a method action, *after* the system completes the action.

Use the following procedure to place breakpoints on the object network diagram.

To insert breakpoints:

1. Click on breakpoint button in the Object Network window. The cursor changes to ⬤.

2. Position the cursor over the object network item you want to place the breakpoint on and click. The system displays the ⬤ icon on the selected item.

Use the following procedure to clear a breakpoint that has been set in the object network diagram.

To remove breakpoints:

1. Click on the breakpoint button in the Object Network window. The cursor changes to ⬤.

2. Move the cursor over the breakpoint you want to remove and click. The system removes the ⬤ icon from the network diagram.

3. If you want to remove *all* breakpoints from the object network diagram, double click on the breakpoint button. The system automatically displays a confirmation dialog window. Click on the **OK** button in the confirmation window to proceed. The system redisplays the object network diagram with all breakpoints removed.

    Click **Cancel** to abort the operation.

Object Network Method Filters

A breakpoint placed on a method, as described in the previous section, does not isolate the effect of the method on a single slot, property, object, or class. If a breakpoint is used on a method, the inference engine will break for each slot, property, object, or class that triggers the method. This may not be desirable if you want to isolate the effect of the method on a single item. To achieve a finer level of granularity a "method filter" can be combined with the breakpoint you place on the method. The selection window shown in Figure 5-10 lets you define the filter.

The column on the left (see Figure 5-10) identifies the complete list of methods in your application; the middle column changes for the method selection to display the attached to slot, object, class, or property; and the right column changes to display the list of atoms that will potentially inherit the method selected in the left column. Use the following procedure to define a method filter for a method displayed on the object network.
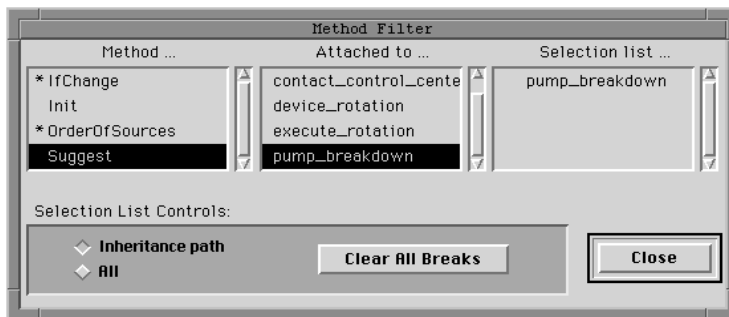
Figure 5-10   Method Filter Selection Dialog

To define a method filter:

1. Display the Object Network window and select the Options... command on the Object menu from the menu bar. The system displays a small dialog window. Click on the Methods checkbox to include methods in the network expansion. Expand the object network diagram.

2. Display the local popup menu for the method that you want to filter and select the Method Filter option. The system displays the method filter selection window with the name of the method highlighted in the left column.

   The local popup menu can be displayed for a method name, action, or condition displayed in the object network diagram.

3. If necessary, click on the name of the slot, object, class, or property shown in the middle Attached To column that goes with the method name highlighted in the left column.

4. Click on the object structures named in the right column that you want the inference engine to pause on after the method is triggered. The system places a filter icon next to the name (clicking on the item name again removes the filter).

   By default the system displays only those object structures that can inherit the method name highlighted in the left column. If desired, select the All checkbox to view all possible object structures in the application.

5. Click on the **Close** button to proceed. The system closes the dialog window and redisplays the object network diagram.

The method filter that you define in the method filter selection dialog can be placed on the object network by clicking on the method item for which the filter is defined. The method filter icon (**F**) t hat you place on the method resembles a regular breakpoint icon with an "F" inside. The inference engine pauses for method filters that you place on the object network as follows.

■ On a method name, *after* the system triggers the method for the item(s) selected in the filter dialog.

■ On a method condition, *after* the system triggers the condition for the item(s) selected in the filter dialog.

■ On a method action, *after* the system triggers the action for the item(s) selected in the filter dialog.

Use the following procedure to place method filters on the object network diagram.

To insert method filters:

1. Display the object network with the method name and define the filter as described in the previous procedure.

   To view methods in the object network, select the Options... command on the Object menu from the menu bar and click on the Methods checkbox.

2. Click on breakpoint button in the Object Network window. The cursor changes to (●).

3. Position the cursor over the method item you want to place the method filter on and click. The system displays the (F) icon on the selected item.

The local popup menu can be displayed for a method name, action, or condition displayed in the object network diagram.

Once a method filter has been specified for a particular method in the method filter selection dialog, the filter can be removed and reinserted on the object network as often as needed. Use the following procedure to temporarily remove a method filter icon from the object network diagram. This operation removes the breakpoint, but does not clear the method filter specified in the method filter selection dialog.

Note: After removing a method filter icon from the object network, you can reinsert the icon as long as the settings in the method filter selection dialog have not been cleared.

To temporarily remove method filter icons from the object network:

1. Click on the breakpoint button in the Object Network window. The cursor changes to (●).

2. Move the cursor over the method filter you want to remove and click. The system removes the (F) icon from the network diagram.

3. If you want to remove *all* breakpoints from the object network diagram, double click on the breakpoint button. The system automatically displays a confirmation dialog window. Click on the **OK** button in the confirmation window to proceed. The system redisplays the object network diagram with all breakpoints removed.

Click on the Cancel button to abort the operation.

Use the following procedure to clear the method filter settings specified in the method filter selection dialog. Once this operation is complete, the method filter must be redefined before it can be used in the object network.

To clear the method filter definition:

1. Display the local popup menu for the method whose filter you want to clear and select the Method Filter option. The system displays the method filter selection window with the name of the method highlighted in the left column and the filtered items in the right column.

2. Click on the Clear All Breakpoints button. The system eliminates the filters from the items in the selection list.

Individual filters can be eliminated by clicking on the filtered item name in the right column.

3. Click on the Close button and return to the Object Network. The system automatically eliminates the method filter but leaves the breakpoint setting in the object network diagram.

To remove the breakpoint setting, click on the breakpoint button and click on the breakpoint icon in the object network.

Rule Network Dynamics

The Rule Network window is useful for conducting dynamic investigations because it can graphically depict strong and weak links between rules. These inferencing links include LHS gates, RHS action affects, and context links on hypotheses (as explained in the Intelligent Rules Element Language Programmer's Guide). The goal of the investigation is to identify these links and explore the resulting reasoning pathways by using the forward and backward chaining investigation approaches. The Rule Network window supplies the left and right arrow icons for this purpose as shown in Figure 5-11 below (refer to Chapter Three, "Application Editing" for more information).

Note: The Rule Network window is unable to depict certain inferencing links. For instance, the network cannot show forward chaining from data that is part of an expression. Also the network diagram cannot show the effects of hypotheses evoked in methods. To identify these particular links, use the Agenda Monitor as explained later in this chapter.
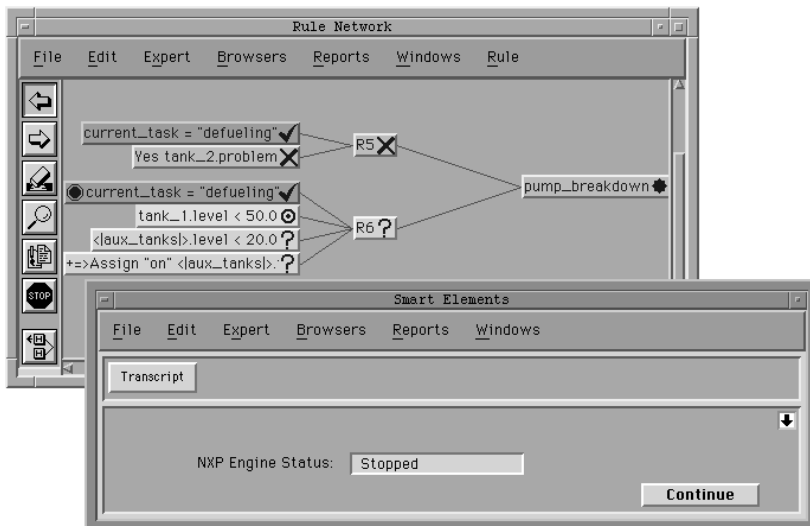
Figure 5-11 Dynamic Rule Network Investigation

The rule network diagram shown in Figure 5-11 is the result of a dynamic investigation because it shows actual processing results. For instance, the major difference between this diagram and the one shown in Chapter Three (Figure 3-2) is the presence of the evaluation icons. Other icons give additional information and are also present only during dynamic investigation. Table 5-7 explains these graphic symbols or network icons that the system uses to represent the evaluation status of individual conditions, actions, rules, and hypotheses.

Network Icon	Description
?	`Unknown`, the initial state and not yet evaluated
✗	`False`, system evaluated and test failed.
✓	`True`, system evaluated and test passed.
▭	`NotKnown`, system evaluated and insufficient data to test.
✦	Evoked hypotheses, hypotheses currently under evaluation.
◉	Current evaluation, condition or action under evaluation.
◉	Breakpoint, allowed throughout network diagram.

Table 5-7 Rule Network Icon Descriptions

The goal of the dynamic investigation is to determine where to pause the inference engine and begin identifying the inferencing links. Use the following procedure to conduct the dynamic investigation in the Rule Network window.

To investigate the rule network dynamically:

1. Optionally, enable the Journal facility in order to record the data entry steps as desired.

2. Initiate processing in order to obtain a session trace, for instance a printout of the Transcript window, and identify the rules that deviated from the expected inferencing behavior.

   If you only have the rule number from the transcript, you can cross-reference the hypothesis name in the List of Rules window.

3. Display the Change Rule Settings dialog window and enable Use Icons, enable Show Priorities, and change the small box length to 70 (from 50). Close the dialog window.

4. Display the Rule Network window and display as much of the network diagram as needed by using the backward and forward approaches to evoking the network.

5. Place strategically positioned breakpoints on the rule numbers of a few rules. These breakpoints let you enable and disable the transcript and return to the network diagram during processing.

6. Initiate processing (if available, use the journal file and replay it "step by step"). The system will pause at the first breakpoint.

7. Return to the network window and select the Focus on Hypothesis option from the global popup menu. The system displays the list of hypotheses. Locate the previously identified hypothesis and double-click on the hypothesis name. The system displays the rule node in the Rule Network window.

8. Place a breakpoint on the first condition of the focused on rule and continue processing. The system will pause on this or another breakpoint.

9. Essentially, the dynamic investigation begins when the system evokes the focused on hypothesis (represented in the network diagram with a star symbol). This indicates that some other rule's data or hypothesis caused this hypothesis to become evaluated by the system. Return to the network window.

10. Select the right arrow icon and position the arrow cursor over the first LHS condition of the focused on hypothesis (the currently evoked hypothesis) and click. The system displays the hypotheses that contain data in common (or "gates") with the condition. Repeat the procedure for each LHS condition.

    Using the right arrow cursor on RHS actions only identifies the hypotheses of rules that become evaluated due to forward action effects. Right-hand side actions cannot be the recipient of forward chaining from data.

11. Now you can select the left arrow icon and position the arrow cursor over one of the elicited hypotheses from the focused on hypothesis and click. The system displays the rule nodes that share the hypothesis.

12. Scroll the network window in order to view the conditions of these rules. The objective is to identify which data in the network diagram caused the gate.

Certain gates cannot be elicited in the network diagram, such as data that appears in a meta-slot or expression. To identify these particular inferencing links, you must use the agenda monitor.

13. Repeat the procedure for each elicited hypothesis from the original focused on hypothesis until you are satisfied with the presence of the focused on hypothesis in the reasoning pathway.

If you identify unwarranted gates, you can use the Strategy operator to restrict them from forwarding. Refer to the Intelligent Rules Element Language Reference manual for more information about this operator's usage.

Object Network Dynamics

The Object Network window is useful for conducting dynamic investigations because it can show the effect of processing at the data level. The system can obtain a slot's value (data) during processing from inheritance, database retrieval (dynamic object creation), and direct assignment through rules and methods. The goal of the investigation is to identify how and frequently when the system determined the value of a particular slot. The Object Network window supplies the left and right arrow icons for this purpose as shown in Figure 5-12 below (refer to Chapter Three, "Application Editing" for more information).

Note: The Object Network window can also help you identify data that causes inferencing links between rules. This is particularly true for data that appears in methods since the Rule Network window is unable to depict the methods attached to a particular slot.
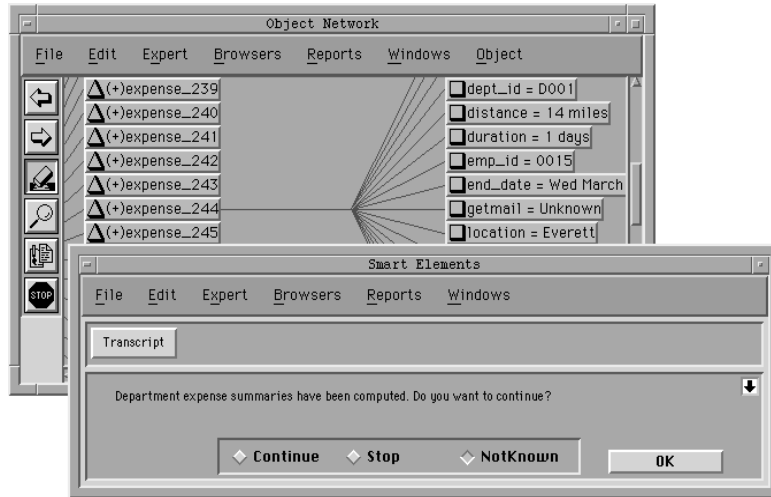
Figure 5-12   Dynamic Object Network Investigation

The object network diagram shown in Figure 5-12 is the result of a dynamic investigation because it shows actual processing results at the data level. For example, the slot expenses_244.distance has a value of 14 miles associated with it. When an object network diagram is displayed prior to initiating processing, as shown in Chapter Three (Figure 3-4), it lacks this additional information. One possible source of the value can be downward inheritance from the parent class. To help you better recognize the parent/child inheritance relationships, the Object Network window can depict object structures with a combination of graphic symbols and user-defined fonts. Table 5-8 explains these graphic symbols or network

icons that the system uses to represent the object structures. To change these structures' fonts refer to Appendix D, "Customizing the Environment."

Network Icon	Description
○	Class structures.
△ or △ (+)	Object structures (with plus sign if created dynamically).
□ or ■	Properties (filled-in only if meta-slot attributes defined, with "P" if slot property belongs to is specified as private).
◉ or ⑤	Breakpoints (with filter if created for methods).
◇ or ◈P	Public methods (with "P" if specified as private).
=>, +=>, or -=>	Conditions, Then Do, and Else Do actions of a method.
▽	Data validation defined on a meta-slot or property.

Table 5-8   Object Network Icon Descriptions

The dynamic investigation procedure you follow in the Object Network window is similar in nature to the one described for the Rule Network. In either case you attempt to discover a cause and effect relationship by eliminating possibilities. In the case of the rule network diagram you might view several possible reasoning pathways to find out which one produced an unexpected inferencing link, whereas you might view several possible inheritance pathways in the object network diagram to find out which one supplied an unexpected data value to a particular slot. The object network investigation is somewhat easier because you already know where to place breakpoints since the data for a known slot is in question.

A complete list of possible object network investigations follows. Each type of investigation entails placing breakpoints on specific object structures in order to pause the inferencing process. Afterwards you can expand the object network diagram, the rule network diagram, or both in combination.

Inheritance   1) Whether a slot received its value through inheritance and 2) if so which pathway supplied the value, and 3) which methods are inherited.

Values   1) At what point during processing a particular slot value was determined and 2) whether the data value is correct.

Objects   1) At what point during processing objects are created either through a database retrieve or a CreateObject operator after using pattern matching and 2) whether the objects are linked to the correct class or object.

Methods without Filter
                         1) At what point during processing are methods triggered and 2) what inferencing links the statements in the methods may produce (requires use of the Cross Reference window or Agenda Monitor).

Methods with Filter
                         At what point during processing are methods triggered by individual slots, objects, or classes. See the "Object Network Method Filters" section for details about defining breakpoints that filter individual slots.

Meta-Slots   At what point during processing are individual meta-slots inherited.

Use the following general procedure to conduct any one of the above dynamic investigations in the Object Network window.

To investigate the object network dynamically:

1. Display the Change Object Settings dialog window and enable Use Icons and Show Values. Close the dialog window.

2. Display the Object Network window and display as much of the network diagram as needed by using the two arrow icons to evoke the inheritance pathways of the network.

3. Place a breakpoint on the desired object or property. This breakpoint lets you determine exactly when the value of any of the slots changes and return to the network diagram during processing.

   If you are not sure which slot to place the breakpoint on, you can place breakpoints at the class level as explained in the previous section on using breakpoints in the Object Network window. To view dynamically created objects in the Object Network window and their links to a particular class, place the breakpoints in the rule network diagram just before and just after the Retrieve or CreateObject operator.

4. Initiate processing. Return to the network window after the system pauses at the breakpoint.

5. Select the Left Arrow icon and position the arrow cursor over the property of the slot you are investigating and click. If the property is related to other object structures, the network diagram expands to show the set of slots that include the selected property.

6. Examine the newly displayed slots for the presence of the "filled-square" icon indicating meta-slot attributes or a "diamond" icon indicating methods. Click on the Right Arrow icon and expand the network for the displayed icons.

7. To identify whether the property of the slot you are investigating appears as data in a rule, you can display the local popup menu for the property. Select the Focus on Rule Network option if available to view the data in the Rule Network window (if not present, the slot is not used in a rule). The diagram that the system displays identifies the hypotheses whose rules contain the slot. You can expand the resulting rule network diagram as desired.

Advanced Inferencing Event Investigation

The Agenda Monitor, as it is referred to, provides a seemingly basic interface and yet provides a very important function. This window lets you directly observe how the inference engine processes the hypotheses of your application with respect to each other. The Agenda Monitor window therefore complements the dynamic investigation function of the network windows. The complete list of inference events the Agenda Monitor depicts during processing follows.

■   Changes to the status of the current strategies.

■   Pre-evaluation queuing of hypotheses on the agenda according to inferencing mechanism categories.

■   Dequeuing of hypotheses for evaluation by inferencing mechanism categories and hypothesis priority numbers.

■   Listing of currently evoked hypotheses for evaluation.

■   Listing of the current focus of evaluation, including the LHS conditions and RHS actions of rules and methods.

Note: Familiarity with the concepts expressed in the Intelligent Rules Element Language Programmer's Guide can improve your application debugging skills when using the Agenda Monitor.

Figure 5-13 shows the Agenda Monitor as it appears before initiating processing.
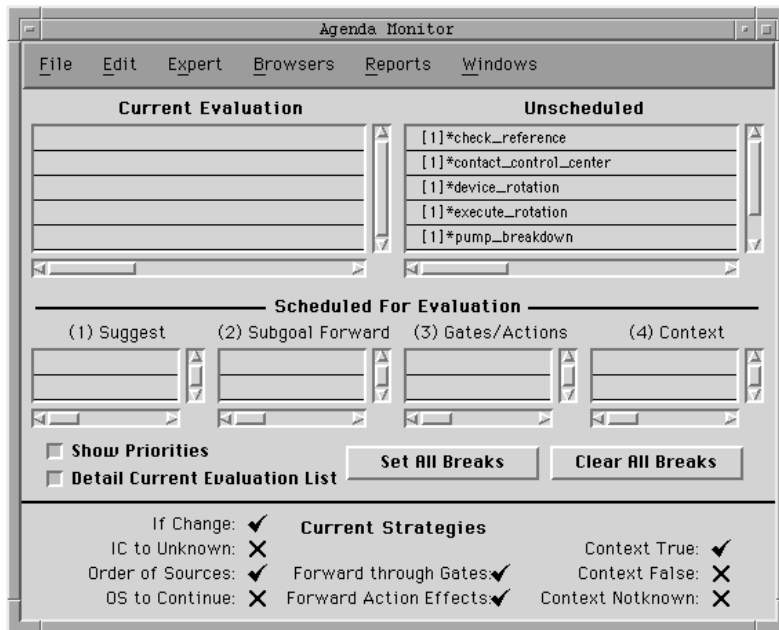
Figure 5-13   Agenda Monitor Window

Use the following procedure to initiate processing from the Agenda Monitor window.

To initiate processing:

1.   Select the Agenda Monitor option on the Expert menu. The system opens the Agenda Monitor window with all the hypotheses displayed in the Unscheduled column.

2.   Locate the desired hypotheses from the list that you want to suggest.

     The list may be too long to view entirely. Scroll the list if necessary.

3.   Position the cursor on the hypothesis name and click. The system displays the local popup menu for that hypothesis. Select the Suggest option from the list. The system suggests the hypothesis and closes the popup menu.

     To unsuggest a hypothesis, redisplay the popup menu for that hypothesis and select the Unsuggest option from the list.

4. If you want to initiate processing immediately, display the Windows popup menu and select the Knowcess option from the list. The system initiates the inferencing session using the suggested hypotheses.

   In order to use the Agenda Monitor dynamically to view the processing of hypotheses, use breakpoints to pause the inference engine as described in the following section.

5. If you prefer not to initiate processing yet want to save the current selections, you may continue using the windowing environment.

   To initiate processing after closing the Agenda Monitor window, select the Start With... Knowledge Base option on the Expert menu.

Agenda Breakpoints

The Agenda Monitor reflects actual inferencing mechanism events and can reveal a wealth of information about the reasoning pathways of your application. To view the queuing of hypotheses for evaluation on the Agenda Monitor window, you must place breakpoints on the individual hypotheses.

Breakpoints on hypotheses in the Agenda Monitor provide a similar function to the network diagram breakpoints. Both determine when the evaluation process is interrupted so you can conduct a dynamic investigation. The inference engine pauses for changes to a value that effects a hypothesis on the agenda. The Agenda Monitor shows these changes as one of the following events.

- Pre-evaluation queuing of the hypothesis
- Evoking the hypothesis for evaluation
- Requeuing of the hypothesis on the agenda.

> **Note:** Agenda Monitor breakpoints differ significantly from network diagram breakpoints because they interrupt the inference engine immediately *before* the system initiates one of the events described above. Network diagram breakpoints occur after a value changes.

Use the following procedure to place breakpoints on hypotheses shown in the Agenda Monitor. This procedure also lets you place breakpoints on hypotheses shown in the Current Evaluation column of the Agenda Monitor. If you select the Detail Current Evaluation List option, the system lets you pause on the individual rules, conditions, and methods currently under evaluation by placing breakpoints on the desired list items. See the next section, "Agenda Monitor Investigations," for more information about the individual Current Evaluation column and agenda queues.

To insert breakpoints:

1. If you want to place a breakpoint on one hypothesis at a time, locate the desired hypothesis in one of the agenda queues.

   If you have not yet initiated processing, all hypotheses appear in the Unscheduled column.

2. Position the cursor in the white space just to the left of the hypothesis name and click. The system inserts a greater-than symbol (>) in the space next to the hypothesis name. This symbol represents the breakpoint in the Agenda Monitor.

   The local popup menu for each hypothesis also lets you insert breakpoints. Select Set Breakpoint from the list.

3. If you want to place breakpoints on all hypotheses at once, click on the **Set All Breaks** button.

   This button inserts breakpoints on all hypotheses whether you have initiated processing or not. To remove individual breakpoints, see the following procedure.

