Aion Development System

User's Guide

PLATINUM *technology, inc.*
555 Twin Dolphin Drive, Suite 400
Redwood City, CA 94065
(415) 591-8200

DC0427

Title and publication number

User's Guide
DC0427

Product version

This manual accompanies Release 7.0 of AionDS/Win.

Acknowledgements

- Writers: Qiron Adhikary and Alex Sato
- Assisted by: Michael Campbell

Table of Contents

Chapter 3 Work at the Knowledge Base Level

Chapter 4 Work with Object List

Chapter 5 Work with Objects

Chapter 6 Work with the Window Editor

Chapter 8 Track Changes in a Knowledge Base

Chapter 9 Debug Knowledge Bases

Chapter 10 Customize AionDS

Appendix Build and Run Character-Based Applications in AionDS

Index

How to Use This Manual

What this manual is about

This manual describes how to use the AionDS graphical development environment (GUI AionDS).

Who should read this manual

You should read this manual if you are not familiar with GUI AionDS. This manual assumes you are already familiar with knowledge base objects and AionDS development concepts.

How to find information

This manual is organized as follows:

Chapter	Contents
Chapter 1	introduces AionDS concepts
Chapter 2	provides a short tour of the functions of AionDS
Chapter 3	describes actions you can perform on an entire knowledge base
Chapter 4	explains how to use an object list window and its various formats
Chapter 5	explains how to edit knowledge base objects and their properties
Chapter 6	explains how to build a knowledge base with a graphical user interface
Chapter 7	explains how to access databases external to AionDS
Chapter 8	explains how to track changes made to knowledge bases
Chapter 9	explains how to debug errors in knowledge bases
Chapter 10	explains how to customize windows and other AionDS environment characteristics
Appendix A	explains how to build character-based user interfaces in GUI AionDS

For more information

To build knowledge bases, you should have the following AionDS manuals:

General Reference

Language Reference

I/O Reference

Messages and Codes

User's Guide (Character-Based)

Installation and Operation Guide for your platform

Application Programming Interface Guide for your platform

If you are building a knowledge base with a graphical user interface, you should also have the following AionDS manuals:

Building GUI Applications: Reference

Building GUI Applications: Tutorial

Aion® product abbreviations

The AionDS documentation set uses the following abbreviations for Aion® products:

Aion® product	Abbreviation
Aion® Development System	AionDS
Aion® Execution System	AionES
Component Aion® Execution System	CAES
Multitasking Aion® Execution System	MAES
Aion® Development System Application Programming Interface	ADSAPI
Aion® High Performance Option	HPO
Aion® Cooperative Processing Option	CPO
Callable Aion® Building System	CABS
Aion® Code Integration Option	CIO

Conventions

The AionDS documentation set uses the following conventions.

Typeface

The documentation set uses these typeface conventions:

Style of type	What it represents
Boldface	new term and its definition
Italic	■ manual titles ■ syntax variables ■ emphasis ■ foreign language phrases
`Monospace`	■ code examples ■ prompts and messages ■ examples of user input (in procedures, for example)

Icons

The documentation set uses these icons:

Symbol	Meaning
☯ *Alternative:*	identifies an alternative to the previous procedure (choice is usually a matter of personal preference; neither offers a distinct advantage)
✋ *Caution:*	helps you avoid mistakes that can produce unexpected or undesirable results (runtime errors and losing work, for example)
☒ *Exception:*	indicates an exception to common functionality or behavior
🕮 *For more information:*	identifies additional sources of information on a particular topic

continued

continued

Symbol	Meaning
Important:	stresses a point that can help you achieve greater success with the AionDS product
Reminder:	reminds you in passing of an important aspect that was discussed previously
☑ *Tip:*	suggests a technique that can result in more efficient use of the system
➢	identifies a one-step procedure

Syntax

The following conventions are used to define the AionDS syntax and command formats in this manual.

- You must type letters that are capitalized. You do not have to capitalize these letters. Lowercase letters are optional. If a command or syntax statement contains both capitalized and lowercase letters, you can type only the capitalized letters:

  ```
  RUn
  ```

  For the above command, you can type the following letters:

  ```
  ru
  run
  ```

  You cannot type the following letter:

  ```
  r
  ```

  Important: For AionDS running on UNIX platforms, AionDS commands are not case sensitive. However, file names that exist in UNIX (for example, knowledge base names) must be typed in uppercase and lowercase exactly as displayed in UNIX.