Use the following procedure to remove breakpoints on hypotheses shown in the Agenda Monitor.

To remove breakpoints:

1. If you want to remove a breakpoint from one hypothesis at a time, locate the desired hypothesis in one of the agenda queues.

   If you have not yet initiated processing, all hypotheses appear in the Unscheduled column.

2. Position the cursor over the greater-than symbol (>) next to the hypothesis name and click. The system removes the breakpoint.

   The local popup menu for each hypothesis also lets you remove breakpoints. Select Unset Breakpoint from the list.

3. If you want to remove all breakpoints from the Agenda Monitor, click on the **Clear All Breaks** button. The system automatically displays a confirmation dialog window.

   This button removes breakpoints from all hypotheses whether you have initiated processing or not. To insert individual breakpoints, see the previous procedure.

4. Click on the **OK** button in the confirmation window to proceed. The system redisplays the Agenda Monitor with all breakpoints removed.

   Click on the Cancel button to abort the operation.

Agenda Monitor Dynamics

The Agenda Monitor lets you view the current status of your application's hypotheses during an inferencing session. This is particularly important for investigating inferencing behavior because the evaluation of hypotheses drives the Rules Element application. To help you investigate unexpected behavior the Agenda Monitor has six columns that display your hypotheses. Hypotheses do not appear in more than one column at a time; these columns are both exhaustive and mutually exclusive.

Agenda Queues

The five narrow columns (see Figure 3-3) are the pre-evaluation queues or agenda portion of the Agenda Monitor window. The queues appear in order of importance, with the highest priority queue (Suggest) on the left side of the window. The system evaluates hypotheses from these columns in a systematic fashion, always searching the highest priority queues first.

Note: Familiarity with the concepts expressed in the Intelligent Rules Element Language Programmer's Guide can improve your application debugging skills when using the Agenda Monitor.

Figure 5-14 shows the Agenda Monitor as it might appear during an inferencing session.
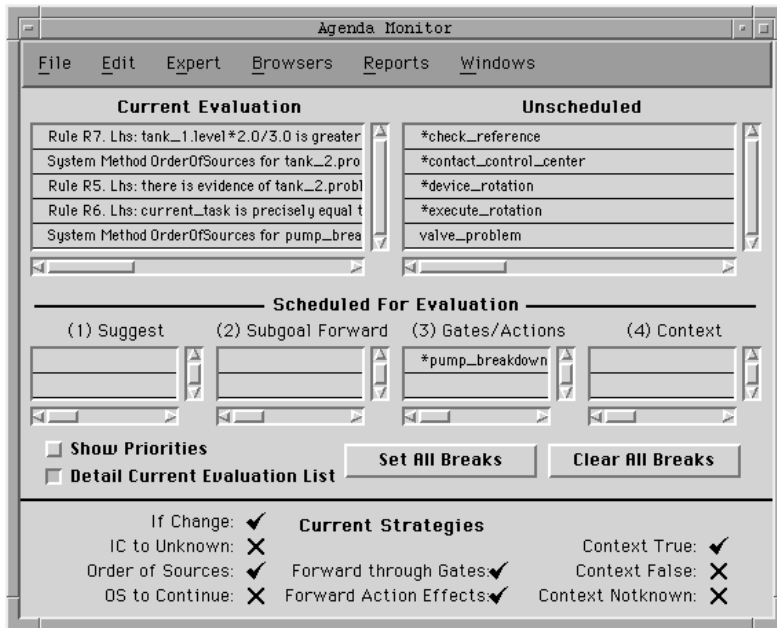
Figure 5-14 Agenda Monitor Investigation

The system prioritizes hypotheses for evaluation during processing as explained below.

Suggest	Contains the list of hypotheses that the user suggested to initiate processing. This is the highest priority pre-evaluation queue.
Subgoal Forward	Contains the list of hypotheses whose left-hand side (LHS) conditions have another already evaluated hypothesis. This is the second highest priority queue, known as subgoal forward.
Gates/Actions	Contains the list of hypotheses whose LHS conditions were found to be TRUE because of known data from other rules. The data can come from another rule's LHS (gates) or RHS (actions). This is the third highest priority queue.
Contexts	Contains the list of hypotheses that have a weak link (context) with another already evaluated hypothesis. This is the lowest priority queue; the system evaluates hypotheses from this queue once all other queues appear empty.

Unscheduled   Contains the list of hypotheses that the system has not yet considered for evaluation. The four agenda queues described above, obtain hypotheses from this list during processing. Before the user initiates processing all hypotheses appear in this list.

Current Evaluation Column

The single column across the top of the Agenda Monitor (see Figure 5-14) identifies all hypotheses actively under evaluation by the system. These hypotheses come directly from the pre-evaluation agenda queues as described above. If there is more than one evoked hypothesis in the Current Evaluation column, the system begins the evaluation from the top of the list and proceeds downward. New hypotheses appear in the list only after the system finishes evaluation of the complete list. In this sense, the list can be thought of as the current backward evaluation combined with any hypotheses evoked from method activity.

The Agenda Monitor lets you toggle the display between two versions of the list as explained below. Like all other Agenda Monitor queues, the Current Evaluation column accepts breakpoints as explained in the previous section, "Agenda Breakpoints."

Current Evaluation   This list shows all hypotheses currently under evaluation by the system. These are the same hypotheses you can identify in the Rule Network window as evoked (look for star symbol), plus any evoked hypotheses from methods (not represented in the network diagram).

Detailed Current Evaluation
                This is the detailed view of the current evaluation; it identifies the rules and conditions, as well as all the methods, that belong to the hypotheses currently under evaluation. As with the hypotheses-only current evaluation list, the system begins the evaluation from the top and proceeds downward.

The goal of the dynamic investigation is to observe the processing of hypotheses in order to determine when one comes up for evaluation unexpectedly. There are only three reasons hypotheses should change from one list to another as follows.

1.   A hypothesis is completely evaluated and is therefore removed from the Current Evaluation column and returned to the Unscheduled column.

2.   When all the hypotheses in the Current Evaluation column have been evaluated, the system refocuses on the hypothesis with the highest priority from the column of highest priority and places it on the Current Evaluation column.

3. When gates, actions, backward chaining, contexts, etc. make hypotheses relevant, the system will put them in the appropriate pre-evaluation queue.

This necessitates finding the connection between the unexpected hypothesis and the hypotheses currently under evaluation. Use the following procedure to conduct the dynamic investigation in the Agenda Monitor window.

To investigate the Rules Element agenda dynamically:

1. Display the Agenda Monitor and click on the **Break on all Hypotheses** button. The system places breakpoints on each hypothesis on the list.

2. Select the **Show Priorities** option on the Agenda Monitor window. The system highlights the checkbox to indicate its selected status.

3. Initiate processing. The system will pause when a hypotheses changes queues.

    Initially this means suggested hypotheses will appear in the Suggest column.

4. Click on the **Continue** button on the session control panel. The system releases the inference engine and continues processing. The system will pause when a hypothesis changes queues again.

    Initially this means the highest priority hypothesis queue will place a hypothesis in the Current Evaluation column.

5. Essentially, the dynamic investigation begins when the system places the hypotheses in the pre-evaluation queues. Observe the order hypotheses enter the queues. If a hypothesis appears on the list before it is expected, that indicates it was evoked by the Current Evaluation.

6. Click on the **Detail Current Evaluation List** button. The system replaces the Current Evaluation column with the more detailed list of all rules, conditions, actions, and methods the system is currently evaluating.

7. Scroll the list of the Current Evaluation column and find rule or object structures that might have queued the unexpected hypothesis.

8. If necessary you can investigate the unexpected hypothesis in the network windows. To display the window directly from the Agenda Monitor, position the cursor on the hypothesis name and click. The system displays the local popup menu for that hypothesis. Select the Focus Rule Network option from the list. The system displays the network diagram of the rule or rules for the selected hypothesis.

9. The goal of the investigation is to find the connection of the focused on hypothesis with the ones currently under evaluation. Use the network window's right arrow icon to elicit hypotheses that contain data in common (or "gates") with the conditions and actions of the focused on hypothesis.

10. Return to the Agenda Monitor window to reexamine the Current Evaluation column for data in common.

    The Current Evaluation column also shows the data and hypotheses that appear in the current methods. The rule network diagram is unable to display this information. You may want to use the Cross Reference window to identify the method associated with the data of the unexpected hypothesis.

11. Continue processing as the system pauses each time the hypotheses change lists and repeat the investigation as desired.

    You may want to reinitiate processing and selectively place breakpoints on the hypotheses as desired. This will prevent the system from pausing unnecessarily.

Controlling Inference Strategies

The Strategy option on the Expert menu displays a monitor window that you can use to control the global behavior of the inference engine. The strategy settings displayed depend on the mode button selected. The Strategy Monitor window displays the settings in the Current mode which reflects changes made to the strategies through rules in the application. Figure 5-15 shows the difference between the default and current settings.

None of these settings need to be changed in order to initiate processing. The Rules Element system's default configuration includes the full range of inference and inheritance strategies.

> **Note:** Changing inference settings is an advanced operation and requires familiarity with the concepts expressed in the Intelligent Rules Element Language Programmer's Guide.
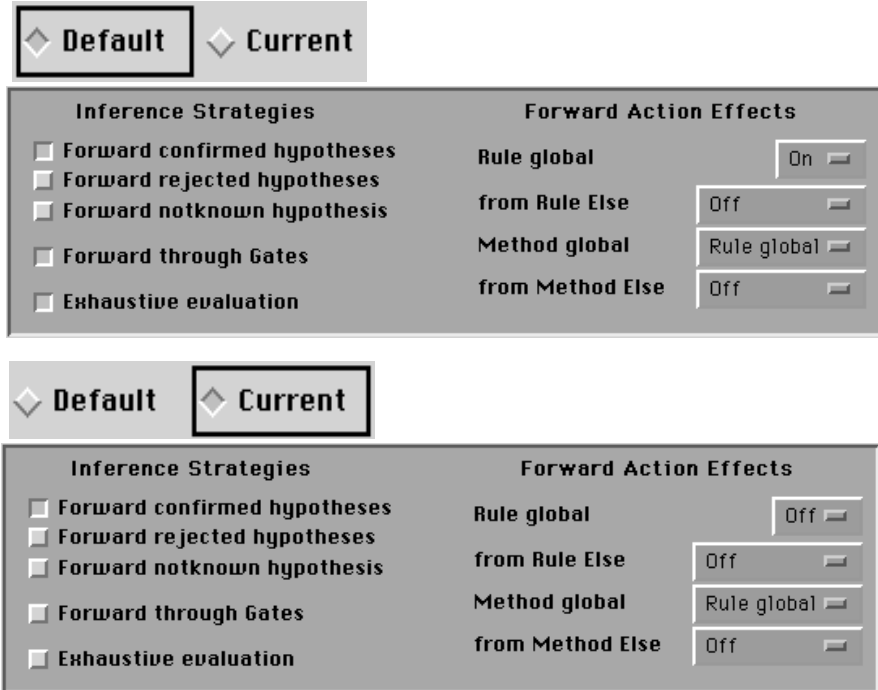
Figure 5-15 Strategy Monitor Inference Selections

Re-configuring the inference strategies is an optional operation. If you do not make adjustments to the default behavior, the system conducts processing with all strategies enabled. However, you can disable combinations of these strategies to simplify the behavior of the inference engine. Use the following procedure, for example, to configure the inferencing session for its most basic mode of operation.

To view and/or modify inference strategies:

1. Select the Strategy option from the Expert menu. The system opens the Strategy Monitor and displays the current status of the inference strategies as affected by processing. Those with highlighted squares are currently enabled.

2. Click on the **Default** button. The system displays the default settings as they appeared at the start of processing.

   If you display the Strategy Monitor before initiating processing, the Current strategy settings are the same as the Default strategy settings.

3. Click on the checkbox that corresponds to the inference strategy you disabled prior to starting processing. The system highlights the square to show its selected status.

    Strategies should be enabled one at a time in order to view their effect on the application during processing.

4. Click on the **OK** button from the Strategy Monitor to compile the new global default settings.

    Click on the Apply button to approve the changes without exiting the Strategy Monitor.

6 *Application Documentation*

This chapter explains how to use the various Intelligent Rules Element documentation facilities to support all phases of your application development effort.

Introduction

Overall, documentation makes an application more effective. When used in combination, the documentation facilities let you fully document the application development effort either for your own benefit or for the benefit of other interested parties. This chapter describes the complete facilities in three sections. The first section addresses documentation of the knowledge base files for developers, the second section addresses session processing records for application testers, and the third section addresses session help for end users.

Documentation that you provide with your application is helpful in a number of ways. It lets you manage the application development effort and anticipate your application's needs. The Rules Element documentation facilities described in this chapter let you add the following benefits to your application.

- Developers can make more intelligent modifications to the application.
- Testers can understand inferencing behavior while testing the application.
- End users can request an explanation for actions the system requires them to perform.

The amount of documentation you provide is your choice. You may want to use only the built-in explanation facility or you may want to combine it with other documentation facilities for full support. You can keep hardcopy records of rule and object structures and inferencing sessions. Or you can record actual sessions for later replay. You can enhance the application with graphic images and text files. You can also customize the default explanation messages as desired.

Knowledge Base Files

Documenting knowledge base files is an important task of the application development effort. This type of documentation, which you create at the level of rule and object structures, lets you keep track of the growing application. Knowledge base documentation lets you communicate your progress to other interested parties and minimize undesired effects when modifying an application. You can display knowledge base documentation in two major ways:

■ From the knowledge base file itself
■ From printouts of the knowledge base.

The following sections explain the facilities available for documenting knowledge base structures and keeping a record of your progress.

Rule Names

The Rule field of the Rule editor lets you assign meaningful names to the rules you create. Unlike the hypothesis name, the rule name you enter is not involved in inferencing. The field merely holds the names you choose to associate with rules displayed in the Rule editor. Once you assign rule names, the Rule editor can be configured to sort by rule names. Figure 6-1 shows a sample rule name as it appears in the Rule editor.

Figure 6-1 Sample Rule Name

Rule names you assign may identify the rule's function and help developers to understand the rule's purpose in the knowledge base. The system requires rule names you assign to be unique across knowledge bases when loaded into memory. A warning message will appear if you try to create a rule with a name already defined by a currently loaded knowledge base. Or, if you load a knowledge base that duplicates a rule name in a currently loaded knowledge base, the system will rename the loading rule to prevent a conflict from occurring. If no name is supplied in the Rule field for a particular rule, the system assigns a unique name in the form of "R_hyponame."

Use the following procedure to assign rule names to a rule you create in the Rule editor.

To assign rule names:

1. Open the Rule editor and click on the desired mode button.

2. Complete the rule components if necessary and click on the Rule field.

3. Type the rule name in the edit line, using an underscore character (_) for the space between words. Press return when done.

    Rule names must be unique and cannot include space characters; use an underscore to form one string.

4. Click on the **OK** button to compile the rule.

    If the name is already used in any currently loaded knowledge base, the Rule editor will prevent the rule from being saved and requires that you assign a unique name before saving.

Use the following procedure to sort the Rule editor by rule names instead of rule hypotheses.

To sort by rule names:

1. Open the Rule editor and click on the desired mode button.

2. Select the Sort option from the window-specific Rule menu on the main menu bar. The options include Hypotheses and Rule Names; select the Rule Names option.

3. To sort by rule names select the desired alphabetic button from the Rule editor lateral index. The system displays the rule with a rule name corresponding to the letters of the box.

Comments

The Rules Element lets you comment your knowledge base and save the comments with the rule and object structures for future reference. This is similar to commenting code in a traditional application because comments let you compare different versions and keep track of modifications. Often comments you supply for individual rule and object structures can prevent another developer from introducing undesired side-effects.

You can add comments directly to the rule and object structures shown in the Rule editor, Method editor, and Meta-Slot editor windows. These particular structures receive your comments because they have the greatest effect on inference engine processing. For instance, a comment you add to a rule might explain that an Order of Sources for a particular left-hand side

condition was established in the Method editor. Figure 6-2 shows a sample comment as it appears in the Rule editor.

Comments	Order of Sources defined for total_load_value in Method editor	w-x
Why		y-z
Inf. Priority Num.	1	Inf. Priority Slot	
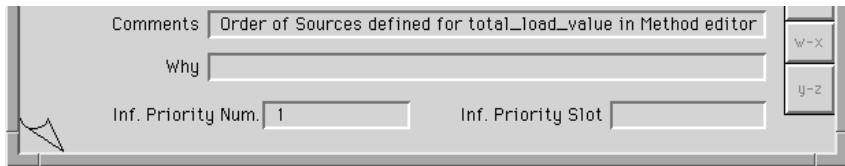
Figure 6-2   Sample Rule Editor Comments

Note: Comments you add to the Rule editor, Method editor, and Meta-Slot editor are also recorded in these rule and object structures' printouts.

Use the following procedure to comment rule and object structures of the Rule editor, Method editor, and Meta-Slot editor.

To comment rule and object structures:

1. Open the Rule editor, Method editor, or Meta-Slot editor and browse the editor to display the desired rule or object structure.

2. Click on the **Modify** button in the editor window.

3. Click on the Comment field and type the comments in the text edit line. Press return when done.

4. Click on the **OK** button in the editor window to exit the edit mode. Unlike the other fields in these windows, the system does not compile and save comments you add.

5. When finished with the current session, select the Save Knowledge Base option on the Expert menu. The system opens the save file dialog window.

6. Ensure the Save Comments and Whys checkbox has a selected status. The default saves comments with the knowledge base.

7. Rename the knowledge base if desired and click on the dialog window **OK** button to write the file with comments to disk.

Printouts

The Print command is a standard option for windows in the Rules Element. You can use it to print out the following items.

■   Structures displayed in editor windows and list windows

- Static or dynamic network diagrams from the network windows
- Text windows that log inferencing events.

It is a good idea to use this facility to keep a record of the growing application. For example, you may want to organize your printouts of the knowledge base into a binder that reflects the important project milestones.

The following paragraphs describe useful knowledge base printouts.

Rule and Object Structures

Window printouts duplicate the information that appears in the window. For example, the List of Rules window prints out the entire window with the text it contains. You can print this window to obtain a record of the knowledge base rules. Listing 6-1 shows the text that appears in the List of Rules window printout.

```
RULE :  Rule Get_Server (#6)
IF
        there is evidence of Get_Server_Type
        And average_server_os is a member of <|CUSTOMER_SYSTEM|>
        And total_load_value is less than or equal to 60
        And server_102.name is assigned to server_choice
THEN GET_SERVER
        is confirmed.
        And server_102.needed is set to TRUE
        And Create Object server_102 |CUSTOMER_SYSTEM|
        And Delete Object average_server_os |CUSTOMER_SYSTEM|
        And Create Object bundled_operating_system |CUSTOMER_SYSTEM|
```

Listing 6-1   Sample List of Rules Window Printout

Other windows, such as the Object editor and Class editor, can be printed to maintain a list of the knowledge base objects; they can also provide a useful index for locating objects used in rules. You can print these windows to obtain a record of the knowledge base objects. Listing 6-2 shows the text that appears in Object editor window printouts.

```
Object :      ai_engine
Classes :   Client_Server
```

Listing 6-2   Sample Object Editor Printout

Windows that you can browse in the windowing environment give you an option to print the current screen display or every rule or object structure the window can access. To facilitate printing large portions of the application at a time, each window's default print mode is set to print all. If you want to limit the printout to the current screen display, you can select the "Only Current Page" option in the print dialog window. Use the following procedure to obtain printouts of the currently loaded knowledge base files.

To print out rule and object structures and text windows:

1. Display the global popup menu for the active window and select the Print option from the list. The system opens the print dialog window.

2. Click on the Only Current Page checkbox if desired. The default prints out the entire contents of the window.

3. Click on the Used In Information checkbox to include cross-references in the printout. Click on any other checkboxes as desired.
   Only the Object editor and Class editor provide additional print options.

4. Click on the edit line and rename the printout header if desired.

5. Click on the **OK** button to print out the contents of the active window.

Network Diagrams

You can also obtain a printout of the network diagrams using the Print command on the desired network windows. A dialog window appears when you select the Print command. You select the page range and other standard print dialog options.

The printout you create duplicates the information that appears in the network window. Depending on the current usage, the network diagram may be a static representation of the network or a dynamic one that shows a particular state of the inferencing session. Use the following procedure to select options from the print dialog window and print out the network diagrams.

Note: Before printing out a network diagram, you can tailor its visual representation. Network diagram settings include fonts, styles, icons, colors, orientation, width, and length. For details refer to Appendix D, "Customizing the Environment."

To print out network diagrams:

1. Display the global popup menu for the network window and select the Print option from the list. The system opens the print dialog window.

2. Select the print dialog options and specify the page range as desired.

   If you want to increase or reduce the number of pages required to print the entire network diagram, see the following platform-dependent sections to scale the network image sent to the printer.

3. Click on the **OK** button to print out the selected portions of the network diagram.

Macintosh Scaling

The default print mode is set to the size of the network diagram on the screen. Figure 6-3 shows the standard Macintosh dialog window used to scale the network diagram printout. Macintosh users select the Page Setup option on the File menu. The scale is adjusted by entering a scaling factor in the Reduce or Enlarge field of the Page Setup dialog window..
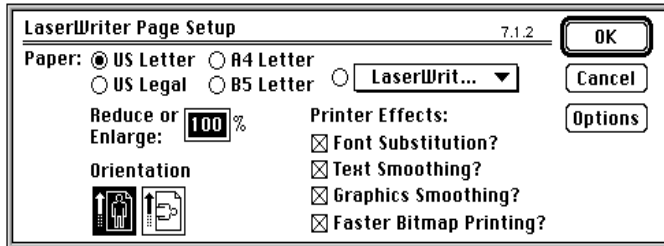
Figure 6-3 Macintosh Page Setup Dialog Window

PC, Unix, and VAX Network Window Scaling

Users of PC, Unix, and VAX platforms can combine scale options from the Print Setup dialog (see Appendix D, "Customizing the Environment") along with several other ways to obtain the fewest number of pages for the network diagram printout. For example, you may want to fit the entire network diagram on one or two printout pages. While the scale can be adjusted to accomplish this, it may result in text that is difficult or too small to read. Therefore the best approach is to experiment with several different reduction approaches. Your success will depend on the size and shape of the network diagram as it appears in the network window. The complete list of modifications you can make is as follows.

Note: Appendix D, "Customizing the Environment" describes the PC, Unix, and VAX workstation print setup dialog window.

- Adjust the scale percentage to approximately 50% (network diagram printout remains readable).
- Select the Landscape option to change the paper orientation (Portrait orientation is the default).
- Reduce the print border to less than one inch.

Session Processing

Records of session processing serve two purposes depending on the status of the application. If the application has reached an important milestone,

you can archive session processing records as documentation of the application development effort. If the application is undergoing testing, session processing records can aid the tester. To support these tasks you can obtain the following types of records during session processing.

■ Comprehensive inference engine events transcript

■ List of values entered during inferencing

■ Recording of the actual inferencing session.

You obtain these records through several different facilities in the Rules Element. The following sections describe the operation of the facilities for creating session processing records.

Transcripts

When you begin an inferencing session, you can enable the Transcript window to obtain a comprehensive log of inference engine events. The system dynamically updates the window and maintains it until the inferencing session ends. Among the many events it logs are: answers, modifications, initial hypotheses, volunteered data, triggered methods, suggested hypotheses, results of internal computations, execute calls, and so on. Listing 6-3 shows typical session information output to the Transcript window.

```
# Suggesting GET_BASICS_USER_INFO
# <|NODES|>=macintoshes,os_2_machines
# Retrieve: 2 records scanned, 2 records retrieved, 2 fields retrieved
# Condition Retrieve "computer.nxp"
   @TYPE=NXPDB;@FWRD=FALSE;@PROPS=number;@FIELDS="number";@ATOMS=<|NODES|>;
   in rule 3. (True).
# macintoshes.number is set to 15
# os_2_machines.number is set to 6
#
<|ACTIVITIES|>=printing,application_server,archiving,database_server,ai_engin
e,
   compiler_engine,database_engine,math_engine
# Retrieve: 9 records scanned, 8 records retrieved, 8 fields retrieved
# Condition Retrieve "activity.nxp"

@TYPE=NXPDB;@FWRD=FALSE;@PROPS=num_Macs,num_OS_2,num_computers;@FIELDS="num_m
acs","
   num_os_2","num_computers";@ATOMS=<|ACTIVITIES|>;
   in rule 3. (True).
# printing.num_computers is set to 5
# application_server.num_computers is set to 2
# archiving.num_computers is set to 4
# database_server.num_computers is set to 7
# ai_engine.num_computers is set to 5
```

Listing 6-3   Sample Transcript Window Log

Note: The Transcript window is also a useful debugging tool. Refer to Chapter Five, "Application Testing" for complete information about using the Transcript window for debugging.

The Transcript window is the same window that the system displays after you launch the Rules Element. Initially, it gives copyright information; you can close the window and enable it later for an inferencing session. Use the following procedure to enable the Transcript window for an inferencing session and obtain a permanent record of the log.

To enable the Transcript window:

1. Select the Transcript option on the Reports menu. The system opens the Transcript window.
   If you want to log the entire inferencing session, enable the Transcript window before initiating processing.

2. Display the Transcript window and click on the write enable icon. The system enables the window.

3. Begin the inferencing session. The system automatically logs the inferencing events in the Transcript window.

4. To discontinue the log, click on the write disable icon.

5. Print out the log or write it to a file to obtain a permanent record of the inferencing session events.

Journal Files

Before starting an inferencing session, you can enable the Journal facility to obtain an interactive record of the inference engine events. A window (see Figure 6-4) appears when you select the Journal command that has the following functions.

■ Record and replay the session (or portion of one)
■ Save slot values in a database file
■ Save and restore complete session state.

These functions create the Rules Element files that let you later recreate the session under the Rules Element. The Rules Element files that you back-up can serve as session case studies of the application. If you plan to archive

these files, you must also make an archive copy of the current knowledge base to reuse with the Rules Element file.
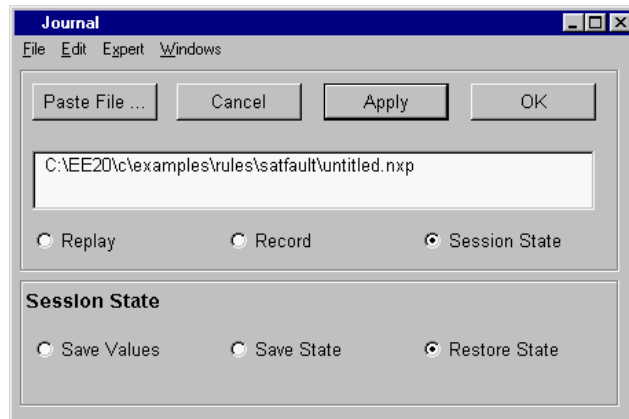
Figure 6-4   Journal Window

The following sections describe the operation of the Journal facility to obtain session processing records.

Record Start and Stop

The Record Start and Stop buttons of the Journal facility let you "tape" the events of an inferencing session that control processing. The specific events the system records include volunteered data, suggested hypotheses, modified data, and end user inputs. The system stores these events in an extended NXP file format that you can view and modify with any text editor. Each line in the journal file has the following format.

```
\slotname\="value"@X
```

where X is: fS for a suggested event
             V for a volunteered event
             Q for a question event

Listing 6-4 shows the contents of a typical journal file you create with the Recording Start and Stop buttons.

```
\GET_BASICS_USER_INFO\=""@S
\server_202.name\="22"@Q
***********
```

Listing 6-4   Sample Journal File

Note: The Journal facility does not record interactions with external functions, processes, or files. This includes actions initiated by the Execute, Write, and Retrieve operators.

You can invoke the Journal facility either before or during an inferencing session. The record function continues even after the end of the session so you can continue inferencing using the modified data or different hypotheses. Use the following procedure to record an inferencing session in a journal file.

To record a journal file:

1. Select the Journal option on the Reports menu. The system opens the Journal window.
    Select the Journal option on the session control panel popup menu once the session begins.

2. Select the **Record** button. The system displays the Record panel options.

3. Select the **Start** button from the Record panel. The system positions the cursor in the text edit line.

4. Edit the file name as necessary and change the search path if desired.

5. Click on the **OK** button to enable the file currently shown in the text edit line for recording.
    Select the Apply button to keep the Journal window displayed.

6. Begin the inferencing session, the system automatically records the inferencing events in the journal file.

7. To discontinue the journal, redisplay the Journal window and select the "Record" **Stop** button. Click on the **OK** button to confirm.
    If you do not manually discontinue the journal, the system continues journalling until you select the Restart Session option (on the Expert menu, for example). This has the same effect as selecting the Stop button.

8. Archive the journal file and make a copy of the text format knowledge base file(s) to archive with the journal file.
    Changes to the original knowledge base file(s) may interfere with the replay fidelity of the archived journal file. To ensure compatibility with later versions of the software, archive knowledge base files in the .tkb format.

Replay Start and Stop

The Replay Start and Stop buttons work with the replay options settings to determine the way to replay previously recorded journal files. The default settings of the replay options let you replay a journal file to view an inferencing session conducted with an archived knowledge base. Only one

of the three replay options shown in Table 6-1 is selected in the default mode.

Option	Default Status	Default Mode Replay Description
Step by Step	Selected	Replays the question/answer dialog in the same order as the original session.
Don't Scan the File for Values	Unselected	Uses the data available to continue inferencing from the journal file.
Skip Show Statements	Unselected	Allows files originally invoked by the Show operator to appear.

Table 6-1   Journal Default Replay Options

Note: You can change the Replay options settings to conduct an interactive debugging session. Refer to Chapter Five, "Application Testing" for complete information about using the Journal facility for debugging.

To ensure the journal file gives an exact duplicate of the recorded session, you must use the original knowledge base. Modifications you make to the application knowledge base files after recording may prevent the journal file from working properly. Use the following procedure to replay a session using a previously recorded journal file, an unmodified knowledge base, and the default replay options.

To replay a journal file:

1. Load the original knowledge base file(s) if necessary.

2. Select the Journal option on the Reports menu. The system opens the Journal window.
   Select the Journal option on the session control panel popup menu once the session begins.

3. Select the **Replay** button. The system displays the Replay panel options.

4. Select the **Start** button from the Replay panel. The system positions the cursor in the text edit line.

5. Enter the file name of the desired journal file and change the search path if necessary.

6. Click on the **OK** button to enable the file currently shown in the text edit line for replay. The system displays the session control panel with the recorded value displayed in the input field.
   Select the Apply button to keep the Journal window displayed.

7. Click on the session control panel **OK** button to continue replaying the journal file with the original value.

Editing the session control panel values is described in Chapter Five, "Application Testing."

8. Continue processing values until the journal file finishes replaying. To manually discontinue the journal, redisplay the Journal window and select the "Replay" Stop button. Click on the OK button to confirm.

Save Values

The Save Values radio button (see Figure 6-4) lets you document the state of data any time during an inferencing session. You can write your inferencing session's current slot values to a database file and later initiate a session using the stored values. Unlike a journal file, a saved values file does not permit replaying the session, but you can independently suggest hypotheses to begin a session with the known data.

The system stores the current slot values in an NXP file format that you can view and modify with any text editor. Each line in the values database file shows a slot value as represented in Listing 6-5.

```
\compiler_engine.load_contribution\="40.0"
\database_engine.load_contribution\="120.0"
\math_engine.load_contribution\="80.0"
\total_load_value\="439.0"
\Get_Server_Type\="True"
\database_engine.storage_needs\="60.0"
\database_server.storage_needs\="70.0"
\archiving.storage_needs\="24.0"
\storage_needed\="249.0"
***********
```

Listing 6-5   Sample Values Database File

Use the following procedure to write the current slot values of the inferencing session to a values database file.

To create a values database file:

1. During the inferencing session, select the Journal option on the session control panel popup menu. The system opens the Journal window with the Session State panel displayed.

2. Select the **Save Values** button from the Session State panel. The system positions the cursor in the text edit line.

3. Edit the file name as necessary and change the search path if desired.

4. Click on the **OK** button to enable the file currently shown in the text edit line to store the current values of the session.
   Select the Apply button to keep the Journal window displayed.

5. Archive the database file and make a copy of the text format knowledge base file(s) to archive with the database file.
   Changes to the original knowledge base file(s) may prevent the system from restoring the state exactly. To ensure compatibility with later versions of the software, archive knowledge base files in the `.tkb` format.

Initiating the inferencing session using only saved slot values, means you must manually specify a starting point for the inferencing session. In particular you can suggest hypotheses to determine how the saved data affects the outcome.

Save State

The Save State radio button (see Figure 6-4) lets you capture the current logical and physical state of the inferencing session. This particular function, used in combination with the Restore State button, is the fastest way to save a session. However, unlike the other Journal facility functions (record session and save values), the save state function does not let you replay or modify inferencing events. The complete session state the system saves includes:

- The current values of all the rule and object structures
- The values and links of existing dynamic objects
- The entire agenda mechanism status.

Note: Unlike other NXP files, the file the system creates to store these items cannot be accessed from a text editor.

Use the following procedure to save the current inferencing session's logical and physical state.

To save the current session state:

1. During the inferencing session, select the Journal option on the session control panel popup menu. The system opens the Journal window with the Session State panel displayed.

2. Select the **Save State** button from the Session State panel. The system positions the cursor in the text edit line.

3. Edit the file name as necessary and change the search path if desired.

4. Click on the **OK** button to enable the file currently shown in the text edit line to store the current session state.
   Select the Apply button to keep the Journal window displayed.

5. Archive the saved state file and make a copy of the text format knowledge base file(s) to archive with the state file.

Changes to the original knowledge base file(s) may prevent the system from restoring the state exactly. To ensure compatibility with later versions of the software, archive knowledge base files in the `.tkb` format.

Restore State

The Restore State radio button (see Figure 6-4) lets you resume an inferencing session at the exact point where you previously selected the Save State button. When you select the Restore State button, the system uses the specially optimized NXP file to restore the session very quickly.

To ensure the restore function works properly, use only the original knowledge base file(s). Session state files created with previous versions of the Rules Element are compatible with more recent software releases. Use the following procedure to restore a session state from its previously saved state.

To restore a session state:

1. Load the original knowledge base file(s) in the saved state sequence. The system displays an error message and aborts the restore operation if files have been loaded out of sequence.

2. Select the Journal option on the Reports menu. The system opens the Journal window with the Session State panel displayed.

3. Select the **Restore State** button from the Session State panel. The system positions the cursor in the text edit line.

4. Enter the file name of the desired saved state file and change the search path if necessary.

5. Click on the **OK** button to restore the session state from the file. The system displays the session control panel to continue processing. Select the Apply button to keep the Journal window displayed.

Session Help

If your application involves end user interaction, the availability of on-line help becomes an important consideration in the application development effort. This kind of help usually gives end users decision-making support to ensure accurate processing input. The Rules Element provides the following facilities to address this need.

- Prompt line (default or custom) for requesting user input
- Explanation facility (default or custom) for the current rule

■ Display files (custom only) that enhance interactions.

The end user accesses these facilities through the session control panel on the main window during an inferencing session. The following sections describe how to customize the facilities for the end users of your application.

Prompt Line

During an interactive inferencing session, the system displays the question in the main window to solicit end user input. In the session control panel appears a single question generated by the system when it cannot determine the value of the slot under evaluation. Figure 6-5 shows the form of a typical default question.
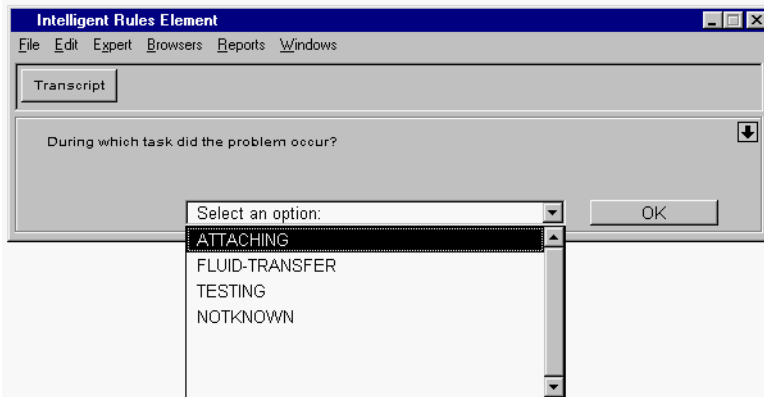
Figure 6-5   Sample Main Window Question

Note: It is also possible to substitute a window of your own design for the question/answer dialog of the main window. The prompt line, if available, will automatically appear in the custom window. For complete instructions about creating question windows, refer to the Open Interface User's Guide.

The system derives the default question for each question/answer dialog from a set of syntactic rules. It automatically generates the question to determine the value of the slot encountered in 1) the LHS condition of a rule or method or 2) the Order of Sources' AskQuestion operator. But because the question follows a standard form, it may not be well suited in every case.

You can customize the default question or "prompt line" through the Prompt Line field of the Meta-Slot editor. The text you enter into this field changes the way the system prompts the end user for the slot named in the Meta-Slot editor. If the system finds a customized prompt line in the

Meta-Slot editor for the encountered slot, it automatically substitutes it for the default question. The system will also try to inherit the prompt line from a class slot by default, if the children of that class do not already have a prompt line defined.

The prompt line you enter for the slot named in the Meta-Slot editor can include variables that the system interprets at runtime. Table 6-2 shows the syntax of prompt line variables.

Variable Format	Substitution	Function
`@V(ObjName.Prop)`	Any slot value.	Lets you name the value of any evaluated slot in the prompt line.
`@V(@SELF.Prop)`	The current slot value.	Lets you create generic prompts - suitable for inheritance by another slot at runtime.
`@SELF.Prop`	The current slot name.	Similar to the default question, but makes prompt generic - suitable for inheritance.
`@PROP`	The current property name.	Lets you name the property of the currently evaluated slot in the prompt line.

Table 6-2   Meta-Slot Prompt Line Variables

Use the following procedure to substitute your own prompt line for the default question associated with slots named in rule conditions and the Order of Sources AskQuestion operator.

To customize the default question:

1. Open the Meta-Slot editor and browse the editor to display the desired slot in the Slot field.
    The slot named must be one used in a rule condition, method condition, or AskQuestion operator of the Order of Sources.

2. Click on the **Modify** button.

3. Click on the Prompt Line field and type the custom prompt in the text edit line. Press return when done.

4. Click on the **OK** button to verify and compile the new system attribute.

An example of prompt line customization follows.

Example

Assume a datum in the application is called `CRT_and_KDU` and has two values, agree and disagree.

The default question reads as:

```
What is the value of CRT_and_KDU?
```

The customized version, that you enter in the Prompt Line field of the Meta-Slot editor, could read:

```
Do the two screens CRT and KDU agree?
```

Continuing the example, assume the end user selects AGREE and the next question in the question/answer dialog concerns the current task of the operator using the application, where the variable is Current_Task.

Once again, you can type the following prompt line in the Meta-Slot editor to substitute for the default question.

```
Since the CRT and KDU @V(CRT_and_KDU), the problem is confirmed and
we need to know the task you are currently undertaking.
```

The answer/dialog lists all the different options for the Current_Task and since the response to the first question was AGREE, the actual prompt line displayed in the session control panel becomes:

```
Since the CRT and KDU AGREE, the problem is confirmed and we need to
know the task you are currently undertaking.
```

Why and How

During an interactive inferencing session, the system displays the session control panel in the main window to solicit end user input. If desired, the end user can obtain a rationale for the requested input from the on-line explanation facility. The facility lets the end user browse the backward chaining links that lead to the slot currently under evaluation.

The end user accesses this facility by selecting the Why option on the session control panel popup menu. The system responds by displaying a dialog

window that gives the browsing functions and explanation text. Figure 6-6 shows a typical explanation dialog window.
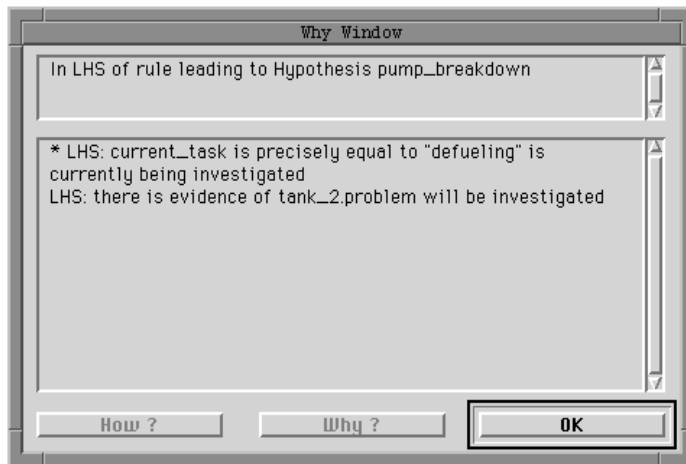
Figure 6-6   Explanation Facility Dialog Window

The explanation shown in the dialog window includes two text boxes. The top box gives information about the hypothesis and the bottom box gives information about the LHS conditions. When the end user selects the Why button, the text changes to reflect the hypothesis and conditions of the next rule in the backward chaining links. When the end user selects the How button the text changes to reflect the hypothesis and conditions of the previous rule in the backward chaining links.

Similar to the default question, the system follows syntactic rules to derive the text that accompanies each explanation dialog window. But because the text follows a standard form, it may not give the most appropriate information in every case. You can use the Why fields of the Rule editor and Meta-Slot editor to customize the way the system explains the current rule. The following sections show how to customize the explanation facility.

Meta-Slot Editor Why

You can use the Why field of the Meta-Slot editor to customize the way the system explains the slot encountered in the hypothesis or Order of Sources AskQuestion operator. If the system finds Why text in the Meta-Slot editor for the encountered slot, it automatically substitutes it for the default explanation shown in the TOP half of the explanation dialog window (see Figure 6-6).

The why text you enter for slots named in the Meta-Slot editor can include variables that the system interprets at runtime. Table 6-3 shows the syntax of why text variables.

Variable Format	Substitution	Function
@V(ObjName.Prop)	Any slot value.	Lets you specify the value of any evaluated slot in the explanation.
@F(FileName)	Any text file.	Lets you give more information in a text file. File can use "@V" variables that the system interprets at runtime.

Table 6-3   Meta-Slot Why Variables

Use the following procedure to substitute your own why text for the default explanation associated with slots named in a hypothesis or Order of Sources AskQuestion operator.

To customize the slot explanation:

1. Open the Meta-Slot editor and browse the editor to display the desired slot in the Slot field.
    The slot named must be one used in a hypothesis or AskQuestion operator of the Order of Sources.

2. Click on the **Modify** button.

3. Click on the Why field and type the explanation in the text edit line. Press return when done.

4. Click on the **OK** button to verify and compile the new system attribute.

5. When finished with the current session, select the Save Knowledge Base option on the Expert menu. The system opens the save file dialog window.

6. Ensure the Save Comments and Whys checkbox has a selected status. The default saves comments with the knowledge base.

7. Rename the knowledge base if desired and click on the dialog window **OK** button to write the file with explanations to disk.

Rule Editor Why

You can use the Why field of the Rule editor to customize the way the system explains the slot encountered in the LHS conditions of rules. If the system finds why text in the Rule editor for the encountered slot, it automatically substitutes it for the default explanation shown in the BOTTOM half of the explanation dialog window (see Figure 6-6).

The why text you enter for rules named in the Rule editor can also include variables that the system interprets at runtime. Table 6-4 shows the syntax of why text variables.

Variable Format	Substitution	Function
`@V(ObjName.Prop)`	Any slot value.	Lets you specify the value of any evaluated slot in the explanation.
`@F(FileName)`	Any text file.	Lets you give more detailed information in a text file. File can use variables (@V) the system interprets at runtime.

Table 6-4   Rule Why Variables

Use the following procedure to substitute your own why text for the default explanation associated with rules named in the Rule editor.

To customize the rule explanation:

1. Open the Rule editor and browse the editor to display the desired rule.

2. Click on the **Modify** button.

3. Click on the Why field and type the rule explanation in the text edit line. Press return when done.

4. Click on the **OK** button to verify and compile the rule.

5. When finished with the current session, select the Save Knowledge Base option on the Expert menu. The system opens the save file dialog window.

6. Ensure the Save Comments and Whys checkbox has a selected status. The default saves explanations with the knowledge base.

7. Rename the knowledge base if desired and click on the dialog window **OK** button to write the file with explanations to disk.

Apropos Files

You can create display files in a compatible text editor or paint program that the end user optionally displays during an inferencing session. These files that you create provide the end user with supplemental information to help them answer the prompt for the current inferencing session activity. For example, a graphics file might show a circuit diagram to help the end user identify a circuit fault.

The end user requests the file before answering the prompt line by selecting the Apropos command on the session control panel's popup menu.

Alternately, you can view apropos files that you define for object structures displayed in the network diagrams by using the Apropos command from local popup menus. The system searches for a file to display that matches the same name of the class, object, property, or slot currently under evaluation. Thus these files are known as "Apropos" files because the system displays only the file appropriate for the object structure involved.

> **Note:** Files that you create for display in the development system may have to be converted to a format that is compatible with the final delivery environment. For example, utilities exist to convert MacPaint files on the Macintosh to bit-mapped graphics for Unix and DEC Windows.

Use the following procedure to create Apropos files.

To associate an Apropos file with a specific object structure:

1. Identify the name of the object structure to which you want to assign an Apropos file.
   All class, object, properties, and slots are valid structures to assign an Apropos file. However, the more specific the structure, the better targeted the information can become. The slot is the most specific item in any application.

2. Create the file and assign the actual file a name that matches the associated structure's name.
   The two names must match exactly, including upper and lower case. For example, to assign an Apropos file to the slot `gyro.z`, the file must be given the name "gyro.z".

3. Place the Apropos file into a file directory that permits the Rules Element to locate it during a session.
   Information about defining the Rules Element search paths appears in Chapter Four, "Application Processing."

Another way of using the Apropos file involves the use of the Show operator in rule left-hand side conditions and right-hand side actions. By including the Show operator in a rule or method of the knowledge base, you can specify precisely when the end user will view information about the inferencing session. Refer to the Intelligent Rules Element Language Reference manual for details about using the Show operator to automatically display Apropos files during an inferencing session.