- Italics indicate a response that you must specify:

  ```
  RUn kb_name
  ```

  The expected response for an italicized word is indicated below the command or syntax statement.

- Brackets indicate an option:

  ```
  ASORT (expr [ON key])
  ```

  The `ON key` phrase is optional.

- A bar indicates a choice:

  ```
  List-objects [State st_name | Vocabulary vocab_nm]
  ```

  In this case, you can type either the `State` choice or the `Vocabulary` choice.

- Braces indicate a required choice:

  ```
  List-objects {State st_name | Vocabulary vocab_nm}
  ```

  You must type a `State` choice or a `Vocabulary` choice.

- You must use other characters, such as parentheses and angle brackets, exactly as specified:

  ```
  EOF (fparam)
  ```

  The parentheses are required.

- An ellipsis (...) indicates that you can repeat a choice:

  ```
  param IS FROM (stexpr, stexpr, [, stexpr ...])
  ```

  You can repeat "`, stexpr`" as many times as you want.

Chapter 1
Introduction to AionDS Concepts

Introduction

This chapter describes the basic Aion Development System (AionDS) concepts that you need to understand to develop graphical knowledge bases.

You can skip this chapter if you are familiar with AionDS on another platform or have attended an AionDS training class.

In this chapter

AionDS concepts

Overview

AionDS is an application building tool and consists of the following:

- a structure for modeling knowledge
- an inference engine

You use AionDS to build knowledge bases. A **knowledge base** is structured information about a specific subject. For example, if your company has a few sales managers who are good at assigning optimal quotas for their sales agents, you can build a knowledge base that contains their expert knowledge. Instead of asking for a sales manager's expert advice, other sales managers can run the knowledge base to set optimal quotas for their sales agents.

Types of information

The information structure is organized into the following categories:

- **Application knowledge** includes facts, judgments, procedures, and interface-specific information.
- **Control knowledge** orders the steps that the knowledge base takes in order to reach the final result.

To build a knowledge base, you create a structure of AionDS knowledge structures that models specific application and control knowledge. **Knowledge structures** or **objects** are reusable modules of knowledge. They separate knowledge into discrete elements that AionDS can manipulate.

Objects

This table shows the objects that can represent certain types of knowledge.

Type of knowledge	Objects
high-level controls	states (agendas)
judgments or business logic	rules
facts or data	classes, slots, types, instances, hierarchy of classes, parameters
procedures	functions, methods
interface-specific information**	displays, messages, graphs,* groups,* processes, reports

**Use the WindowObject class library to build graphical user interfaces.

*These objects are only valid for character-based knowledge bases.

Libraries

A **library** is a collection of objects that you can include in a knowledge base. You can include any knowledge base as a library in another knowledge base. A library can include all types of objects. In a knowledge base you can create objects that are owned by library objects, such as a subclass of a library class.

Using knowledge bases as libraries is an easy way to reuse objects and quickly build other knowledge bases.

Restrictions

The following restrictions apply to libraries:

- The number of objects in a library must be equal to or less than 65,535.
- A knowledge base can include up to 255 libraries.
- Libraries are always external to the knowledge base.
- When you include a library in a knowledge base, you cannot change the objects in the included library from within the knowledge base.
- When you include a library in a knowledge base, the library's entry state and the entry state's objects are not included.

Inference engine

The other part of AionDS is the inference engine. The **inference engine** is an integrated collection of problem-solving algorithms. These problem-solving algorithms process the knowledge base according to its application and control knowledge to compute application results. The inference engine can be applied to many different kinds of applications and cannot be modified. The inference engine does not contain any application-specific knowledge and is separate from the knowledge base.

For example, the inference engine determines the execution order for if-then rules based on known information. If your knowledge base has the following rule:

```
if residence = 'CA'
then taxrate = '7.5'
```

Before processing this rule, the inference engine must find another rule that determines the value of `residence` or ask the user for the value of `residence` if no such rule exists.

📖 *For more information:* See Appendix A, "Porting Knowledge Bases," of the *General Reference* for a detailed description of the inference engine.

Knowledge base user interfaces

In AionDS, knowledge bases can have either a graphical or character-based user interface. This manual assumes that you are building and running knowledge bases with graphical user interfaces. Throughout the rest of this manual, the term knowledge base is assumed to mean a knowledge base with a graphical user interface.

AionDS/2

In AionDS/2 Version 6.2 or newer, you can build and run knowledge bases with graphical or character-based user interfaces. In AionDS/2 Version 6.11 and older, you cannot build knowledge bases with graphical user interfaces nor can you run knowledge bases that have graphical user interfaces.

AionDS/Win Version 6.2

You can build and run knowledge bases with graphical and character-based user interfaces.

AionDS/Win Version 6.4 and 7.0

These versions of AionDS/Win contain a Graphical User Interface (GUI) development and execution environment (GUI AionDS) and a character-based development and execution environment. In GUI AionDS, you can:

- build and run knowledge bases with graphical user interfaces
- run knowledge bases with character-based user interfaces (AionDS/2 only)
- build all parts of the knowledge base except for its character-based user interface

In character-based AionDS, you can:

- build and run knowledge bases with character-based user interfaces
- run knowledge bases with graphical user interfaces
- build all parts of the knowledge base except for its graphical user interface

📖 *For more information:* See the *User's Guide (Character-based)* to learn more about building knowledge bases with character-based user interfaces in a character-based development environment. See Appendix A, "Building and Running Character-based Applications in AionDS," for details about building knowledge bases with character-based user interfaces in a GUI development environment.

Knowledge base structure

Knowledge base components

A GUI knowledge base consists of the following components:

- knowledge base object hierarchy (application and control knowledge)
- WINLIB library, which includes the WindowObject class hierarchy and the DDEObject class hierarchy
- system library
- windows, comprising:

  application window
  other windows and dialogs
  desktop window

Knowledge base object hierarchy

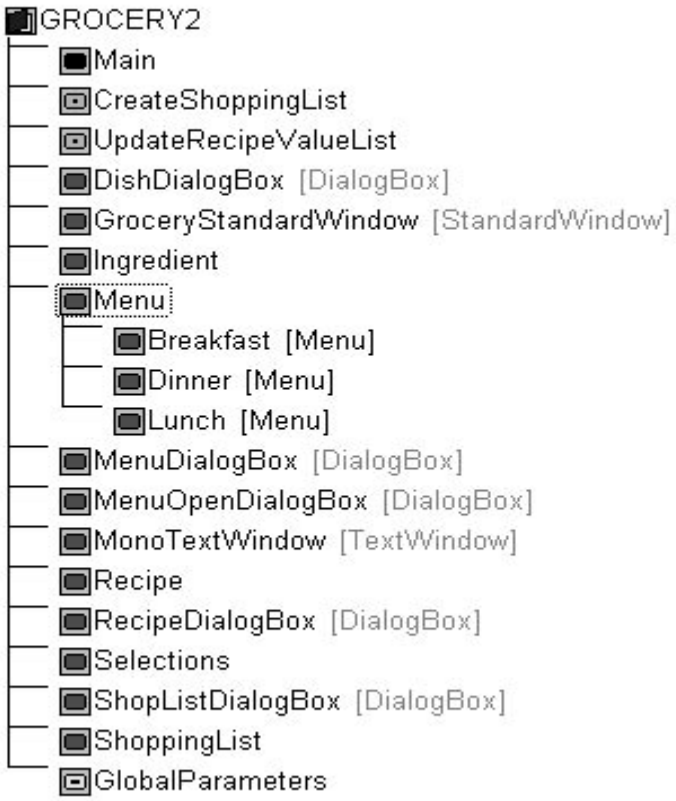
Overview

To integrate knowledge base objects that contain application and control logic into a working knowledge base system, organize the knowledge base objects in hierarchical structures. Knowledge base objects contained in the hierarchies are the following:

States*	Methods
Classes*	Parameters
Vocabularies*	Processes
Displays	Reports
Functions	Rules
Instances	Slots
Messages	Types

*independent context

Example Object hierarchies of the GROCERY2 knowledge base are shown as follows:



States, classes, and vocabularies are contexts. As **contexts**, they are the only objects that can be **independent**—that is, start at the top of a tree and not exist under another object. For example, the Menu class is an independent class, the Main state is an independent state, and GlobalParameters is an independent vocabulary. You can also create dependent states (substates) and classes (subclasses). A **dependent** state or class is a state or class that exists under another class or state in the tree. For example, the Breakfast class is a subclass of the Menu class.