7 *Application Data*

This chapter introduces the Intelligent Rules Element database interface and its capabilities. It includes information on using the database interface in Rules Element applications, and introduces the Retrieve and Write operators.

About the Database Interface

The Rules Element database interface is a link between your database and the Rules Element. The database interface supports a wide variety of file and database formats, including flat-files, spreadsheets, and relational databases. Logically, all of these are organized in one of two ways:

■ Spreadsheets, which are composed of independent data values, or cells.
■ Databases, which are composed of records which are divided up into fields.

Through this link, you can do two things: retrieve and write. You can retrieve data from your database and create objects in the Rules Element, or you can write to your database using Rules Element objects.

Invoking the Database Interface

The Rules Element database interface is invoked when the Rules Element processes a Retrieve or Write statement. The Retrieve gets data from a database into the Rules Element's working memory, the Write moves data from the Rules Element's working memory to a database. These statements can occur in the following places in the Rules Element:

Operator	Where you can use the operator
Retrieve	Left-hand side or right-hand side of rules and methods.
Write	Left-hand side or right-hand side of rules and methods.

Like most Rules Element statements, Retrieve and Write consists of three parts:

■ The Retrieve or Write operator
■ A filename or database access string. Some external data sources, like flat files, require that you name the target file name here. Others, like

relational databases, may require you to enter a identifier, such as userid and password.

■ One or more arguments, which provides specific information about the type of access to be done by the Rules Element database interface.

The last part of the Retrieve or Write statement, the arguments, passes specific information about the database access to the Rules Element database interface. These arguments are built by filling in the fields in the database retrieve or write windows shown in Figures 7-4 and 7-5.

Possible Operations

Using the database interface, you can do these tasks with the Retrieve or Write operator:

Operator	Tasks you can do
Retrieve	Modify existing, compiled objects. Create dynamic objects.
Write	Select records. Modify records. Create records. Create new files.

When using relational databases, you can also do tasks such as delete record, create tables, and delete tables, but you can only do these tasks implicitly. To do these tasks, specify the appropriate query(s) that perform the tasks in the BEGIN field of a Retrieve or Write and then specify a Retrieve or Write that doesn't have any effect. See the Begin field in Chapter Three, "Database Integration Topics" of the Language Reference for details.

Rules Element Representation Review

The Rules Element represents data in an object-oriented fashion which optimizes its ability to separate knowledge (rules, methods) from facts or data. In review, data is structured as follows:

■ Objects represent things. Objects represent a specific person, place, or thing which is reasoned over by the rules and methods in the knowledge base. As an example, each car in an automobile inventory could be represented by an object. Every object in the Rules Element's working memory has a name which uniquely identifies it. As an example, you could represent a car with an object whose name was `MyCar`.

■ Properties represent the characteristics of object. Each object has one or more properties which represent its attributes. For example, if the

objects in our knowledge base represent cars, then each object could have properties like `Model`, `Price`, `Sportive`, etc. Each property also has a type, which describes the kind of data stored in it. Valid types include `integer`, `boolean`, `time`, etc.

■ An individual object's properties are called slots. For example, the property `Price` of the object `MyCar` is referred to as the slot `MyCar.Price`, and may contain a value like `12,232`. Although the terms property and slot are sometime used interchangeably, it's important to remember that a slot is where the actual data value is stored.

■ Classes represent groups of objects which share the same properties. For example, you might create a class `cars_class` because all cars have attributes like `Model`, `Price`, and `Sportive`. Like objects, classes also have their attributes represented by properties. However, the purpose of properties at the class level is to provide a template of properties for objects created in that class. When an object is created in a class, it inherits the properties of that class. For example, if you create a class called `cars_class`, and associate the properties `Model`, `Price`, and `Sportive` with it, all objects created in the class `cars_class` will have those properties. So, if `MyCar` is created or attached to the class `cars_class`, it will automatically inherit the properties `Model`, `Price`, etc.

There are other, more complex constructs in the Rules Element's object representation such as subclasses and subobjects which have not been discussed here. For information on these topics, and more complete information on the Rules Element's object representation, refer to the Intelligent Rules Element Language Programmer's Guide.

Database Nomenclature

The Rules Element database bridge supports many different types of files and databases, including spreadsheets, flat files, and relational databases. In general discussions, when we refer to a database, it includes ALL these file types unless otherwise noted. Here are a few commonly used terms.

Records, Rows	Each file or database type has its own representation of data - flat files use records, relational databases use rows. In general, we will refer to records.
Fields, Columns	Each file or database type also has its own nomenclature for the subdivisions of records - flat files use fields, spreadsheets use cells, relational databases use columns. In general, we will refer to fields.

Spreadsheet Files

Spreadsheet files are those produced by spreadsheet programs such as Excel or Lotus 1-2-3 when a spreadsheet is saved to disk. Like the original spreadsheet, spreadsheet files are organized into cells. There are several standard formats for spreadsheet files in the industry, the Rules Element database interface can process files in the SYLK format (for Excel), WKS format (for Lotus 123), or the Rules Element's own NXP format. Data stored in these formats is accessed using the Rules Element Query Language (similar to SQL).
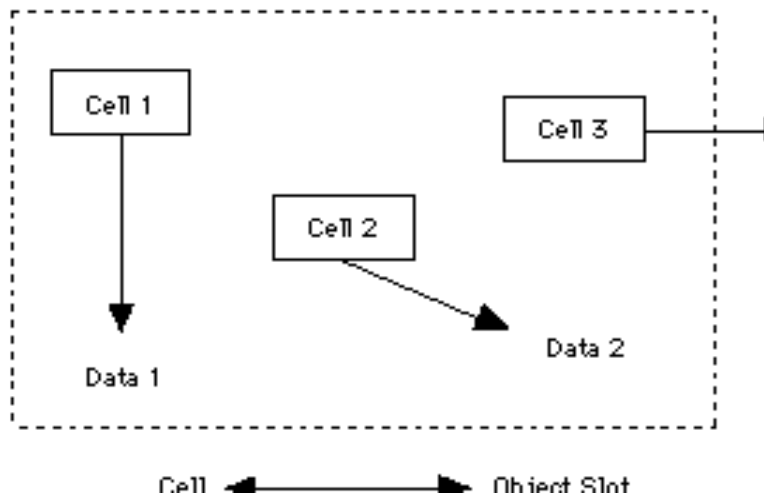
Figure 7-1   Spreadsheet Model

Flat-File Databases

Flat-file databases use a more general model than spreadsheets. In a flat-file database the information is stored in a table. The table consists of a set of records with each record having several fields. A record represents a logical unit of information (an employee, a travel expense, a part in inventory). A field represents an attribute of the records (name, age for employee records; amount or location for expense records). The database table is stored in a file which can be accessed and modified directly by the Rules Element, such as the DBF3 format (for dBaseIII) or the Rules Element's own NXPDB format.

Like spreadsheet files, data stored in these flat-file formats is accessed using the Rules Element Query Language (similar to SQL).
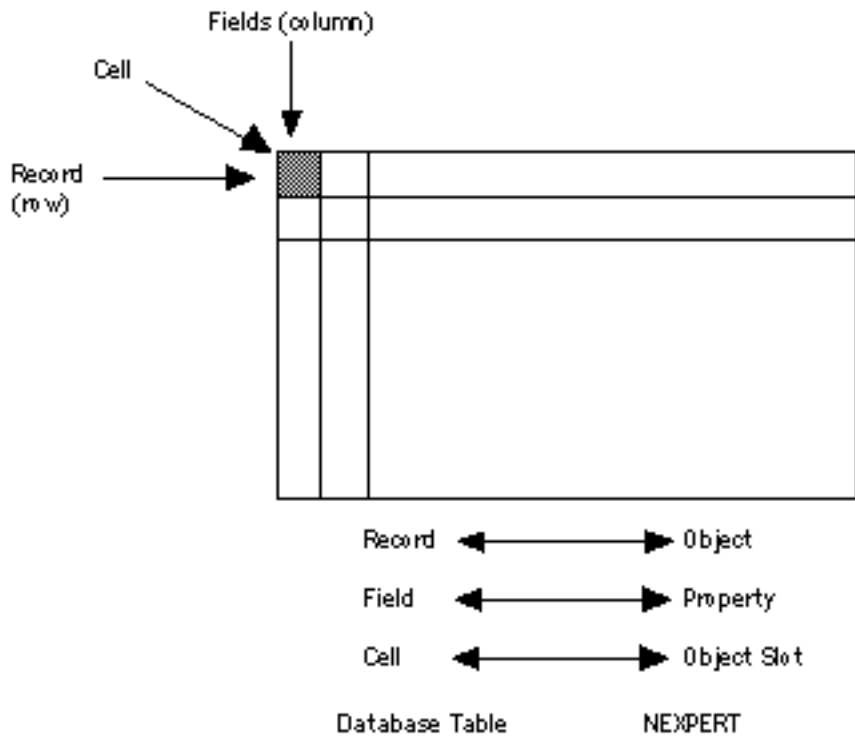
Figure 7-2   Flat-File Database Model

Relational Databases

Relational databases are products like Oracle, Sybase, Ingres, Informix, and SQL/DS which organize data using the relational model. Data stored in relational databases is accessed using Structured Query Language (SQL). Typically, data in a relational database is accessed through a database

manager, which processes the SQL statements, along with other services such as data security, multi-user access, and recovery facilities.
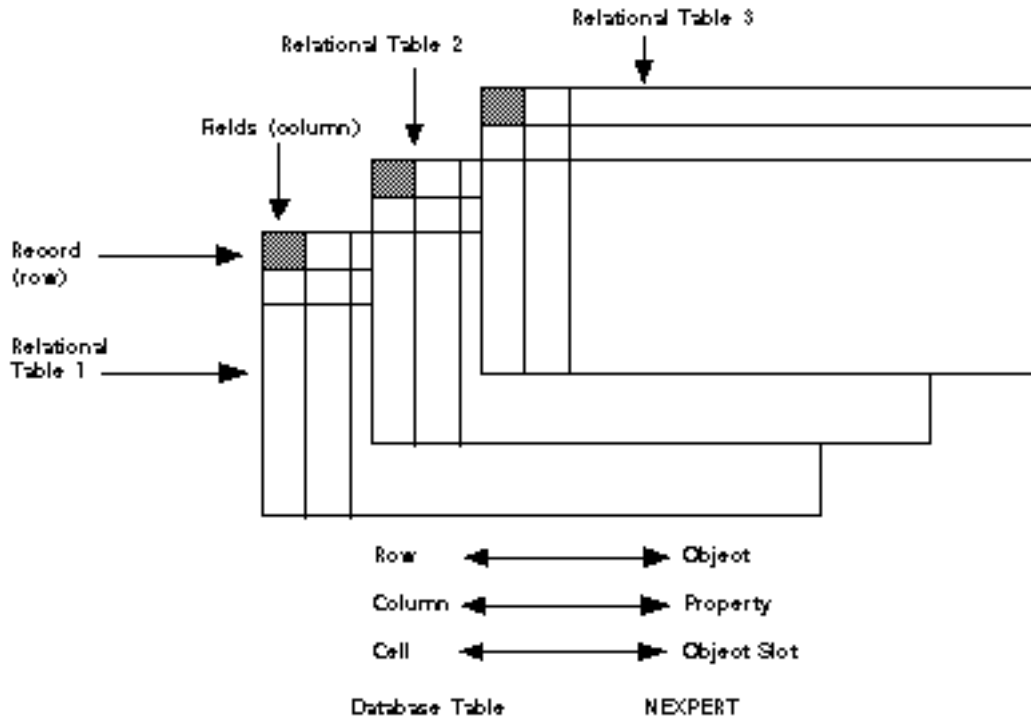
Figure 7-3 Relational Database Model

Three Ways to Use the Rules Element Database Interface

As illustrated in the previous section, the database interface provides three types of operations allowed with relational and flat-file databases:

■ Retrieving or writing one record. This is also referred to as an Atomic operation .

■ Retrieving or writing more than one record, one at a time. This is also referred to as a Sequential operation.

■ Retrieving or writing more than one record in a single operation. This is known as a Grouped operation.

Atomic Operations

Atomic operations transfer exactly one record between a database and the Rules Element's working memory, or vice versa.

Typically, an atomic operation is used to retrieve or write a single item of data during the inferencing process. For a write operation, this item of data consists of a fixed set of slots (object.property combinations), for a retrieve it consists of a single record's fields.

The exact data written during an atomic write is explicitly controlled by the list of slots passed to the database interface. Although there are a number of ways to specify this list, the end result is still that a fixed set of slots are transferred from the Rules Element's working memory to the database as a set of fields in a single record.

For an atomic retrieve, criteria in the Query field controls which record is transferred from the database into the Rules Element's working memory. Even if the query isn't strict enough to retrieve only one record from the database, only the first record will be retrieved.

Unlike sequential or grouped operations, atomic operations are completely self contained - there's no need to build any additional rules or object representations to support an atomic operation.

Sequential Operations

Sequential operations transfer many records between an external file or database and the Rules Element's working memory (or vice versa), one at a time.

Typically, sequential retrieval is used to process a number of records, one at a time. Each record is retrieved into a set of slots, processed, and the next records retrieved into the same slots for processing.

Sequential writes are typically associated with sequential reads since the slots will be written to the last record retrieved by the sequential retrieve. In this case, a record would be retrieved into a set of slots, processed, the slots written back out (to the same record), and the next record retrieved for processing.

Sequential operations are actually a special case of atomic operations. The difference is that sequential operations remember their position in the external file or database, and operate on the next record in the file. Using sequential operations is a bit more complicated than atomic operations because you must provide a data structure to remember the position in the

external file or database (a cursor), and set-up the knowledge base to re-execute the operation until all of the records have been processed.

Grouped Operations

Grouped operations transfer many records between an external file or database and the Rules Element's working memory (or vice versa), all in one operation.

Grouped retrieval is typically used to populate a class of objects with data from an external source. Once the group of records is retrieved and transformed into objects, Rules Element rules can process the objects without further concern for the source of the data.

Grouped write takes a group of objects (typically all the objects in a class) and write them out to an external file or database in one operation. No special processing of the objects is necessary before the write operation to prepare them for writing to the database.

Grouped operations usually don't require additional logic be built into the knowledge base to account for their operation, but the representation for the data - classes, objects, properties - must be provided to read the records into or write them out of.

Starting with Retrieve / Write

For all three methods of using the database interface, you need to start with the Retrieve or Write operator. This is how to use Retrieve or Write:

1. Display the Rule Editor if you want to put the Retrieve or Write operator in a rule. Go to the Method Editor if you want to put the Retrieve or Write operator in a method, such as Order of Sources or If Change of a particular slot.

2. Click on the **New** button in the editor window's menu bar. The system automatically highlights the first field for editing.

3. If you are using the Rule Editor, display the popup menu for the first column of either the Left-hand side or Right-hand side. Select the Retrieve or Write options as desired.

4. If you are using the Method Editor, display the popup menu for the first column of either the Left-hand side or Right-hand side. Select the Retrieve or Write options as desired.

5. Depending on which database you're using, specify the information your particular database requires for access. Here are some examples.

Oracle   "scott tiger t:hyperion:HYPERIONSID"

Sybase   "scott tiger hyperion servername appname
            database"

Lotus   c:\dir1\dir2\filename.wk1   (PC example)

A more complete description can be found under the database topics in the Intelligent Rules Element Language Reference. After you specify the connecton parameters, the system displays the Retrieve window or the Write window. The following figures show these windows.

Figure 7-4   Database Retrieve Window

The rest of the information in this chapter gives general procedures for using the database interface. For more detailed information about Retrieve and

Write operations refer to Chapter Three, "Database Integration Topics" of the Language Reference.
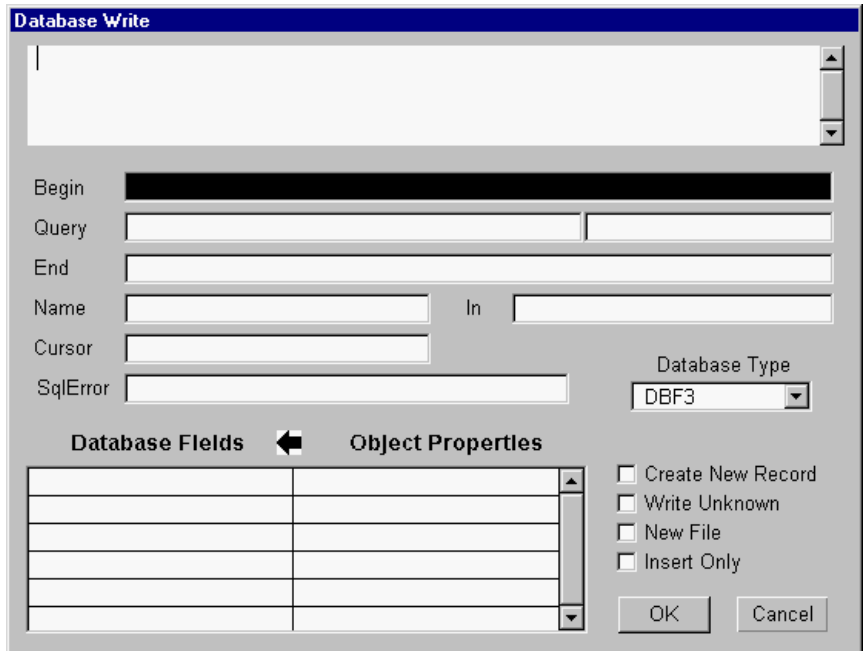
Figure 7-5   Database Write Window

Retrieving One Record

Atomic retrieves are used when the knowledge base needs to retrieve information from a single record into a single object. For example, if you want to select the record for car_1 from the CARS table, you can use a single-record query to get that information.

To indicate the operation is an atomic retrieve, the rule or method that invokes the retrieve must assign the value UNKNOWN to a cursor slot. The value of the slot determines whether the database interface will continue processing or terminate the operation. To begin the operation the cursor slot must have the value UNKNOWN. If it is not UNKNOWN, the database operation terminates. Use the Reset operator in a rule or method to initialize the cursor slot as shown below.

```
reset dummy_object.dummy_cursor     retrieve "db_access_string"
```

Figure 7-6 shows an example of a Retrieve window set-up for atomic processing. In this example the property slots of a single object are passed values from a single database record. Although this example is oriented towards relational databases, it is also applicable to flat-file databases. Refer to Chapter Three, "Database Integration Examples" for additional atomic processing examples.

Figure 7-6   Atomic Retrieve Operation

This is a description of each field in a single-record retrieve operation:

1. For most databases this field should be left blank for atomic retrieve operations. Some databases require a statement here.

If you are using a relational database, you can specify a statement in the Begin field. The statement is sent to the database server when the Retrieve initiates.
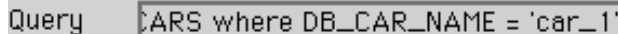
See the Begin topic or your database type in Chapter Three, "Database Integration Topics" of the Language Reference for details about the Begin field.

2. Select the type of database you're using in the Database Type window.

You can scroll the Database Type window to see all the types of databases supported by the database interface. Select the type that represents your database.

3. Specify the record that you want to retrieve using a query.

Use the query to specify which record you want to retrieve from. If more than one record meets the specifications of the query, only the first record is processed.

If you are using a flat-file type database, Neuron Data provides a SQL-like query language which is described in Chapter Three, "Database Integration Topics" of the Language Reference. See the Query Language topic for details.

4. For most databases, this field should be left blank. Some databases may require a statement here.

See your database type in Chapter Three, "Database Integration Topics" of the Language Reference for details about the End field.

5. Indicate that this is a single-record transaction by specifying an object slot with a value of UNKNOWN in the Cursor field.

Cursor | dummy_object.dummy_cursor

The cursor will indicate whether the transaction was successful. The cursor slot must have a value of UNKNOWN for this transaction to be treated as a single-record transaction. Make sure the cursor slot's value is set to UNKNOWN as explained in the introduction to this section.

6. Specify under Database Fields the fields in your database whose values you want to send across the database interface, and specify the corresponding Rules Element slots to be updated in the second column.

Database Fields ➡	Object Properties	
DB_MODEL	MyCar.Model	▲
DB_MODEL_DATE	MyCar.Model_date	
DB_PRICE	MyCar.Price	

See the Slot Specification for Retrieves and Writes in Chapter Three, "Database Integration Topics" of the Language Reference for details about specifying these fields.

7. If you want objects with a value of UNKNOWN to be processed, select the Retrieve Unknown option. Otherwise, objects with a value of UNKNOWN are not processed.

☐ **Retrieve Unknown**

8. Specify the forwarding strategy by selecting one of the following options: Always Forward, Current Forward, or Do Not Forward.

◇ **Always Forward**
◇ **Current Forward**
◇ **Do Not Forward**

Always Forward means that inference engine forwards the values in the rules network; Current Forward means that the inference engine uses the current forwarding strategy; and Do Not Forward means the inference engine does not use any forwarding strategy.

Writing One Record

Atomic writes take the slots from one or more objects and write them out to fields in a database record. In the vast majority of the cases, the slots are written to a single record, but its also possible to write multiple records with an atomic write operation.

To indicate the operation is an atomic write, the rule or method that invokes the retrieve must assign the value UNKNOWN to a cursor slot. The value of the slot determines whether the database interface will continue processing or terminate the operation. To begin the operation the cursor slot must have the value UNKNOWN. If it is not UNKNOWN, the database operation terminates. Use the Reset operator in a rule or method to initialize the cursor slot as shown below.

```
reset         dummy_object.dummy_cursor
retrieve"     db_access_string"
```

Figure 7-7 shows an example of a Write window set-up for atomic processing. For example one record in a database is updated with the data from the slots of a single object. Although this example is oriented towards relational databases, it is also applicable to flat-file databases. Refer to Appendix A, "Database Integration Examples" of the Language Reference for additional atomic processing examples.
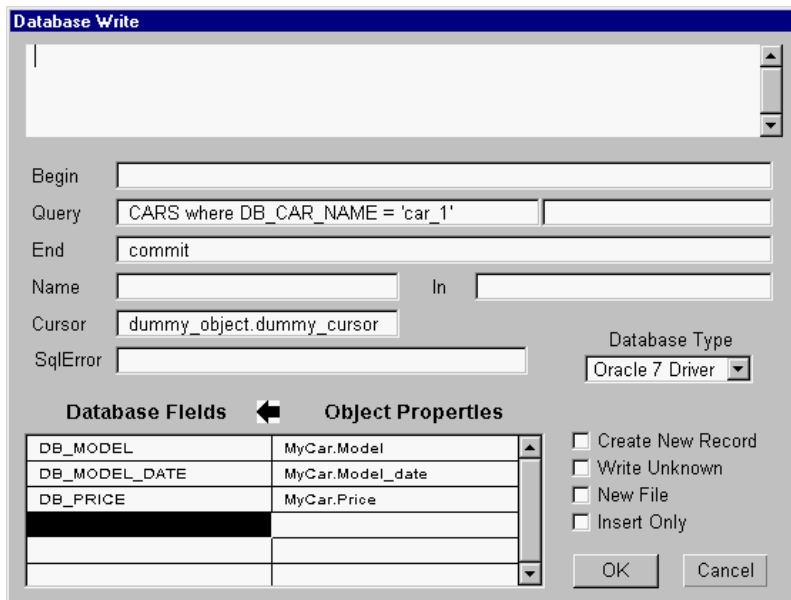
Figure 7-7 Atomic Write Operations

This is a description of each field in a single-record write operation:

1. For most databases this field should be left blank for atomic write operations. Some databases, such as Sybase, require a statement here.