Contexts own other objects. **States** integrate mostly business logic and high-level control logic.

Classes integrate mostly facts grouped as data values and the corresponding procedures that act on them.

Vocabularies contain global variables and constants.

States

States primarily contain rules that model business logic and the Agenda property that models high-level (control) instructions. **Properties** are attributes that define an object. When the knowledge base is started, the knowledge base's entry state is executed. The **entry state** is the first part of the knowledge base to be executed. A knowledge base can have only one entry state. The default entry state is the Main state.

States can own the following objects:

Displays	Reports
Messages	Rules
Parameters	Types
Processes	

Classes

Classes and the class hierarchy model the relationships between data and the procedures to manipulate that data. Classes contain primarily **slots** which are data definitions, **methods** which define procedures, and **instances** which contain the actual data values. Classes can also access external databases.

Classes can own the following objects:

Displays	Parameters
Functions	Processes
Instances	Reports
Messages	Slots
Methods	

Vocabularies

Vocabularies can own the following objects:

Displays	Processes
Functions	Reports
Messages	Types
Parameters	

For more information

Topic	Refer to
Instructions on displaying various object relationships in lists	Chapter 4
Instructions on building, editing, listing, exporting, importing, and printing objects	Chapter 5
A description of each object	*General Reference*

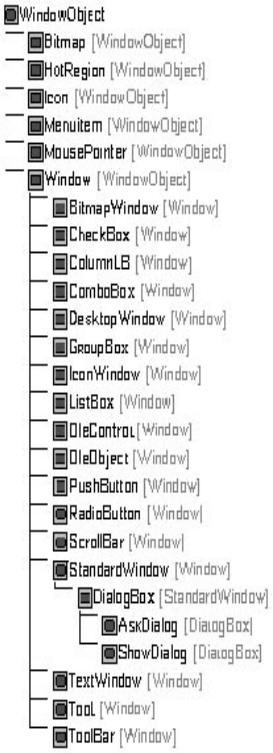
WindowObject class hierarchy

Overview

The WindowObject class hierarchy consists of predefined classes that you use to build a graphical user interface for your knowledge base.

WindowObject class hierarchy

The WindowObject class hierarchy includes the following classes:



The WindowObject class hierarchy contains predefined classes for the different elements of a graphical user interface, such as top-level windows, menus, push buttons, and text windows. Each window is built as an instance of one of these predefined classes from the WindowObject class hierarchy. The basic characteristics of each window are determined by the class to which it belongs. You must subclass the standard window and the dialog classes of the WINLIB before creating these instances. The additional characteristics that make each instance unique are set in the instance definition when it is created.

For example, the PushButton class contains all graphical characteristics that make up a push button. It is rectangular, shaded in gray, its title is displayed in the center of the rectangle, and so on. The exact size, position, and title of a push button are specified in its instance definition.

📖 *For more information:* See the *Building GUI Applications: Reference* for a description of each object in the WindowObject class hierarchy.

DDE Object class hierarchy

Overview

You can use DDE to pass data between two applications running on the same PC. The two applications that are passing data using DDE are having a **conversation**.

To build a DDE system, you need to use the classes in the DDE object class hierarchy.

📖 *For more information:* See the *Building GUI Applications: Reference* to learn more about a knowledge base as a DDE client and server.

Procedure for building knowledge bases

Overview

To build a knowledge base, you use the various object editors and the Window editor. An **Object editor** is a tool that you use to create and modify objects. For example, to create a class, you use the Class editor. You use object editors to build a hierarchy of objects that contains the knowledge base's application knowledge—facts, judgments, procedures, and high-level controls.

The **Window editor** is a tool you use to create and modify GUI windows. In conjunction with the object editors, you use the Window editor to build the GUI windows and their underlying logic. You can also build windows that use DDE using the Window editor and object editors.

What to do

The process of building a knowledge base is divided into the following tasks. They are arranged in the most logical order. To build a knowledge base, complete these tasks:

- create the knowledge base
- include libraries
- build the object hierarchies
- build the application window
- build other windows and dialogs

You must create the knowledge base first. You should create the application window before you build other windows and dialogs. Otherwise, you can concurrently build the object hierarchies, application window, and other windows and dialogs. For example, as you build the application window, you might have to create a parameter in the object hierarchy to hold the login name from the login field in a dialog.

Other AionDS features and utilities

List of features and utilities

AionDS has other features that expand its functions and utilities. These features are used in the development process. Brief descriptions of these features and utilities and where you can find more information about them are listed as follows:

- Set up the knowledge base as a DDE client, DDE server, or both.

  📖 *For more information:* See Chapter 6, "Working with the Window Editor," of the *Building GUI Applications: Reference*, and the *Building GUI Applications:Tutorial* to learn about DDE.

- Customize the DesktopWindow instance to specialize a knowledge base's error processing.

  📖 *For more information:* See the *Building GUI Applications: Reference* to learn about customizing the DesktopWindow instance.

- Access information in external databases from your application.

  📖 *For more information:* See Chapter 7, "Working with External Databases," and the *I/O Reference* to learn how to set up your application to access external databases.

- Track changes in an application using Change Management.

  📖 *For more information:* See Chapter 8, "Tracking Changes in a Knowledge Base," to learn how to implement Change Management effectively.

- Debug your application when you execute it.

  📖 *For more information:* See Chapter 9, "Debugging Knowledge Bases," to learn about finding and fixing errors in your application using the debugger.

- Customize your development and execution environments by modifying your profile.

  📖 *For more information:* See Chapter 10, "Customizing AionDS," to learn about customizing AionDS windows and profiles.

- Build and run character-based applications in GUI AionDS.

  📖 *For more information:* See Appendix A, "Building and Running Character-based Applications in AionDS."

Chapter 2
Touring AionDS

Introduction

This chapter provides an overview of AionDS by taking you on a tour of a sample knowledge base. If you are a new user, this chapter helps you get a broad overview of AionDS. You need not read this chapter to use the information in the rest of this book.

This chapter does not provide detailed explanations of each feature, but it does reference chapters containing more information on each topic.

In this chapter

Knowledge base example: GROCERY

Overview

Working with the simple knowledge base GROCERY lets you concentrate on AionDS features and how to use them. You need not understand fully the design of GROCERY to complete the tour of AionDS successfully.

After a default installation of AionDS, version 0.01 of GROCERY is available in the AionDS main directory under the `\EXAMPLES\DOC\GROCERY` subdirectory.

The AionDS convention for version numbers starts with 0.01 and is incremented by 0.01 for each consecutive version number. For example, the version number following 0.01 would be 0.02.

What does GROCERY do?

Purpose of GROCERY

GROCERY creates a shopping list for a summer camp based on your meal selection. Here is some general information about the GROCERY knowledge base:

- The camp using the GROCERY knowledge base serves three meals a day, five days a week.
- The GROCERY knowledge base allows users to plan one week of meals based on predefined menu selections.
- For each meal, you select dishes from several lists. Depending on the meal, the lists available are main dish, side dish, beverage, dessert, appetizer, soup, vegetable, and starch. You can add new dishes to each list.
- Based on meal selections, the knowledge base:
  - determines the grocery items to purchase
  - determines quantities needed
  - converts these quantities into standard package sizes
- Cafeteria meals work in the following way:
  - breakfast, lunch, and dinner are served daily
  - one week of meals is planned at a time
  - breakfasts have a main dish, side dish, and beverage
  - lunches have a main dish, side dish, beverage, and dessert
  - dinners have a main dish, beverage, appetizer, soup, vegetable, starch, and dessert
- Recipes are consulted to determine grocery needs. Recipes are based on a single serving.
- A database file contains a list of all ingredients you can purchase at a local grocery store.

What is implemented so far?

The following objects and logic are defined:

- The Main state agenda starts up the Grocery knowledge base and displays the application window. The GrocerySW instance is the application window.
- The CreateShoppingList state is defined.
- lasses are defined for Menu, ShoppingList, Recipe, Ingredient, and Selections. The Menu subclasses Breakfast, Lunch, and Dinner are defined.
- The GlobalParameters vocabulary is defined.
- To create menus:
  - select Menu.Open in the application window to open the Menu Open Selection dialog boxes
  - select Menu.Open All to open all the dialog boxes for each meal and day
  - select Menu.Add Dish to open the Add Dish dialog boxes
  - select ShoppingList.Open to open the Shopping List dialog boxes
- Various windows and dialog boxes have been created as follows:
  - the Grocery KB application window from which you can select the dishes for a specific meal and day, add dishes to a meal list, and display and print the shopping list
  - the Menu Open Selection dialog box from which you select the day and meal
  - a dialog for each day and meal from which you select the dishes
  - the Add Dishes dialog box from which you can add dishes to each meal list
  - the Shopping List dialog box that displays the shopping list