    Begin

    If you are using a relational database, you can specify a statement in the `Begin` field. The statement is sent to the database server when the Retrieve or Write initiates.

    See the Begin topic or your database type in Chapter Three, "Database Integration Topics" of the Language Reference for details about the Begin field.

2. Select the type of database you're using in the Database Type window.

    Database Type

    ORACLE

    You can scroll the Database Type window to see all the types of databases supported by the database interface. Select the type that represents your database.

3. Specify table name and a unique record name that you want to update using a query. If more than one record meets the specifications of the query, only the first record is processed.

    Query   CARS where DB_CAR_NAME = 'car_1'

    If you are using a flat-file type database, Neuron Data provides a SQL-like query language which is described in Chapter Three, "Database Integration Topics" of the Language Reference. See the Query Language topic for details.

4. For Oracle and most other relational databases, this field should contain a `Commit` statement to make the changes to the table permanent, if the row is updated successfully. The statement is sent to the database server when the Write initiates.

    End   Commit

    See the End topic or your database type in Chapter Three, "Database Integration Topics" of the Language Reference for details about the End

field.

5. Indicate that this is a single-record transaction by specifying an integer slot with a value of UNKNOWN in the Cursor field.

Cursor ┌─────────────────────────────┐
       │ dummy_object.dummy_cursor   │
       └─────────────────────────────┘

The cursor will indicate whether the write operation was successful. The cursor slot must have a value of UNKNOWN for this operation to be treated as a single-record transaction. Make sure the cursor slot's value is set to UNKNOWN as explained in the introduction to this section.

6. Specify in the Rules Properties field the Rules Element slots whose values you want to send across the database interface, and specify the corresponding fields of the database record whose values you wish to update in the first column.

Database Fields ⬅	Object Properties	
DB_MODEL	MyCar.Model	▲
DB_MODEL_DATE	MyCar.Model_date	
DB_PRICE	MyCar.Price	

See the Slot Specification for Writes in Chapter Three, "Database Integration Topics" of the Language Reference for details about specifying these fields.

7. Do not select the Create New Record option. New records cannot be added to the database with atomic writes.

▢ **Create New Record**
▢ **Write Unknown**
▢ **New File**

8. If you want objects with a value of UNKNOWN to be processed, select the Write Unknown option. Otherwise, objects with a value of UNKNOWN are not processed.

▢ **Write Unknown**

Processing Records One at a Time

Use sequential processing when you need to process only one record at a time, or if you don't have enough memory to do group processing with a very large group. For example, when you process credit card transactions and are looking for unreported stolen credit cards, you only need to examine the current transaction to determine if it fits the profile of a suspicious transaction.

When you use sequential processing, you only have one Rules Element object in memory. As you step through the database, the contents of the current record are read into the object. The database interface evaluates a special slot you define called the cursor to determine whether the operation was successful. Initially, the value of the cursor slot must be set to 0 in a rule or method as shown in the sample rule below

Rule 1 (Hypothesis "BeginOperation")

```
Assign      0                dummy_object.dummy_cursor
Assign      ReadTable        ReadTable
```

The cursor will be set to a value > 0 if successful or a value < 0 if unsuccessful. A second rule or method can test the cursor slot's value to determine whether sequential processing should continue:

Rule 2 (Hypothesis "ReadTable")

```
>=          dummy_object.dummy_cursor
Retrieve    "db_access_string"
Reset       ReadTable
```

Figure 7-8 shows an example of a Retrieve window set-up for sequential processing. In this example data from multiple database records is read into the property slots of a single object one record at a time. The retrieve is invoked once for each record in the table. Although this example is oriented towards relational databases, it is also applicable to flat-file databases. Refer to Appendix A, "Database Integration Examples" of the Language Reference for additional sequential processing examples.

Note: Sequential Writes cannot be used with relational databases. Sequential Retrieves are permitted with all relational databases.

Figure 7-8   Sequential Processing Operations

This is a description of each field in sequential processing:

1. For most databases this field should be left blank for sequential retrieve operations. Some databases require a statement here.

If you are using a relational database, you can specify a statement in the Begin field. The statement is sent to the database server when the Retrieve initiates.

See the Begin topic or your database type in Chapter Three, "Database Integration Topics" of the Language Reference for details about the Begin field.

2. Select the type of database you're using in the Database Type window.

Database Type

ORACLE

You can scroll the Database Type window to see all the types of databases supported by the database interface. Select the type that represents your database.

3. Specify the group of records that you want to process using a query in the `Query` field.

Query   CARS where DB_PRICE >= 20000

Use the query to specify which records you want included in the group. If you are using a flat-file type database, Neuron Data provides a SQL-like query language which is described in Chapter Three, "Database Integration Topics" of the Language Reference. See the Query Language topic for details. If you want to process all records in a table, you do not need to specify a query.

Do not specify a complete database query in the `Query` field. If you are using SQL, only specify the tables and, if necessary, the where clause. For example, if your query is:

```
select DB_MODEL, DB_MODEL_DATE, DB_NAME from CARS where DB_SPORTIVE >= 'Yes'
```
only use the part of the query that is bold. Specify the database fields, such as `DB_MODEL`, under Database Fields. The database interface creates the rest of the query for you and sends it to the database.

4. For most databases, this field should be left blank. Some databases may require a statement here.

End

See your database type in Chapter Three, "Database Integration Topics" of the Language Reference for details about the End field.

5. Indicate that this is a sequential transaction by specifying an object slot in the Cursor field.

Cursor   dummy_object.dummy_cursor
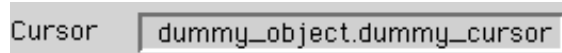
The cursor keeps track of the last record that was sent across the

database interface. The data type of a cursor slot must be an integer. Do not change the value of the cursor during sequential processing. Specify the fields in your database that you want to send across the database interface under Database Fields, and specify the corresponding slots in the Rules Properties field.

Database Fields ➡	Object Properties	
DB_MODEL	MyCar.Model	▲
DB_MODEL_DATE	MyCar.Model_Date	
DB_PRICE	MyCar.Price	

Use this field to tell the database interface which fields in your database you want to use. For each database field, you should specify a slot. For sequential processing, you specify the slots of one object. In this example, the object is `MyCar`.

See the Slot Specification for Retrieves in Chapter Three, "Database Integration Topics" of the Language Reference for details about specifying these fields.

6. If you want objects with a value of UNKNOWN to be processed, select the Retrieve Unknown option.

☐ **Retrieve Unknown**

Otherwise, objects with a value of UNKNOWN are not processed.

7. If you are using Retrieve, specify the forwarding strategy by selecting one of the following options: Always Forward, Current Forward, or Do Not Forward.

◇ **Always Forward**
◇ **Current Forward**
◇ **Do Not Forward**

Always Forward means that inference engine forwards the values in the rules network; Current Forward means that inference engine uses the current forwarding strategy; and Do Not Forward means inference engine does not use any forwarding strategy.

Retrieving a Group of Records

Use grouped processing when you need to use information provided by a group of records. A grouped retrieve operation reads multiple records in one operation. As the Rules Element processes each record, its fields are read into slots. All of the fields from a given record are read into the same object's slots -- "transforming" the record-field relationship into an object-property relationship.

Figure 7-9 shows an example of a Retrieve window set up for grouped processing. In this example data from multiple records in the database is retrieved into the property slots of a group of objects in a single operation. Although this example is oriented towards relational databases, it is also applicable to flat-file databases. Refer to Appendix A, "Database Integration Examples" of the Language Reference for additional grouped processing examples.

Figure 7-9 Grouped Retrieve Operation

This is a description of each field in a grouped record retrieve operation:

1. For most databases this field should be left blank for grouped retrieve operations. Some databases require a statement here.

Begin

If you are using a relational database, you can specify a statement in the Begin field. The statement is sent to the database server when the Retrieve or Write initiates.

See the Begin topic or your database type in Chapter Three, "Database Integration Topics" of the Language Reference for details about the Begin field.

2. Select the type of database you're using in the Database Type window.

Database Type

ORACLE

You can scroll the Database Type window to see all the types of databases supported by the database interface. Select the type that represents your database.

3. Specify the group of records that you want to process using a query in the Query field.

Query CARS where DB_SPORTIVE = 'Yes'

Use the query to specify which records you want included in the group. If you are using a flat-file type database, Neuron Data provides a SQL-like query language which is described in Chapter Three, "Database Integration Topics" of the Language Reference. See the Query Language topic for details. If you want to process all records, you do not need to specify a query.

Do not specify a complete query in the Query field. The Rules Element builds the query from the information you supply in this window. If you are using SQL, only specify the tables and, if necessary, the where clause. For example, if your query is:

```
select DB_MODEL, DB_MODEL_DATE, DB_NAME from CARS where DB_PRICE >= 20000
```

only use the part of the query that is bold. Specify the database fields, such as DB_MODEL, under Database Fields. The Rules Element creates the rest of the query for you and sends it to the database.

4. For most databases, this field should be left blank. Some databases may require a statement here.

End	

See your database type in Chapter Three, "Database Integration Topics" of the Language Reference for details about the End field.

5. Use the Name field to specify the object name to be used in the retrieve operation. Use the syntax `!fieldx!` to specify the object name, where `fieldx` should match the field name of the record whose values you want to retrieve.

Name	!DB_CAR_NAME!

Specific object names can be constructed by combining the field specified with a string constant. See the Object Names In Retrieve Operations topic in Chapter Three, "Database Integration Topics" of the Language Reference for details.

6. If you want only existing objects to be updated, you can specify a list of object names in the In field. The list can be defined explicitly by separating object names with commas or you can use pattern matching on a class to specify an object list.

In	<|cars_class|>

7. If you want the retrieve operation to create new objects for the fields defined by the Name field, you can specify a class to attach them to for inheritance purposes. To use the Link To field the Create Object option must also be selected.

Link To	|cars_class|

8. Under Database Fields specify the field names whose values will be retrieved. Specify the corresponding field names of the record(s) specified in the Name field whose values you want to retrieve in the first column, and specify the properties of the object(s) specified in the In field under Rules Properties.

Database Fields ➡	Object Properties
DB_MODEL	Model
DB_MODEL_DATE	Model_date
DB_PRICE	Price

In this example, the values of the slots formed by the three properties

Model, Model_Date, and Price belonging to objects in the class cars_class will retrieve data from records whose fields matched the Name field specification and the Query field. Alternately specific slot names may be given under the Rules Properties field.
See the Slot Specification for Retrieves in Chapter Three, "Database Integration Topics" of the Language Reference for details about specifying these fields.

9. You can specify that dynamic objects are created by selecting the Create Object option. If you specified a class name in the Link To field, the new objects inherit the properties of that class.

☐ **Create Object**

When retrieving database records, the Rules Element first tries to find a matching object. If unsuccessful, and if you selected the Create Object option, the Rules Element creates a dynamic object using the contents of the database record.

10. If you want objects with a value of UNKNOWN to be processed, select the Retrieve Unknown option. Otherwise, objects with a value of UNKNOWN are not processed.

☐ **Retrieve Unknown**

11. Specify the forwarding strategy by selecting one of the following options: Always Forward, Current Forward, or Do Not Forward.

◇ **Always Forward**
◇ **Current Forward**
◇ **Do Not Forward**

Always Forward means that the inference engine forwards the values in the rules network; Current Forward means that the inference engine uses the current forwarding strategy; and Do Not Forward means the inference engine does not use any forwarding strategy.

Writing a Group of Records

A grouped write will write multiple object's slots in one operation. All of the slots written to a given record come from the same object, transforming the Rules Element's object-property relationship to a record-field relationship in the database.

Figure 7-10 shows an example of a Write window set up for grouped processing. In this example data from the slots of two objects is written to the database in a single operation. Each object is written as an individual record. Although this example is oriented towards relational databases, it is also applicable to flat-file databases. Refer to Appendix A, "Database Integration Examples" of the Language Reference for additional grouped processing examples.

Figure 7-10 Grouped Write Operation

This is a description of each field in a grouped record write operation:

1. For most databases this field is to be left blank for grouped write operations. Some databases, such as Sybase, require a statement here.

Begin

If you are using a relational database, you can specify a statement in the Begin field. The statement is sent to the database server when the Write initiates.

See the Begin topic or your database type in Chapter Three, "Database Integration Topics" of the Language Reference for details about the Begin field.

2. Select the type of database you're using in the Database Type window.

Database Type

ORACLE
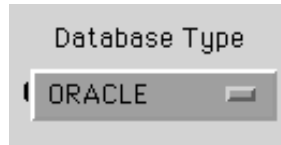
You can scroll the Database Type window to see all the types of databases supported by the database interface. Select the type that represents your database.

3. Specify the table to which the records are to be written in the Query field. For flat-file databases this field must be left blank .

Query   CARS

You may also use the Query field to specify a full query, although this operation is not usually performed with Grouped Writes.

4. If you are initiating a write operation to a relational database, you can specify a statement in the End field. You can use the End field to send a Commit or Rollback statement to the database server. The statement is sent to the database server when the Write initiates.

End   Commit

See the End topic or your database type in Chapter Three, "Database Integration Topics" of the Language Reference for details about the End field.

5. Use the Name field to determine which records the database interface will update. To accomplish this the syntax `!fieldx!` is used to specify

how to parse object names, where `fieldx` is the value of the field that matches an object name. When a match is found, a record "key" is formed that specifies which records should be written. See the Record Specification for Writes topic in Chapter Three, "Database Integration Topics" of the Language Reference for details.

Name	!DB_CAR_NAME!

This field essentially builds the Select statement of a Query. If your object names do not correspond directly to field values, you can divide the object name into a string component and a field component. See the Object Names In Retrieve Operations topic and the Name topic in Chapter Three, "Database Integration Topics" of the Language Reference for details.

6. If you want to restrict the Name field to a list of object names to parse, you can specify a list of object names in the In field. The list can be defined explicitly by separating object names by commas or you can use pattern matching on a class to specify an object list as shown below.

In	<|cars_class|>

7. Use the Rules Properties field to specify the slot or property names whose values will be written. Specify the properties of the object(s) specified in the In field under Rules Properties, and specify the corresponding field names of the record(s) specified in the Name field whose values you want to update in the first column.

Database Fields ⬅	Object Properties
DB_MODEL	Model
DB_MODEL_DATE	Model_Date
DB_PRICE	Price

In this example, the values of the slots formed by the three properties `Model`, `Model_Date`, and `Price` belonging to objects in the class `cars_class` will update records whose fields matched the Name field "key." Alternately specific slot names may be given under the Rules Properties field.

See the Slot Specification for Writes in Chapter Three, "Database Integration Topics" of the Language Reference for details about specifying these fields.

8. You can specify that new records may be added to the database by selecting the Create New Record option.

☐ **Create New Record**
☐ **Write Unknown**
☐ **New File**
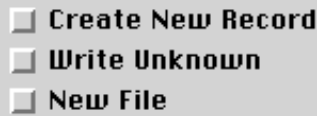
When the Rules Element is writing objects to your database, it first tries to find an existing record for each object. If a matching record does not exist, and if you have selected the Create New Record option, the Rules Element creates a new record.
You must select this field for new records to be added to the database. If you are using flat files, you can select the New File option to create a whole new file.

9. If you want objects with a value of UNKNOWN to be processed, select the Write Unknown option. Otherwise, objects with a value of UNKNOWN are not processed.

☐ **Write Unknown**

8 *Application Delivery*

This chapter describes the facility for making knowledge base files secure from inspection. The utility, called the Application Specific Toolkit (AST), currently provides the ability to encrypt a knowledge base. A special version of the Intelligent Rules Element will be required or the user will be prompted for a password.

Overview

The Application Specific Toolkit (AST) is a utility which lets you encrypt and decrypt knowledge bases. This capability lets you deliver knowledge bases to end users that cannot be viewed or modified.

Both text knowledge bases (TKBs) and compiled knowledge bases (CKBs) can be encrypted. Similar to the case with non-encrypted knowledge bases, encrypted TKBs are portable to other machines while CKBs are not portable.

When you encrypt a knowledge base it becomes password protected. This means that if the appropriate password isn't supplied, the file will be completely unreadable to the Rules Element, the runtime library, any text editor, or any other application for that matter.

The password can be supplied from one of two places:
- In the application
- From the user.

The developer can also decide to use any combination of the above password locations to create a "super-password" which requires different components from different locations.

There are different reasons for keeping the password in different places. If the application supplies the password, this ensures that no other application can use the knowledge base. This allows you to "lock" the knowledge base to one particular application. If the user supplies the password, it ensures that only people who have knowledge of the password can use the application. This lets you control who uses the application.

Currently, you must use the Data Encryption Standard (DES) to encrypt your knowledge base. In the future, you will be able to use another approach supplied by Neuron Data as well as create your own encryption algorithm. The Data Encryption Standard (DES) is a very secure approaych.

Using the Application Specific Toolkit

The Application Specific Toolkit (AST) is located in the utilities directory of the UNIX or VAX platforms. For PC or Macintosh users, the AST is a standalone application (AstGfx) located in the Bin and Util directories respectively. There are three files:

- `ast.exe` (VMS) or `ast` (UNIX) which is the executable file
- `ast.obj` (VMS) or `ast.o` (UNIX) which is an object file that you can link to your own routines
- `ast.c` (VMS and UNIX) which is the c source code

To run the executable file "`ast.exe`", type "`ast`" at the prompt:

```
$ run ast (or % ast from UNIX)
```

The following menu will come up:

```
Choose one of the following:
    0. Exit
    1. Encrypt file
    2. Decrypt file
Please enter your choice:
```

> **Note:** PC and Macintosh users should locate the AstGfx application in the Bin and Util directory respectively and double-click on the AstGfx icon to start it. See the help on-line for additional information.

Exiting

Typing 0 at the main menu depicted above will exit the AST application and return you to the default environment.

Encrypting a Knowledge Base

Typing 1 at the main menu depicted above will give the following prompt:

```
Enter name of input file:
```

to which you supply the name of the file that you want to encrypt. You are next prompted with:

```
Enter name of output file:
```

to which you supply the name of the encrypted file. The final prompt is:

```
Enter password for output file:
```

where output file is replaced by the filename given earlier. The password can be any series of alphanumeric characters. Your password will be case sensitive, so be careful to use the same upper and lower case letters when

typing the password. You will not be prompted a second time for verification, so you must be sure to type precisely what you want. There cannot be any spaces in the password.

Decrypting a Knowledge Base

Typing 2 at the main menu depicted above will produce a similar set of prompts. First, it asks:

```
Enter name of input file:
```

to which you enter the name of the encrypted file which you want to decrypt. The next prompt is:

```
Enter password for <input file>:
```

to which you supply the password. The last prompt asks you for the name of the new decrypted version of the knowledge base which you wish to produce:

```
Enter name of output file:
```

Loading an Encrypted Knowledge Base

When the Rules Element loads an encrypted knowledge base, it passes whatever password it has determined is appropriate to the DES algorithm which then decrypts the knowledge base using the password. If the password is wrong, the loaded knowledge will be scrambled and the Rules Element will produce the appropriate warnings.

It is also important to note here that if the wrong password is used, the application programming interface (API) will only provide access to whatever is loaded, not the decrypted version of it.

Getting a Password from the User

When a knowledge base has been encrypted, the default strategy the Rules Element uses is to prompt the user for the password before loading the knowledge base. The strategy the Rules Element uses to prompt depends on whether it is the development system or runtime system which is running, and on whether the developer has installed a handler to customize how the Rules Element prompts (for more details on Handlers, see the section below).

Getting a Password from the Application

By installing an appropriate handler, the Rules Element will look for the password in another location instead of prompting for it. It can generate the

password from the application itself or it can use any environment variables. You can also generate the password from a variety of sources: the user provides part of it, environment variables provide part, and let your application generate the rest.

Password Prompting from the Development System

When a knowledge base which has been password protected from the user is loaded from the development system, the following window comes up:
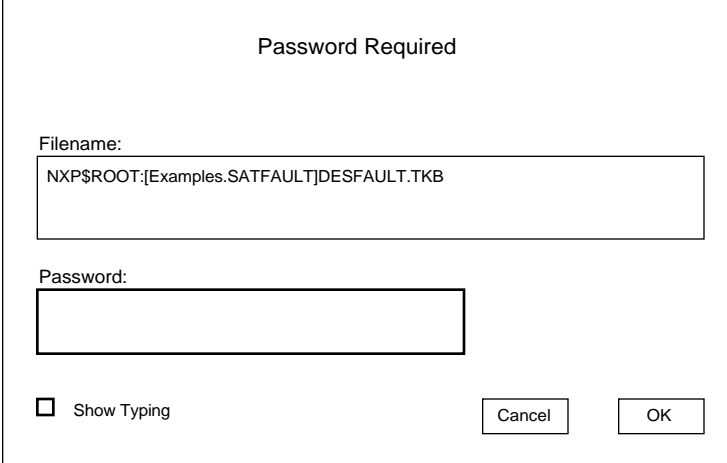
```
                        Password Required


    Filename:
     NXP$ROOT:[Examples.SATFAULT]DESFAULT.TKB



    Password:



    ☐  Show Typing                        Cancel        OK

```

Figure 8-1   Password Entry Screen

The password is entered in the appropriate box. By default, the Rules Element will not echo the password you type. If you wish the Rules Element to show you what you are typing (or have typed), check the "Show Typing" box.

Password Prompting from the Runtime System

The runtime system runs in character-based mode by default. The password prompt for this is simply:

```
$ Enter password for <filename>:
```

Password Handler

The only current AST handler is NXP_PROC_PASSWORD. In the future, Neuron Data will also support two other handlers: NXP_PROC_ENCRYPT and NXP_PROC_DECRYPT.

NXP_PROC_PASSWORD

NXP_PROC_PASSWORD is an option in the NXP_SetHandler call. When the Rules Element loads an encrypted knowledge base which has been password protected, it will prompt for the password by default. You may specify an alternate procedure to be called to provide the password. This procedure may then customize the way the user is queried, base the password on the filename, get it from an environment variable, retrieve it from a database, simply provide the password, or whatever else you want.

The procedure is called with a 255 character "string" buffer (descriptor for Fortran). The user should return the password in this buffer. It must be NULL or "0" terminated. The format is:

```
NXP_Set_Handler(NXP_PROC_PASSWORD, myPassword, (char*)0)
```

The password handler is then provided as in the following example:

```
int     myPassword( filename, password );
char    *filename;
char    *password;
{
        print(" providing password for: %s\n", filename );
        strcpy( password, "hello" );
        return 1;
}
```

In this example, the filename is merely printed out for information. Other possibilities include using it to derive the password, or simply ignoring it. The password is hard-coded in the above example for all cases to be the string "hello." Note that this example is not "secure," in that an individual might be able to browse the executable image, notice the word "hello," and gain an edge in breaking your password. In other words, this example, while it illustrates the principle, is not ideal if high levels of security are desired.

Summary

In summary, you supply the AST with any TKB or CKB along with a password and the AST encrypts the knowledge base using the password as a parameter:

TKB or CKB ⟶

AST

Encrypted TKB or CKB
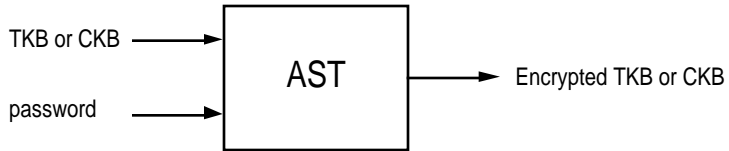
password ⟶

Similarly, you can supply the AST with an encrypted knowledge base, along with the password which tells the AST how to decrypt the knowledge base, and the AST produces the decrypted version of the knowledge base:

Encrypted TKB or CKB ⟶

AST

⟶ TKB or CKB

password ⟶
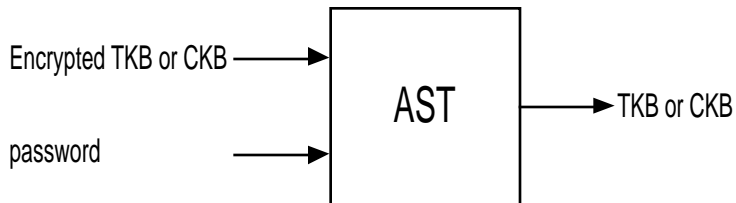
Figure 8-2   Application Specific Toolkit Functions

When the Rules Element receives the LoadKB command, whether it is from the user interface, from a rule or method, or from the API, the Rules Element must first determine whether or not the knowledge base is encrypted. If it isn't encrypted, the Rules Element can just load it. If it is encrypted, then the Rules Element must determine the password, pass the password as a parameter to the DES algorithm, and finally load the knowledge base.

Figure 8-3 depicts the decision tree the AST follows in order to load knowledge base files.
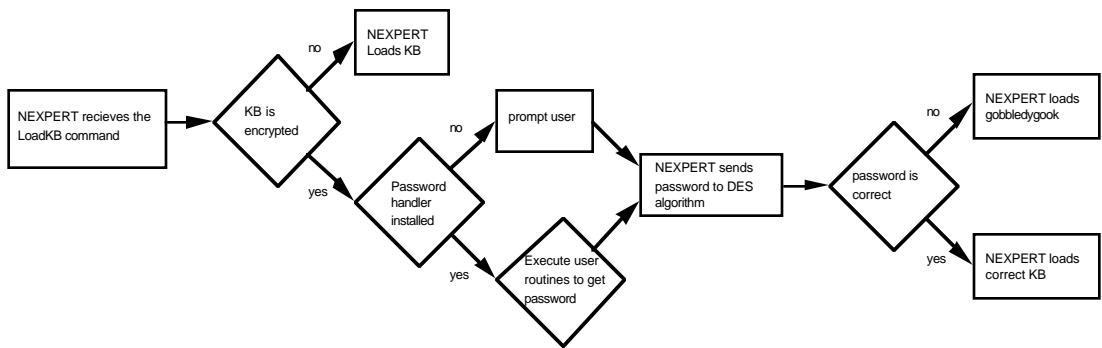
Figure 8-3   Application Specific Toolkit (AST) Decision Tree

A *Main Menu Options*

The main menu bar in the Intelligent Rules Element provides option menus that you can select to develop your application and control the windowing environment. The main menu bar appears along the top of the window and provides the same set of options for every platform.

> **Note:** In the Macintosh environment the main menu is the familiar bar that runs across the top of the screen. Click on the menu name itself to display the desired options menu.

File Menu

The options provided by the File menu control the appearance of the Rules Element windowing environment.

New Text File

This command opens a new, empty window on the screen. Text can be typed in and edited, using the basic Cut, Copy, and Paste commands. The window's default title is "Text #N," where N-1 equals the number of Text windows previously opened. See the `Save Text As...` command to change the title and save the contents in a file.

Open Text File...

This command opens an existing file of type TEXT in a window. It brings up the standard Macintosh file dialog box and lets you select a drive, directory, and a file. The text file can not be larger than 32K. This command lets you read or modify a text knowledge base file. Since the system loads KB files in read-only mode, you can keep your text window open while reading its contents in the windowing environment. It can also be used for editing related knowledge base files such as reports, comments, or why files that will be used in rules or Show operations.

Close

This command closes the active (or front-most) window. If the window is a text window that has changes, you can choose to save it first.

Save Text File

This command saves the text in the file attached to the active window. It is equivalent to the `Save Text As...` command if the window's contents have not already been saved.

Save Text As...

This command brings up the standard Macintosh dialog box to modify the default file name attached to the active window. The file name of predefined windows such as Transcript, Current, Rule, Conclusions, ect. cannot be changed.

Set up Environment

This command opens a window that gives many session control options. You can change the settings for the current session or record them to a file for later sessions. See the `Save Environment` command.

Save Environment

This command records the current environment settings to a file including the size and position of open windows. You can access the same environment on your next session. You can overwrite previous settings if you want to update the environment settings file.

Page Setup...

This command lets you set the paper size, orientation, and certain effects for printing. On the Macintosh it brings up the dialog box for the Laser Printer, depending on the printer currently configured for the Macintosh.

Print

This command lets you print the contents of any active window in the Rules Element windowing environment. On other hardware platforms this command is available as a popup menu option for specific windows.

Quit

This command exits the Rules Element application and returns you to the platform's operating system. You can save session changes before exiting to the operating system.

Edit Menu

The options provided by this menu let you invoke the Rules Element editors to create your application's knowledge structures. Chapter Two, "Application Implementation" addresses these menu options. Additionally, the Rules Element's Edit menu shows text editing commands.

Cut

This command can be used in the text-edit line of the editors and Text windows. It deletes the current selection, and keeps it in a clipboard for later use with the `Paste` command. The `Cut` command also deletes a block of selected fields in the Rule or Method editor windows.

Copy

This command can be used in the text-edit line of the editors and Text windows. It copies the current selection without affecting it, and keeps it in a clipboard for later use with the `Paste` command. The `Copy` command also copies a block of selected fields in the Rule or Method editor windows.

Paste

This command can be used in the text-edit line of the editors and Text windows. It pastes the text that was previously cut or copied, replacing the current selection. The `Paste` command also puts back a block of selected fields previously copied in the Rule or Method editor windows.

Clear

This command can be used in the text-edit line of the editors and Text windows to delete the current selection. It does not overwrite the current contents of the clipboard; instead, use the `Cut` command to place a selection into the clipboard. The `Clear` command also deletes a block of selected fields in the Rule or Method editor windows.

Rule

This command opens an edit mode window for strong inferencing links. You can display, create, modify, copy, and delete rules and inferencing links of the current knowledge base.

Context

This command opens an edit mode window for weak inferencing links. You can display, create, modify, copy, and delete weak links between knowledge island links of the current knowledge base.

Object

This command opens an edit mode window for objects and related knowledge structures. You can display, create, modify, copy, and delete objects and subobjects of the current knowledge base.

Class

This command opens an edit mode window for classes and related knowledge structures. You can display, create, modify, copy, and delete classes and subclasses of the current knowledge base.

Property

This command opens an edit mode window for unattached properties. You can display, create, modify, copy, and delete properties of the current knowledge base.

Meta Slot

This command opens an edit mode window for system attributes of properties that belong to a object or class. You can display, create, modify, copy, and delete system attributes of the current knowledge base.

Method

This command opens an edit mode window for methods that belong to a slot, object, property, or class. You can display, create, modify, copy, and delete methods of the current knowledge base.

Script

This command opens the Script Editor with the currently loaded application script displayed in the script template area. The changes you make with the Script Editor must be saved with the Save... Script... command.

Expert Menu

The options provided by this menu let you initiate and control your application's inferencing session. Chapter Four, "Application Processing" addresses these menu options.

Volunteer

This command directly volunteers data you first select, from the Data Notebook for example.

Suggest

This command directly suggests hypotheses you first select, for example, from the Hypotheses Notebook.

Suggest | Volunteer...

This command opens a dialog window that gives options to start inferencing with any combination of suggest, volunteer, or send message preselected. You can select data, hypotheses, and messages to initiate knowledge processing from the dialog window.

Run... Knowledge Base

This command lets you display the sub-menu options that you can use to start an application from the Rules Element development environment. The Knowledge Base option causes the Rules Element system to start inferencing based upon the currently selected hypotheses and/or data. You can initiate an inferencing session or reinitiate inferencing using the modified values of a previous session. (Note: This is equivalent to the Knowcess command from previous versions.)

Run... Application Script

This command lets you display the sub-menu options that you can use to start an application from the Rules Element development environment. The Application Script option initiates application processing using the currently loaded application script. The application script usually specifies application startup characteristics and can initiate application processing with or without a GUI file. To view the currently loaded application script, select the Edit Application Script command.

Restart Session

This command sets the values of all knowledge structures of the current knowledge base to "UNKNOWN." It also erases all inferencing session constructs including: conclusions reached, data gathered, and dynamic objects and links. You can clear the current knowledge base for further inferencing or terminate knowledge processing of a current session.

Agenda Monitor

This command opens an interactive window that lets you monitor inferencing. It shows the the queuing and dequeuing of hypotheses for evaluation by the system. You can select hypotheses to suggest, place breakpoints on hypotheses, and initiate knowledge processing from this window.

Strategy

This command opens a dialog window that gives the Rules Element inferencing and inheritance control options. You can modify the default parameters for the current inferencing session or save them for later inference sessions.

Load..
Knowledge Base, **Script / Resource**

This command lets you display the sub-menu options that you can use to display a file selection dialog window to load knowledge base files, application script files (to specify application startup characteristics), or GUI resources from the current directory. You can change directories to display additional files or you can load a file. The system automatically compiles knowledge base files saved in text format when loaded.

Save...
Knowledge Base, **Script / Resource**

This command lets you display the sub-menu options that you can use to save the current knowledge base file, newly created or the last loaded application script file, or newly created GUI resources. You can save a new file or rename an existing one. This command does not clear the saved files from system memory; see the Clear... command.

Note: You can also save the knowledge base file in the compiled format for faster loading on the same platform.

Clear...
Knowledge Base, **Script / Resource**

This command lets you display the sub-menu options that you can use to display the list of currently loaded knowledge base files, application script file, or GUI resources. You can clear one or all of the files from system memory. This command does not save changes made to these files; see the Save... command.

Set Knowledge Base

This command opens a dialog window that shows the list of currently loaded knowledge base files. You can change the current knowledge base for editing or inferencing when using multiple knowledge bases.

Browsers Menu

The options provided by this menu pertain to the graphical windows only. Chapter Five, "Application Testing" addresses their application debugging capabilities. Chapter Two, "Application Implementation" addresses their static display capabilities.

Rule

This command opens an interactive window that gives graphic display options for rules. You can follow the inferencing pathways or display individual rules during an inferencing session. You can also display the rule network statically.

Object

This command opens an interactive window that gives graphic display options for objects and related knowledge structures, including classes, properties, methods, and meta-slots. You can follow the inheritance pathways or display individual knowledge structures during an inferencing session. You can also display the object network statically.

Cross Reference

This command opens an interactive window that lets you view the interrelationships between knowledge structures in the application. Related items are depicted as components of the same network branch. Individual items displayed in the network can be viewed in the rule or object networks.

Resource

This command opens an interactive window that is the main window of the graphical user interface builder called the Resource Browser. It gives access to a window editor, script editor, and individual graphical user interface editors.

List Of...

This command lets you display the sub-menu options that you can use to display categorized and updated information about the application knowledge structures. This includes separate options for rules, hypotheses, data, objects, classes, properties, and methods.

Reports Menu

The options provided by this menu let you analyze the events of inferencing during and after knowledge processing. Chapters Five and Six of this guide address these menu options.

Case Status

During or after an inferencing session use this command to open a text window that shows all processed data and hypotheses. It also shows their values. You can view the values of relevant data and hypotheses.

Full Report

During or after an inferencing session use this command to open a text window that shows all processed hypotheses, their values, and the state of all the rules leading to them. You can view the justifications of an inference session.

Transcript

During an inferencing session use this command to open a text window that dynamically shows inference engine events. You can then enable the window to keep a record of the current inferencing session and to track the inferencing pathways.

Current Rule

During an inferencing session use this command to open a text window that dynamically shows the rule currently under evaluation. You can then enable the window to view a textual format version of the current rule.

Current Hypothesis

During an inferencing session use this command to open a text window that dynamically shows the hypothesis currently under evaluation. You can then enable the window to follow the inference engine's precise focus of attention.

Conclusions

During an inferencing session use this command to open a text window that dynamically shows the status of hypotheses the system evaluates. You can then enable the window to keep a record of the inferencing conclusions.

Journal...

Either before or during an inferencing session use this command to open a dialog window that gives options to record or save an inferencing session. You can later replay or restore the recorded session.

B *Popup Menu Options*

This appendix lists and describes every popup menu option available in the Rules Element. It serves as a reference section that lets you get a concise description of a particular option before using it. All options appear in alphabetical order in this appendix.

The popup menus you access through the windowing environment provide additional functionality at all levels of interaction with the Rules Element system. The commands that you can select belong to the following general categories.

Session Control Panel. These commands, which you access through the Main Window Session Control Panel popup menu, let you interact with the Rules Element system during an application processing session. The options displayed depend on whether a rule is currently under evaluation or not.

Application Design. These commands include a wide array of operators that you select for the Operator field of the Rule editor, Object editor, Method editor, and Meta-Slot editor. Additionally, copy commands exist to facilitate design efforts. The options listed depend on which field of these editor windows you select. The system displays only valid operators for a given editor window.

Environment Control. These commands make up the majority of the popup menu options that you select from the windowing environment. They include commands for displaying and editing rule and object structures from the currently active window, as well as commands to initiate inference engine processing.

Refer to Chapter One, "The Windowing Environment" for further background information about popup menus. Additionally, entries in the index of this manual identify where to find procedural information related to popup menu options.

Option Descriptions

Descriptions of the popup menu options follow. All options belong to one of three popup menu windows: a local, global, or Windows popup menu. Options that are not operators are usually global popup menu options unless noted otherwise. Those options called "operators" belong to the local

popup menus you display for certain fields of a window. Locate the desired option by name in the alphabetical listing.

> **Note:** For more information about popup menu options that are "operators," refer to the Language Reference manual.

<, >, <=, >=

These operators compare the value of slots or expressions and constant values. Used in conditions of the Rule editor.

<>, =

These operators compare the values of slots or expressions with a list of constant values. Used in conditions of the Rule and Method editor.

Apropos

1. This command displays a file during application processing that is associated with the slot currently under investigation. The file is any graphic or text file of the developer's design. Available from the main window session control panel. This command is not available if the session has been interrupted.

2. This command is also available from the Object, Property, Context, and Meta-Slot editors for a slot displayed in the current window.

3. This command is also available from the network window's local popup menu for an item you select. It provides the same function as the window's document icon.

AskQuestion

This operator prompts the end user during application processing for the value of a slot. Used in the Order of Sources method of the Method editor.

Assign

This operator assigns the value of an expression, a constant to a slot of any type. Values that this operator acts on may be forwarded according to the current strategy. Used in conditions and actions of rules and methods.

Backward

This operator triggers backward chaining from a given slot to determine a value. Used in the Order of Sources method of the Method editor.

Cancel

This command lets you abort the current editing mode without writing the rule or object structure to the application. Available from the Rule, Method, Object, Class, Property, Context, and Meta-Slot editors. Provides the same function as the editor window's Cancel button.

Change KB

This command lets you assign the currently displayed rule or object structure to a new knowledge base file. Available from the Rule, Method, Object, Class, Property, Context, and Meta-Slot editors.

Change Settings

This command displays the network dialog window used to customize parameters of the network diagram. Available from the network windows.

Change to Class

This command lets you transform a structure listed in the application as an object into a class. Available from the Object editor.

Change to Object

This command lets you transform a structure listed in the application as a class into an object. Available from the Class editor.

Change Value Type

This command lets you reassign the data type of the currently displayed slot. Available from the Meta-Slot editor.

Clear

This command erases the current network diagram from the network windows. Available from the network windows.

This local popup menu command lets you erase a single field from the editor windows. Available from Rule, Method, Object, Class, Property, and Context editors.

Close

A standard window command that closes the currently active window. See also Print, Push Behind, and Write to File.

Copy

This local popup menu command lets you copy existing information from the application into the fields of the various editor windows. This command depends on the currently selected field. Options for copying include hypotheses, data, classes, objects, functions, and files. Available from Rule, Method, Object, Class, Property, and Context editors.

This global popup menu command lets you copy an entire rule or object structure for editing in the editor windows. The structure copied depends on the current editor window. Available from Rule, Method, Object, Class, Property, Context, and Meta-Slot editors. Provides the same function as the editor window's Copy button.

CreateObject

This operator creates objects and/or links between objects and/or classes. Used in methods (Method editor), or in conditions and actions of rules. See also DeleteObject.

Current Rule, Current Object

This command opens or activates the network window during application processing. The window automatically focuses on the rule or object currently under evaluation. Available from the main window session control panel.

Delete

This command lets you delete the currently displayed rule or object structure from the application. Available from Rule, Method, Object, Class, Property, Context, and Meta-Slot editors, as well as the List of Rules window. Provides the same function as the editor window's Delete button.

Delete Line

This local popup menu command lets you delete a single line from the editor windows. The cursor position determines the line the system deletes. Available from the Rule, Method, Object, and Class editors. See also Insert Line.

DeleteObject

This operator deletes objects and/or links between objects and/or classes. Used in methods (Method editor), or in conditions and actions of rules. See also CreateObject.

Disable Write

This command prevents the system from writing inside the concerned window. Available from the Transcript, Current Hypothesis, Current Conclusions, and Current Rule windows. See also Enable Write.

Display All

This command clears the network windows and displays the complete network diagram for rules or objects. Available from the network windows. The window's current extension mode affects the appearance of the display; see the Extension Mode icon description in Appendix C.

Edit xxx, Edit...

This command displays the corresponding editor window for the currently selected rule or object structure. Available from the network windows, list windows, and certain editor windows.

Enable Write

This command lets the system write inside the selected window. Available from the Transcript, Current Hypothesis, Current Conclusions, and Current Rule windows. See also Disable Write.

Erase

This local popup menu command removes a rule or object structure previously displayed in the network diagram. It has no effect if the rule or object structure was not the originator of any extension. Can also erase the entire object network if applied to the root structure. Available from the network windows, provides the same function as the window's Eraser icon.

Execute

This operator initiates a Rules Element execute library function, a user-written routine, or program. Used in methods (Method editor), or in conditions and actions of rules.

Extend Left, Right

These local popup menu commands let you investigate inferencing and inheritance pathways. Displays the related rule or object structures when available. For an object it shows methods, properties, subobjects, subclasses, and classes. For a rule it shows either hypotheses or entire rules. Available from the network windows, provides the same function as the window's Arrow icons.

Focus

This local popup menu command re-focuses the network diagram on the selected rule or object structure. It permits you to display a new path for investigation independent from the one currently displayed. Available from the network windows; provides the same function as the window's Magnifying Glass icon.

Focus Object Network

This command lets you view the object network diagram for the currently displayed slot. Available from the Rule Network window, Object editor, Property editor, Class editor, as well as the List of Methods, Objects, Classes, and Properties windows.

Focus Rule Network

This command lets you view the rule network diagram for the currently displayed data or hypothesis. Available from the Object Network window, Rule editor, Context editor, as well as the List of Rules, Data, and Hypotheses windows.

Full Left Extent, Full Right Extent

This local popup menu command lets you investigate inferencing and inheritance pathways. Reveals the complete pathway in the direction of the arrow cursor. Available from the network windows. The window's current extension mode affects the appearance of the display; see the Extension Mode icon description in Appendix C.

InhMethod

This operator triggers the automatic inheritance of methods for a given slot. Used in the methods of the Method editor. See also NoInherit.

InhValueDown, InhValueUp

This operator triggers the automatic inheritance of the slot value from a parent or child. Used in the methods of the Method editor.

Insert Line

This local popup menu command lets you insert a single line between existing entries for the editor windows. Available from the Rule, Method, Object, and Class editors. See also Delete Line.

Interrupt

This operator suspends an application processing session. Used in the methods of the Method editor.

Journal...

This command displays the journaling control window during application processing. The journaling mechanism is used to record and replay a session, or to save and restore a state. Available from the main window session control panel.

Knowcess

This command lets you start application processing based on previously suggested hypotheses and/or volunteered data. Available from the Windows popup menu of the active window and the global popup menu of the main window session control panel.

List of Rules

This command displays the List of Rules window for the currently displayed rule. Available from the Rule editor.

LoadKB

This operator loads or enables a knowledge base file during application processing. Used in methods (Method editor), or in conditions and actions of rules. See also UnLoadKB.

Member

This operator tests whether a given object or list of objects belongs to another list of objects. Used in the conditions of the Rule and Method editors. See also NotMember.

Modify

This local popup menu command lets you modify a previously volunteered value and revolunteer the new value for application processing. Available from the List of Data, Class, and Object windows, as well as the Rule Network window. See also Volunteer.

This global popup menu command lets you modify an entire rule or object structure for editing in the editor windows. The rule or object structure modified depends on the current editor window. Available from Rule, Method, Object, Class, Property, Context, and Meta-Slot editors. Provides the same function as the editor window's Modify button.

This command is also available from the List of Rules window to display the current rule for editing in the Rule editor window.

New

This command lets you create an entirely new rule or object structure starting from an empty editor window. Available from Rule, Method, Object, Class, Property, Context, and Meta-Slot editors. Provides the same function as the editor window's New button.

This command is also available from the List of Rules window to display the current rule for editing in the Rule editor window.

No

This operator tests boolean data (statements that are either true or false). They are usually used to test the logical states of subgoal hypotheses. Used in the conditions of the Rule and Method editor. See also Yes.

NoInherit

This operator prevents inheritance of values and methods from occurring for a given slot. Used in the methods of the Method editor. See also InhMethod.

NotMember

This test operator determines whether a given object or list of objects does not belong to another list of objects. Used in the conditions of the Rule and Method editor. See also Member.

OK

This command compiles and verifies the syntax of the currently displayed rule or object structure. Available from the Rule, Method, Object, Class, Property, Context, and Meta-Slot editors. Provides the same function as the editor window's OK button.

Overview

This command invokes the Overview window for the current network diagram. Available from the network windows.

Print

A standard window command that displays the print options window. You can print a single screen or all rule or object structures represented by the current window. See also Close, Push Behind, and Write to File.

Push Behind

A standard window command that lets you send the active window behind the second window on the screen. This is convenient for accessing windows hidden by larger windows such as the network windows. See also Close, Print, and Write to File.

Reset

This operator causes the specified slot to return to the UNKNOWN state. Used in methods of the Method editor, or in conditions and actions of rules. Also available on a slot in the Object Network window or a hypothesis in the Rule Network window.

Restart Session

This command resets all attributes and logical states in the current knowledge base to the UNKNOWN state. Available from the main window session control panel and the Windows popup menu.

Retrieve

This operator retrieves information from a database or data file. Used in methods (Method editor), or in conditions and actions of rules. See also Write.

RunTimeValue

This operator provides a default value for a given slot during application processing. Used in the Order of Sources method of the Method editor.

Select

This local popup menu command lets you preselect one or several object structures for application processing. This is a preliminary step to suggest/volunteer. Available from the List of Data and Hypotheses windows. See also Unselect.

Setup Colors

This command displays the Setup window that lets you customize the colors of your environment. Available from the main window session control panel and the global popup menu, if you are using a color monitor.

Setup

This command displays the Setup window that lets you control the appearance of certain windows during an application processing session.

Available from the global popup menu of the main window session control panel.

Show

This operator displays a static graphic or static text file on the screen during application processing. Used in methods (Method editor), or in conditions and actions of rules.

Show Class editor

This command displays the Class editor window while creating objects to inspect, create, modify, or delete classes. Available from the Object editor.

Show Object editor

This command displays the Object editor window while creating classes to inspect, create, modify, or delete objects. Available from the Class editor.

Strategy

This operator modifies the current inferencing strategies during application processing. Used in methods (Method editor), or in conditions and actions of rules.

Suggest

This local popup menu command directly suggests a hypothesis you select for application processing. Available from the List of Hypotheses window and the network windows. See also Unsuggest.

This command also displays the Suggest/Volunteer dialog window preset to suggest mode. Available from the main window session control panel and Windows popup menu.

Suggest Hypothesis

This command lets you suggest the currently displayed rule for application processing. Available from the Rule editor and List of Rules window. See also UnSuggest Hypothesis.

UnloadKB

This operator unloads or disables a knowledge base file. Used in methods (Method editor), or in conditions and actions of rules. See also LoadKB.

UnSelect

This command lets you prevent previously selected object structures from undergoing application processing. Available from the List of Hypotheses window. See also Select.

UnSuggest

This command lets you remove the previously suggested hypothesis from the list for application processing. Available from the List of Hypotheses window. See also Suggest.

UnSuggest Hypothesis

This command lets you remove the previously suggested hypothesis from the list for application processing. Available from the Rule editor and List of Rules window. See also Suggest Hypothesis.

View Line

This command opens the View Line window in order to display full strings for the currently selected rule or object structure in the network windows. Available from the network windows.

Volunteer

This local popup menu command directly volunteers the value of a selected object structure. Available from the List of Data, Object, and Class windows. See also Modify.

When used with the Object Network window, this command initiates application processing.

Why

This command displays a window during application processing that provides a reasoning explanatory mechanism for the current rule. Available from the main window session control panel.

Write

This operator updates a database or data file. It can also create a data file. Used in methods (Method editor), or in conditions and actions of rules. See also Retrieve.

Write to File

A standard window command that saves the contents of the current window in a specified file. See also Close, Push Behind, and Print.

Yes

This test operator tests boolean data (statements that are either true or false). They are usually used to test the logical states of subgoal hypotheses. Used in the conditions of the Rule and Method editor. See also No.

C *Network Icons*

This appendix summarizes the iconic buttons for the Rule, Object, and Cross-Reference Network windows that you select from the Browsers menu. Each window displays an icon palette that you can use to navigate the networks. Although several of the iconic buttons appear the same, the way you use them differs. For this reason the descriptions that follow, address the Rule, Object, and Cross-Reference Network icons separately.

Note: The system duplicates most iconic button functions through popup menus you display on items in the network diagram.

The Rule Network Icons

Descriptions of the Rule Network icons follow. The functions these icons provide act only on the Rule Network window. Procedures for using the icons to aid in investigating your application appear in Chapters Three and Five of this guide.

Note: If you change the default orientation of the rule network diagram, the system reverses the function of the Left Arrow and Right Arrow icons. See Appendix D, "Customizing the Environment" to modify the display orientation.

Left Arrow

This icon invokes the backward chaining investigation mode and changes the mouse cursor to resemble the left arrow. The system extends the existing rule network diagram in the direction of the arrow to show any rules that point to the hypothesis you select with the left arrow cursor. The hypothesis can be a terminal hypothesis, a condition as hypothesis, or an action using a hypothesis.

The Extend Left popup menu option provides the same function. To exit this mode select another icon or popup menu option.

Right Arrow

This icon invokes the forward chaining investigation mode and changes the mouse cursor to resemble the right arrow. The system extends the existing rule network diagram in the direction of the arrow to show the hypotheses of any rules whose conditions or actions use a datum or hypothesis you

select with the right arrow cursor. The system can also show links for rules whose hypotheses share contexts. The rule network diagram represents context links as dashed lines.

The Extend Right popup menu option provides the same function. To exit this mode select another icon or popup menu option.

Network Eraser

This icon invokes the erase mode and changes the cursor to resemble an eraser. You can simplify the existing rule network diagram by clearing unwanted extensions from the display. The system clears the network extensions from the hypothesis or condition you select. The function has no effect if the selected item was not the originator of an extension.

The Erase popup menu option provides the same function. To exit this mode, select another icon or popup menu option.

Network Focusing

This icon invokes the focus network mode and changes the cursor to resemble a magnifying glass. You can start a fresh investigation from the hypothesis, rule, or data (in a condition or an action) you select. The system displays the knowledge structure separately from the existing rule network diagram.

The Focus popup menu option provides the same function. To exit this mode, select another icon or popup menu option.

Documentation

This icon invokes the display file mode and changes the cursor to resemble a file. You can display information from files that were previously associated with the slot you select. The knowledge base must be configured to use the Apropos meta-slot before the system can display a file. If information is available, it can add another dimension to the investigation of inferencing pathways.

The Apropos popup menu option provides the same function. To exit this mode, select another icon or popup menu option.

Navigation Breakpoints

This icon invokes the insert/remove breakpoint mode and changes the cursor to resemble a stop sign. Breakpoints you place in the rule network diagram cause the inference engine to pause during knowledge processing as follows.

■   On a condition or action, *after* the system evaluates it.

■ On a rule name/number, *after* the system fires the rule.

■ On a hypothesis, *after* the system evaluates it.

Thus breakpoints let you analyze the current status of the network diagram and make changes to the knowledge base interactively. To remove breakpoints from the rule network diagram, position the stop sign cursor over the ones you want to remove and click.

To exit the insert/remove breakpoint mode, select another icon or popup menu option.

Extension Mode

This icon toggles the system between two full display extension modes. The default mode (represented by single left arrow icon) prevents the Display All global popup menu option from showing duplicate rules extending from identical hypotheses. The alternate mode (represented by two left arrows icon) allows the Display All global popup menu option to show duplicate rules wherever they appear. The default mode ensures that the full network diagram you display starts off as clean as possible.

The extension mode selection also affects the appearance of extensions revealed with the Full Left Extent popup menu option. Full right extensions *never* duplicate pathway branches regardless of the current extension mode.

The Object Network Icons

Descriptions of the Object Network icons follow. The functions these icons provide act only on the Object Network window. Procedures for using the icons to aid in navigating your application appear in Chapters Three and Five of this guide.

Note: f you change the default orientation of the object network diagram, the system reverses the function of the Left Arrow and Right Arrow icons. See Appendix D, "Customizing the Environment" to modify the display orientation.

Left Arrow

This icon invokes the generalization investigation mode and changes the mouse cursor to resemble the right arrow. You can reveal the inheritance pathways "upward" along the knowledge base parent/child relationships. The system extends the existing object network diagram to show knowledge structures as follows.

■ From a class displays the superclasses.

- From an object displays its classes and superobjects.
- From a property attached to an object or class displays its source, where it is used in the parents of the object.
- From an independent property displays all the classes and all the objects with this property attached, in two separate branches.

The Extend Right popup menu option provides the same function. To exit this mode, select another icon or popup menu option.

Right Arrow

This icon invokes the specialization investigation mode and changes the mouse cursor to resemble the left arrow. You can reveal the inheritance pathways "downward" along the knowledge base parent/child relationships. The system extends the existing object network diagram to show knowledge structures as follows.

- From a class displays the class properties, its subclasses, and its object instances.
- From an object displays its properties and its subobjects.
- From a property attached to an object or class displays its meta-slots.
- From an independent property has no effect.

The Extend Left popup menu option provides the same function. To exit this mode, select another icon or popup menu option.

Network Eraser

This icon invokes the erase mode and changes the cursor to resemble an eraser. You can simplify the object network diagram by clearing unwanted network extensions from the display. The system clears the network extensions from the class, object, or property you select. The function has no effect if the selected item was not the originator of an extension.

The Erase popup menu option provides the same function. To exit this mode, select another icon or popup menu option.

Network Focusing

This icon invokes the focus network diagram mode and changes the cursor to resemble a magnifying glass. You can start a fresh investigation from the object, class, or property you select. The system displays the knowledge structure separately from the existing object network diagram.

The Focus popup menu option provides the same function. To exit this mode, select another icon or popup menu option.

Documentation

This icon invokes the display file mode and changes the cursor to resemble a file. You can display information from files that were previously associated with the slot you select. The knowledge base must be configured to use the Apropos function before the system can display a file. If information is available, it can add another dimension to the investigation of inheritance pathways.

The Apropos popup menu option provides the same function. To exit this mode, select another icon or popup menu option.

Navigation Breakpoints

This icon invokes the insert/remove breakpoint mode and changes the cursor to resemble a stop sign. Breakpoints that you place in the object network diagram cause the inference engine to pause during knowledge processing as follows:

- On a class or object, *after* the value of any of its slots changes.
- On a property, *after* the value of any of the slots changes, for all slots related to the property.
- On a slot, *after* the value of the slot changes.
- On a method, *after* the system completes the method.

Thus breakpoints let you analyze the current status of the network diagram and make changes to the knowledge base interactively. To remove breakpoints from the object network diagram, position the stop sign cursor over the ones you want to remove and click.

To exit the insert/remove breakpoint mode, select another icon or popup menu option.

The Cross-Reference Network Icons

Descriptions of the Cross-Reference Network icons follow. The functions these icons provide act only on the Cross-Reference Network window. Procedures for using the icons to aid in investigating your application appear in Chapter Three of this guide.

Left Arrow / Right Arrow

These icons invoke the cross-reference extensions mode. The system extends the item to be referenced in the direction of the arrow to show the occurrences of that item in the knowledge base. Cross-referenced items can include hypotheses, rules, methods, context links, validation functions,

inheritance atoms, and inference atoms. If you change the default orientation of the cross-reference network diagram, the system reverses the function of the Left Arrow and Right Arrow icons. See Appendix D, "Customizing the Environment" to modify the display orientation.

The Extend Right and Extend Left popup menu options provide the same functions. To exit this mode select another icon or popup menu option.

Network Eraser

This icon invokes the erase mode and changes the cursor to resemble an eraser. You can simplify the existing cross-reference network by clearing unwanted extensions from the display. The system clears the network extensions from the item you select. The function has no effect if the selected item was not the originator of an extension.

The Erase popup menu option provides the same function. To exit this mode, select another icon or popup menu option.

Network Focusing

This icon invokes the focus network mode and changes the cursor to resemble a magnifying glass. You can isolate an existing item in the network in order to view its cross-references. The system displays the new extensions separately from the existing cross-reference network diagram. This mode is typically used in combination with the Options window that you select from the Window-specific menu in order to change the cross-reference choices.

The Focus popup menu option provides the same function. To exit this mode, select another icon or popup menu option.

D *Customizing the Environment*

This appendix summarizes customizing operations you can perform in order to customize the Intelligent Rules Element windowing environment. Some of these operations were previously described in the context of the preceding chapters. They are repeated here along for the sake of completeness. Refer to the chapters for more details about their usage.

Set Up Environment Options

You can customize the overall environment by selecting the Set Up Environment option on the File menu or from the pop-up menu in the session control panel. Figure D-1 shows the Set Up Environment dialog window.
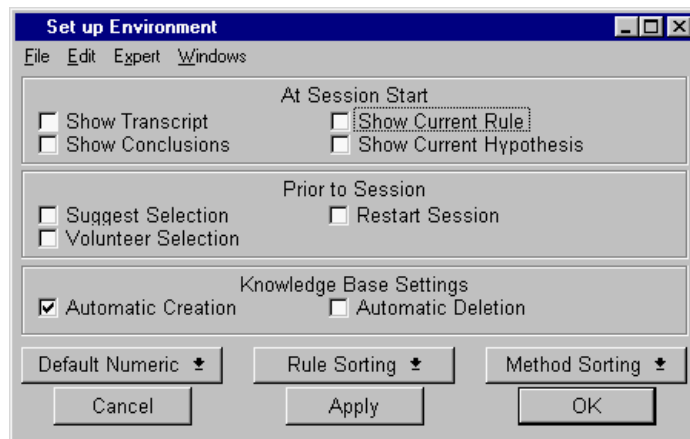
Figure D-1 Set Up Environment Dialog Window

Note: To save the settings between sessions, use the Save Environment option of the File menu. The next time you start the Rules Element, the system reloads all settings. The position, size, and colors of specific windows are also saved.

At Session Startup

The top part of this window controls the opening and closing of the monitoring windows. For each of these windows, you can specify if the window should appear automatically when you initiate the knowledge processing session, and if it should disappear when you close the session control window. Refer to Chapter Five, "Application Testing" for details about these options.

Prior to Session

The check buttons in this section of the window allow you to automate the Restart, Suggest, and Volunteer commands for knowledge testing. If you select the Restart Session button, the system restarts the session automatically before executing the Knowcess command. Furthermore, if you select the Suggest Selection and/or the Volunteer Selection the system will suggest and/or volunteer the lists that you have selected in the Suggest/Volunteer dialog window (with the Keep button) or in the Hypotheses and Data Notebooks. The system performs this operation between the automatic Restart Session (if any) and the start up of the inference engine.

Editor Settings

The next block of settings controls the behavior of the editors. With the default setting (Automatic Creation selected), the Rules Element automatically creates objects, classes, and properties when you edit rules or meta-slots and ensures unique names are assigned. You are prompted only when the Rules Element cannot determine by itself the nature (object or class) or the data type (boolean, integer, float, date, etc.) of an atom. This setting is very convenient when you start your design. When your design is more advanced, your object and class base becomes more stable and you want to monitor more precisely the creation of objects, classes and properties (In the autocreate mode the Rules Element will create new knowledge structures if you make a typographical error during your editing). At this stage, you can turn off the automatic creation so that the Rules Element informs you when the system creates new knowledge structures.

The Automatic Deletion setting controls whether or not the Rules Element will attempt a cleanup operation when you delete a rule or meta-slot. If the Automatic Deletion is enabled, the Rules Element will delete all the objects, classes, slots, and properties that were referenced in the rule or the meta-slot and which are no longer referenced (in other rules, meta-slots, contexts, etc.) after the deletion of the rule or meta-slots. Otherwise, you must delete these objects manually.

Default Numeric Data Type

The Default Numeric Data Type setting allows you to choose the default data type for numeric operations. The default setting is to use the floating point data type. This setting keeps the interaction very simple (you aren't continuously prompted to choose the data type of properties or slots). If you want to use the other data types (integer, date, time), you must select the Prompt User setting. Then the Rules Element will prompt you more often but you will be able to correctly edit expressions using these new data types.

Rule and Method Sorting

These settings let you specify how to display rules and methods when using the alphabetic search mechanism of the editor and list windows. In the case of rules, the search can be by hypothesis name or by rule name. In the case of methods, the search can be by method name or by the object names to which the methods are attached.

Window Color Selection

The Save Environment command of the File menu lets you reuse the color settings you make for individual windows between sessions. The remaining sections of this appendix give information about setting colors for browser type windows. In general color selections are made as follows:

Editor windows	Use the popup menu attached to the window
Trace windows	Use the popup menu attached to the window.
	Note: A color setting made to an open multiple instance window, such as Case Status and Full Report windows, will apply to all subsequently opened windows of the same type.
Browser windows	Use the Change Settings dialog to change the background color of the browsers (rule, object, and cross-reference). To change the Resource Browser window, select the Preferences option from its popup menu.

Note: The background color of windows is affected by windowing system defaults in the case of the PC (Control Panel) and workstations (Xdefaults).

Network Window Settings

Figure D-2 shows the Network Settings window, obtained by selecting the Change Settings option on the global pop-up menu, followed by selecting the Browser... button in the small dialog window. The settings are the same for all Browser windows that you use to display networks. This includes the Rule Browser, Object Browser, Cross-References Browser and Resource Browser windows.
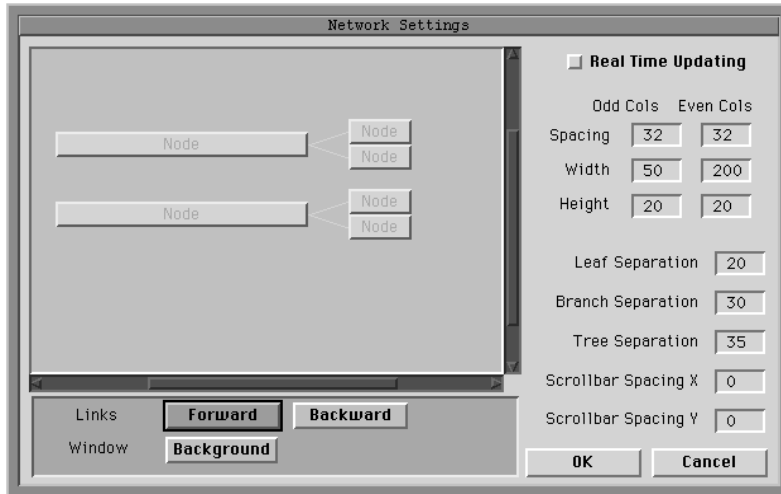
Figure D-2 Network Settings Dialog Window

Note: **T**he Network Settings are part of the global settings saved in a database with the Save Environment option on the File menu.

The right area of the dialog window allows you to customize the length of different items in the network diagram. The left area shows two simple rules with one originating node on the left and two extension nodes on the right (an example of a Left to Right orientation). Figure D-2 shows the default string's lengths as follows.

- 200 pixels for strings including hypothesis names, LHS conditions, and RHS actions
- 32 pixels for the single line between the hypothesis name and rule name
- 50 pixels for the rule name (or rule number)
- 32 pixels for all lines between the rule name and the LHS conditions and RHS actions.

The lengths can be modified simply by clicking over the number to be replaced and typing in a new number (maximum is 1000). Adjustments to these settings affect the current network's display and either make more or less information visible within the fixed-size window. Generally, 200 pixels are sufficient for displaying the hypotheses' names and the first two strings of a LHS or RHS. It is possible, however, that 50 pixels will not be enough to fully display a rule name.

Note: :Some conditions or actions will always be cut because their arguments make them too long. The ViewLine option on the window's global popup menu is provided to display such long strings in a separate window.

The Network Settings dialog lets you customize several settings that control the appearance of the network diagram. Table D-A identifies the customizable settings of the Browser windows.

Setting	Usage
Real Time Updating	When you select this option, the system scrolls the contents of the Browser at the same time that you perform the scrolling operation. The default is unselected since this option may slow the scrolling of complex diagrams.
Spacing, Width, and Height	You can specify the pixel values that control the size of the nodes displayed in the network diagram. Any value can be specified, although the text may appear cropped when made too small.
Branch Separation	This option lets you specify a pixel value to separate the extension nodes. If your network displays nodes, you can reduce its size by minimizing the leaf separation distance.
Tree Separation	This option lets you specify a pixel value to separate the originating nodes when you display more than one network at a time in the Browser.
Scrollbar Spacing X, Y	This option lets you change the separation distance between scrollbars and the scrollable area itself. (This option is not available on the PC and Macintosh computers.)
Background Color	This button lets you customize the color choices for the Browser background.
Forward and Backward Color	These buttons let you customize the color choices for the Browser nodes and their links.

Table D-1 **.** Network Diagram Settings

Use the following procedure to modify the appearance of your network diagram when you need to conserve space.

To customize the network diagram appearance:

1. Display the Network window-specific menu from the main menu bar and select the **Change Settings** command from the list. The system displays the Network Settings dialog for that window.

2. Select the **Browser...** button from the dialog window. The system displays the Networks Settings dialog.

3. Edit the text edit fields of the settings as desired and click on the **Apply** button. The sample diagram reflects the changes. If necessary repeat the procedure to achieve the desired effect.

4. When you are satisfied with the current settings click on the **OK** button of both dialog windows. The system closes the Settings dialogs and applies the new settings to the actual network diagram.

Rule Network Display Options

Figure D-3 shows the Rule Network Settings dialog, obtained by selecting the Change Settings option on the global pop-up menu. The other network window settings are almost identical, and appear later in this appendix.
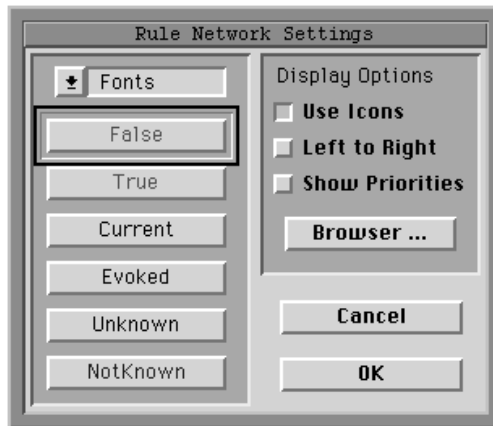
Figure D-3 Rule Network Settings Dialog

Note: The Network Settings are part of the global settings saved in a database with the Save Environment option on the File menu.

Use Icons

The Use Icons checkbox lets you display icons in the network diagram to indicate the state of a condition, an action, a rule, or a hypothesis, as shown in Figure D-4.
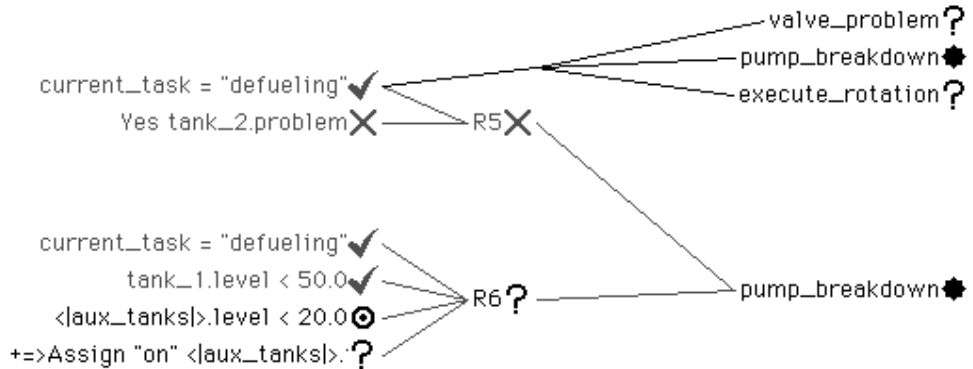
Figure D-4 Rule Network Diagram Icons

The icons that appear in the network diagram include the following.

Question Mark	The condition, rule, or hypothesis is unknown and has not yet been evaluated. The action has not been processed.
Bold Checkmark	The condition or hypothesis is TRUE. The rule was evaluated as TRUE. The action was processed.
Reverse Checkmark	The condition or hypothesis is FALSE. The rule was evaluated as FALSE.
Rectangle	The condition is NOTKNOWN or undetermined (user uncertainty).
Target	The condition is currently under evaluation.
Star	The hypothesis is under investigation (current focus).

Left to Right

The Left to Right checkbox lets you modify the orientation of the display. In the Rule Network window, it corresponds to the *deductive* direction and its default value is right to left.

Show Priorities

The Show Priorities checkbox lets you display the inference priorities of the various knowledge structures, thereby showing the fine architecture of the knowledge base. Refer to Chapter Two, "Application Implementation" for a description of priorities. Unlike the Object Network, default categories (value 1) appear as well. In the RHS or LHS, only the *priority numbers* of the data or slots are shown. For a rule, the priority number is between angle brackets, but if there is a *priority slot*, its value is used instead and written followed by a star (the knowledge structure itself is visible in the ViewLine):

[3*]	the value of knowledge structure's priority is 3
[N*]	the value of knowledge structure's priority is NotKnown
[U*]	the value of knowledge structure's priority is UnKnown

The conventions are the same for a hypothesis, except that square brackets mean that it is a terminal hypothesis and parentheses mean that it is used elsewhere as a subgoal. Figure D-5 shows a rule with the Show Priorities option selected.
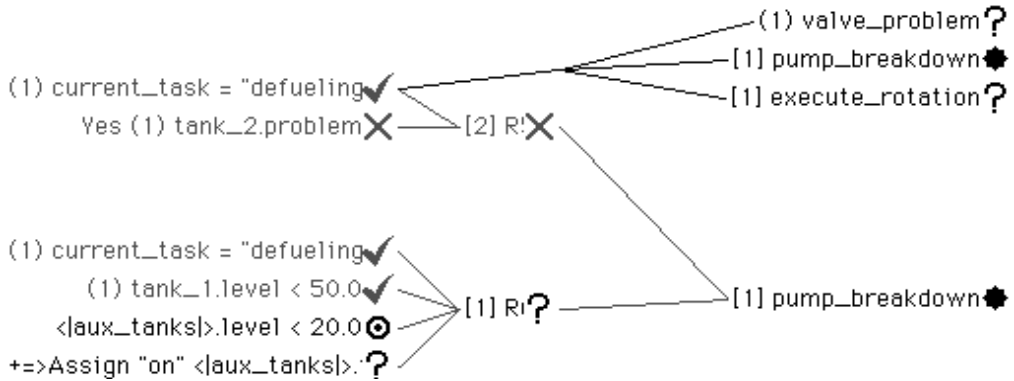
Figure D-5 Rule Network Diagram Priorities

Color and Fonts

The six buttons at the bottom of the Rule Network Settings dialog window let you choose a specific typeface and color representation for the various states of the network diagram's knowledge structures at any moment

during a session. Each state can be customized with a different font style (their icon equivalents are described above).

False	Button for the conditions and hypotheses known as FALSE.
True	Button for the conditions and hypotheses known as TRUE.
Current	Button for the representation of the precise condition under evaluation.
Evoked	Button for the hypotheses currently under investigation (focus of attention).
Unknown	Button for the unknown facts or hypotheses (for which no value has been sought).
Notknown	Button for the conditions and hypotheses known as being Notknown.

Click in any of the buttons to obtain a dialog window that lets you select a font type, a font size, a style, and a color.

Object Network Display Options

Figure D-6 shows the Object Network Settings dialog window, obtained by selecting the Change Settings option on the global pop-up menu.

Note: The Network Settings are part of the global settings saved in a database with the Save Environment option on the File menu.
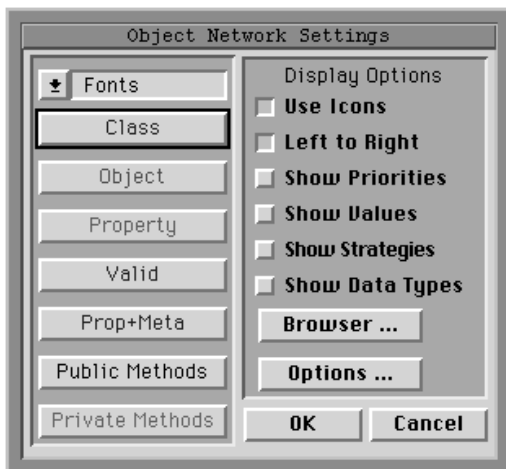
Figure D-6 Object Network Settings Dialog

Use Icons

The Use Icons checkbox lets you display small icons that indicate knowledge structure type. The icons the system uses to represent these structures include the following.

Circle for class.

Triangle for object or dynamic object (with plus).
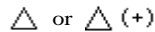
Square for property - with meta-slots if filled.
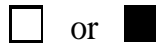
Diamond for method.

Upsidedown Triangle for data validation function.

Left to Right

The Left to Right checkbox lets you modify the orientation of the display. In the object network diagram, it corresponds to the *generalization* or *composition* direction.

Show Priorities

The Show Priorities checkbox lets you display the inheritance and inference priorities for individual slots. The priorities appear in front of the slot in the following ways.

[i,j]   Where i is the Inheritance category and j the Inference category,

Important: Nothing is displayed if both i and j are equal to the default value 1.

[i*,j]   The star indicates that a priority slot exists and its value is displayed in place of the priority number. The slot name can be seen in the ViewLine window.

[N*,j]   The priority slot's value is `NotKnown`.

[U*,j]   The priority slot's value is `UnKnown`.

Show Values

The Show Values checkbox lets you display the values of each slot. Figure D-7 shows the object network with values displayed.
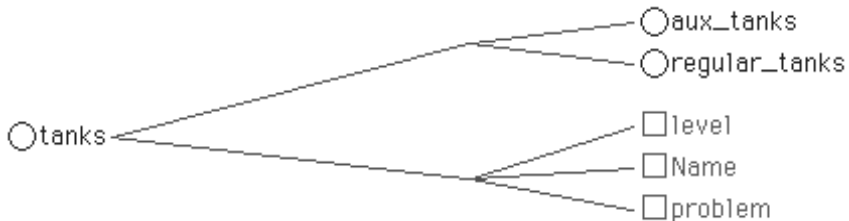
Figure D-7Object Network Diagram Values

Show Strategies

The Show Strategies checkbox lets you display the strategies attached to a slot and entered in the Meta-Slot editor window. They are visible only when they are different from the default strategy. Code letters are added after the slot name, between slashes. The possible combinations are:

VU,VD,VUD = **V**alue **U**p & **D**own

PU,PD,PUD = **P**roperty **U**p & **D**own

OD,OB,CD,CB = **O**bject first, **C**lass first, **D**epth first, **B**readth first.

Figure D-8 shows Prop1 has inheritance strategies defined: the slot's Value should be inherited Up and Down, and the path strategy is Object First - Depth First.
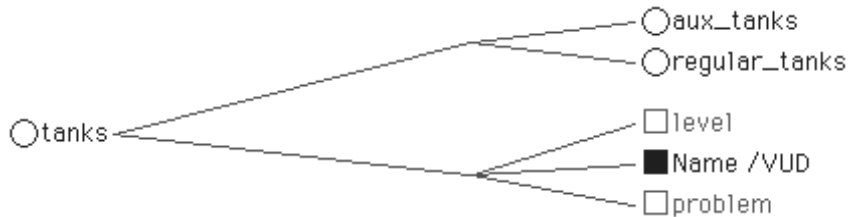
Figure D-8Object Network Diagram Strategies

Show Data Types

The Show Data Types checkbox lets you display slots or properties' data types with one letter in parentheses: (B) boolean, (I) integer, (F) float, (D) date, (T) time, or (S) string. The type of the special property "Value" is object dependent and can be seen with this option. For example, a hypothesis will have its slot displayed as (B) Value since it is a boolean.

Color and Fonts

The six buttons at the bottom of the Settings dialog let you choose a specific typeface and color representation for each type of knowledge structure. The types can be customized with a different font style and are also easily recognized by their icons.

Class	Button for the structures of type *Class* (circle icon).
Object	Button for the structures of type *Object* (triangle icon).
Property	Button for the structures of type *Property* or *Slot*, a slot representing a property attached to a particular object or a class (square icon).
Prop+Meta	Button for the structures of type *Slot* which have meta-slots different from the default settings, i.e.

either strategies, categories, data validation, etc. (filled square icon).

Public Method   Button for the methods of type *Public* methods (diamond icon).

Private Methods   Button for the methods of type *Private* methods (diamond icon with "P" inside).

Options...

The Options... button displays a small dialog window (see Figure D-9) that lets you simplify the network diagram by choosing whether or not to display classes, objects, properties, validation functions, and methods when expanding the object network.

Figure D-9 Object Network Display Options

Cross-Reference Network Display Options

Figure D-10 shows the Cross Reference Settings dialog window, obtained by selecting the Change Settings option on the global pop-up menu.

Note: : The Network Settings are part of the global settings saved in a database with the Save Environment option on the File menu.

Figure D-10 Cross-Reference Settings Dialog

Note: Currently only the Use Icons and Left to Right display options are valid. The Show... options have no effect on the cross reference network diagram.

Use Icons

The Use Icons checkbox lets you display small icons that indicate knowledge structure type. The icons the system uses to represent these structures include the following.

Circle for class.

Triangle for object or dynamic object (with plus).

Square for property - with meta-slots if filled.
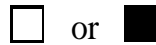
Diamond for method.

Upsidedown Triangle for data validation function.

Figure D-11 shows several of these icons used in a cross reference network diagram.

Property: level □ Referenced in Rule R6 (1 references)▽
Referenced in Rule R7 (1 references)▽
Referenced in Rule R6 (1 references)▽
Used in Validation function for regular_tank:■
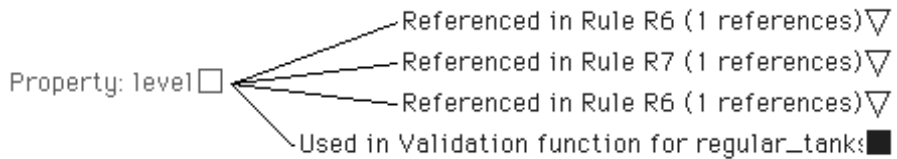
Figure D-11Cross-Reference Network with Icons

Left to Right

The Left to Right checkbox lets you modify the orientation of the display. In the cross reference network diagram, the meaning is the same in both directions.

Color and Fonts

The six buttons at the bottom of the Settings dialog let you choose a specific typeface and color representation for each type of knowledge structure. The

types can be customized with a different font style and are also easily recognized by their icons.

Class	Button for the structures of type *Class* (circle icon).
Object	Button for the structures of type *Object* (triangle icon).
Property	Button for the structures of type *Property* or *Slot*, a slot representing a property attached to a particular object or a class (square icon).
Prop+Meta	Button for the structures of type *Slot* which have meta-slots different from the default settings, i.e. either strategies, categories, data validation, ect. (filled square icon).
Public Method	Button for the methods of type *Public* methods (diamond icon).
Private Methods	Button for the methods of type *Private* methods (diamond icon with "P" inside).

Options...

The Options... button displays a small dialog window (see Figure D-12) that lets you simplify the network diagram by choosing whether or not to display

hypotheses, rules, inheritance atoms, inference atoms, context links, methods, and validation functions when using the Cross Reference window.
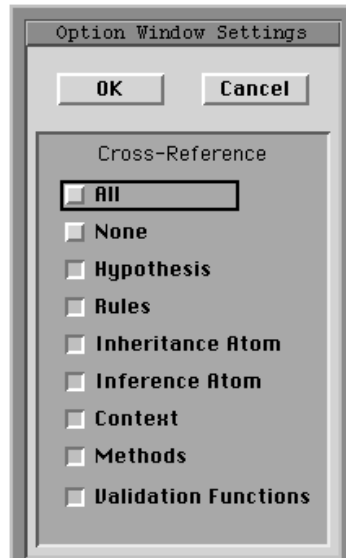
Figure D-12 Cross Reference Display Options

Page Setup Options

Users of Unix or VAX platforms are able to select several options to customize the appearance of knowledge base printouts. The customization choices, provided by the Page Setup dialog window, must be made before initiating the Print command from a window. Display the dialog window

by selecting the Page Setup option on the System menu. Figure D-13 shows the window with its default settings.
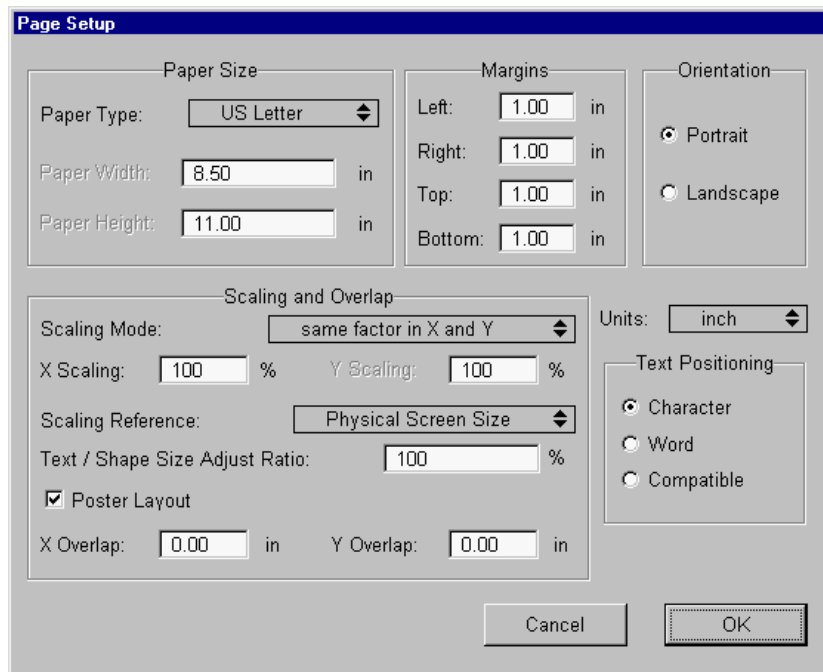
Figure D-13 Page Setup Dialog Window for Unix and VAX

Once the options are selected from this dialog window, they apply to all following printouts. If you do not display the dialog window, the system uses the default choices as shown in Figure D-13. The following sections describe the customization choices you can make to print on Unix and VAX systems.

Note: **M**acintosh users can display a similar dialog window by selecting the standard Page Setup option on the File menu. The options are the usual Macintosh dialog box options including paper size, image scale, orientation, and certain effects for printing.

The following five options of the Unix and VAX Page Setup dialog window (see Figure D-13) let you control the appearance of the printed page.

Paper Size

This option lets you match the printout page size with the type of paper available for printing. Many standard choices are already provided that are

compatible with a wide range of printers. You can also type over the paper size values for customized dimensions. The first value adjusts the paper width, and the second value adjusts the paper length. Dimension units can be represented in either inches, millimeters, or points.

Margins

The page margins are adjustable and print at one inch all the way around the page unless changed. The actual units depend on the current Unit selection and can be inches, millimeters, or points. Margins you supply need not be symmetrical although this probably helps when printing the network diagrams.

Orientation

The paper orientation options change the way the printer uses the paper to print the image. If the default Portrait checkbox is selected, the image appears printed in the usual vertical orientation. If the Landscape orientation is selected instead, the image appears printed lengthwise across the page.

Note: This option can help to optimize the use of paper when printing the network diagram.

Scaling

This option lets you reduce or enlarge all printouts that you can make from the windowing environment. The default 100% figure produces a printed image close to the size of the image shown on the screen. You can type a percentage less than 100 to reduce the printed image size, or you can type a percentage greater than 100 to increase the printed image size. The X value controls the horizontal dimension, while the Y value controls the vertical dimension.

Note: This option can help to print full lines of text that would otherwise appear cut off. Reduce the scale by an appropriate percentage. This option can also help to optimize the use of paper when printing the network diagram.

Overlap

This option lets you cause multi-page printouts (for example, a large network diagram printout) to overlap by the amount that you specify. The default 0 inches produces printed images that have no overlap. You can change the default value to help identify which page edges are the adjoining ones when the printout is completed. The X value controls the horizontal edge, while the Y value controls the vertical edge.

E *Text KB Syntax Description*

This appendix describes the format in which knowledge bases are saved (text format or non-compiled format) to disk. This format is also the format which should be used if you want to compile a portion of text with the `NXP_Compile` call of the the Intelligent Rules Element Application Programming Interface (API).

Knowledge Base Description

A text knowledge base consists of a version identification followed by one or several knowledge base units.

knowledge_base:

```
version_identification
knowledge_base_unit1
knowledge_base_unit2
knowledge_base_unit3
...
```

Knowledge base units can be of the following types:

knowledge base unit:

```
        comments
or      property_description
or      class_description
or      object_description
or      slot_description
or      rule_description
or      method_description
or      globals_description
```

Version Identification

The version_identification has to be the following string:

```
(@VERSION= 0XX)
```

Comments

The knowledge base comments unit has the following syntax:

```
(@COMMENTS= quoted_string)
```

If the knowledge base contains several comments units, only the last one will be loaded. Comments can be used to hold history or source code control (SCCS strings) information.

> **Note:** Lines beginning with two slashes (//) are ignored by the Rules Element parser. You should use this commenting facility with care because these comments are not loaded in memory and will be lost the next time you save your knowledge base from the Rules Element.

Conditions, RHS, and Methods

In the following description, we will designate by xhs the text representation of a condition, a RHS action, an Order of Sources, or an If Change action. The syntax of an xhs is one the following:

xhs:

```
        (operator(      (argument) )
or      (operator(      (argument) (argument) )
```

The operator is one of the operator keywords appropriate for the xhs.

The argument is the text of the argument as displayed in the Rule or Method editors. The precise syntax of arguments is described in the previous sections.

Globals Description

The globals description contains global information attached to the knowledge base: inference and inheritance strategies, and lists of selected hypotheses or data. Its syntax is the following:

```
(@GLOBALS= globals_item_list)
```

The globals_item_list is a list of global items. The syntax of a global_item is the following:

global_item:

```
        @PWTRUE=        true_or_false;
or      @PWFALSE=       true_or_false;
or      @PWNOTKNOWN=    true_or_false;
or      @PFACTIONS=     true_or_false;
or      @PFEACTIONS=    true_or_false;
or      @PFMACTIONS=    true_or_false;
or      @PFMEACTIONS=   true_or_false;
or      @EXHBWRD=       true_or_false;
or      @SOURCESON=     true_or_false_or_continue;
or      @CACTIONSON=    true_or_false_or_continue;
or      @PTGATES=       true_or_false;
or      @INHOBJUP=      true_or_false;
```

or	@INHOBJDOWN=	true_or_false;
or	@INHCLASSUP=	true_or_false;
or	@INHCLASSDOWN=	true_or_false;
or	@INHVALUP=	true_or_false;
or	@INHVALDOWN=	true_or_false;
or	@INHBREADTH=	true_or_false;
or	@INHPARENT=	true_or_false;
or	@VALIDUSER=	true_or_false;
or	@VALIDENGINE=	true_or_false;
or	@SUGLIST=	slot_list;
or	@VOLLIST=	slot_list;

The `list_of_conditions_xhs` and the `list_of_rhs_actions_xhs` are lists of `xhs` separated by spaces, tabs, or new lines.

Knowledge Structure Descriptions

Property Description

A property_description has the following syntax:

`(@PROPERTY=identifier @TYPE=type; format )`

The identifier is the name of the property. The type is one of the following keywords:

> Boolean
> Integer
> Float
> String
> Date

The format is optional and describes the output and input formats for the property. Its syntax is the following:

`@FORMAT= quoted_string ;`

The quoted string contains the format description. Format syntax is described in the Reference Manual.

Class Description

The syntax for a class description is the following:

`(@CLASS=        identifier          class_item_list)`

The `class_item_list` is a list (eventually empty) of class items. The syntax of a `class_item` is the following:

class_item:

```
        (@SUBCLASSES    subclasses_list)
or      (@PUBLICPROPS  public_properties_list)
or      (@PRIVATEPROPS private_properties_list)
```

The `subclasses_list` is a list of the names of the subclasses (separated by spaces, tabs, or newlines).

The `properties_list` is a list of the names of the properties (separated by spaces, tabs, or newlines). If the property `Value` is present in the list, it must be followed by a description of its type:

```
Value           @TYPE=type;
```

Object Description

The syntax for an object description is the following:

```
(@OBJECT=       identifier           object_item_list)
```

The `object_item_list` is a list (eventually empty) of object items. The syntax of an `object_item` is the following:

object_item:

```
        (@CLASSES        classes_list)
        (@SUBOBJECTS    subobjects_list)
or      (@PUBLICPROPS  public_properties_list)
or      (@PRIVATEPROPS private_properties_list)
```

The `classes_list` is a list of the names of the classes (separated by spaces, tabs, or newlines).

The `subobjects_list` is a list of the names of the subobjects (separated by spaces, tabs, or newlines).

The syntax of the `properties_list` has been described in the previous section.

Slot Description

The slot description contains the information attached to the object or class slots: meta-slots and/or contexts. Its syntax is the following:

```
(@META=         slot       slot_item_list)
```

The `slot_item_list` is a list of slot items. The syntax of a slot_item is the following:

slot_item:

```
        @INFCAT=        integer;
or      @INHCAT=        integer;
or      @INFATOM=       slot;
or      @INHATOM=       slot;
or      @PROMPT=        quoted_string;
```

```
or      @COMMENTS=      quoted_string;
or      @FORMAT=        quoted_string;
or      @INHSLOTUP=     true_or_false;
or      @INHSLOTDOWN=   true_or_false;
or      @INHVALUP=      true_or_false;
or      @INHVALDOWN=    true_or_false;
or      @INHBREADTH=    true_or_false;
or      @INHPARENT=     true_or_false;
or      (@CONTEXTS=     slot_list);
or      (@INITVAL=      quoted_known_value);
```

The `slot_list` is a list of slots separated by spaces, tabs, or newlines. The `list_of_sources_xhs` and the `list_of_if_change_actions_xhs` are lists of `xhs` separated by spaces, tabs, or newlines.

Method Description

The syntax of a method description is the following:

(`@METHOD=`   identifier   method_item_list)

The method_item_list is a list of method items. The syntax of a method_item is the following:

method_item:

```
        @ATOMID=        xxx;
or      @TYPE=          type;
or      @FLAGS=         xxx;
or      @COMMENTS=      quoted_string;
or      @WHY=           quoted_string;
or      (@ARG1=_        name;@NATURE=slot;@TYPE=type;@DEFVAL="xxx";)
or      (@LHS=          (list_of_conditions_xhs) );
or      (@RHS=          (list_of_then_actions_xhs) );
or      (@EHS=          (list_of_else_actions_xhs) );
```

The `list_of_conditions_xhs`, `list_of_then_actions_xhs`, and the `list_of_else_actions_xhs` are lists of `xhs` separated by spaces, tabs, or newlines.

Rule Description

The syntax of a rule description is the following:

(`@RULE=`   identifier   rule_item_list)

The rule_item_list is a list of rule items. The syntax of a rule_item is the following:

rule_item:

```
        @INFCAT=i       nteger;
or      @INFATOM=       slot;
or      @COMMENTS=      quoted_string;
or      @WHY=           quoted_string;
or      (@HYPO=         slot );
```

```
or        (@LHS=              (list_of_conditions_xhs) );
or        (@RHS=              (list_of_then_actions_xhs) );
or        (@EHS=              (list_of_else_actions_xhs) );
```

The `list_of_conditions_xhs`, `list_of_then_actions_xhs`, and the `list_of_else_actions_xhs` are lists of `xhs` separated by spaces, tabs, or newlines.;

Index

Symbols

A

B

C

D