Starting AionDS

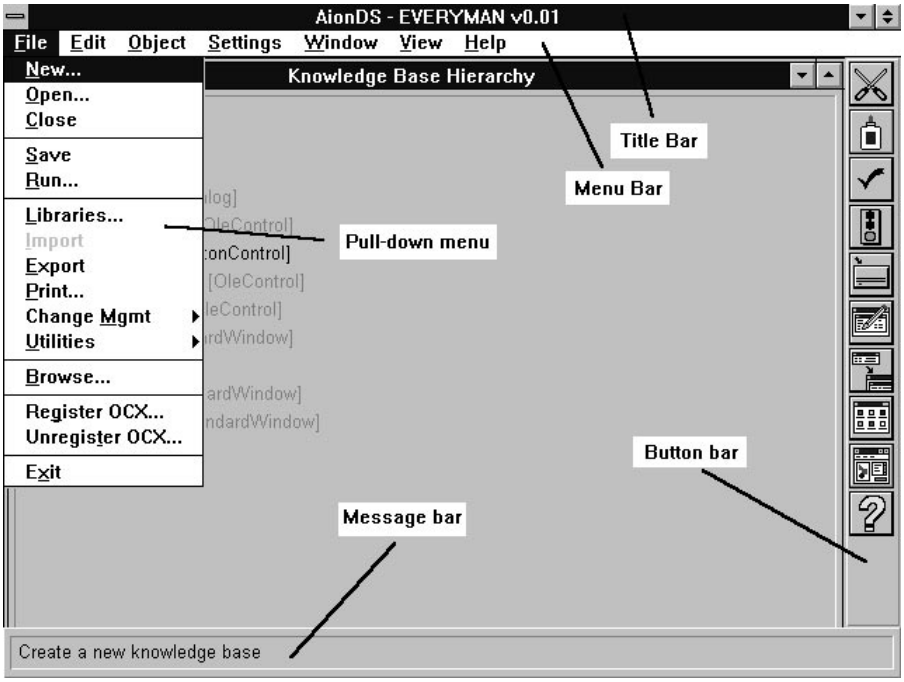
What to do

➢ To start AionDS, double-click the icon for GUI AionDS.

Result

When you first start AionDS, the primary window opens. The primary window, shown in the following figure, displays "AionDS" in its title bar. All knowledge base development takes place within a primary window.

Use the primary window to tell AionDS what to do.



☑ *Tip:* When a knowledge base is open, the words "AionDS" and the name and version of the current knowledge base display in the title bar.

Menu bar

Below the title bar is the menu bar, which provides you with a selection of actions you can take. Click any menu bar entry (or press ALT and the underlined letter in the menu bar) to display a pull-down menu.

Message bar

At the bottom of the primary window is the message bar. It provides a short phrase describing the action currently highlighted. The message bar also provides the text of error and informational messages that AionDS issues during knowledge base development.

📖 *For more information:* See Chapter 10, "Customizing AionDS," to learn how to turn the message bar on and off or place it in another part of the primary window.

Button bar

On the right side of the primary window, the button bar shows you a number of graphical buttons. These buttons are shortcuts to often-used AionDS actions. When you select a button, it initiates an action.

The button bar shortcuts are introduced throughout this book as their actions are described. From top to bottom, these buttons are:


cut


paste


save object


run knowledge base


save knowledge base


open an object


select properties to open


open an object list window


open the Window editor or Screen editor


task help

☑ *Tip:* The button that opens the Window editor in GUI applications opens the Screen editor in character-based knowledge bases. Use the Screen editor to build screens for a character-based knowledge base.

📖 *For more information:* See Chapter 10, "Customizing AionDS," to learn how to turn the button bar on and off or to place it in another part of the primary window.

Quitting AionDS

What to do

➢ To quit AionDS, select File.Exit.

☯ *Alternative:* You can also quit AionDS using one of the following procedures:

- select Close from the system menu
- double-click on the system menu

If you have not saved changes to knowledge base objects or AionDS settings, a confirmation dialog displays. Click Save to save the changes and terminate AionDS.

Click Discard to quit AionDS without saving changes.

Click Cancel to stay in AionDS without saving changes.

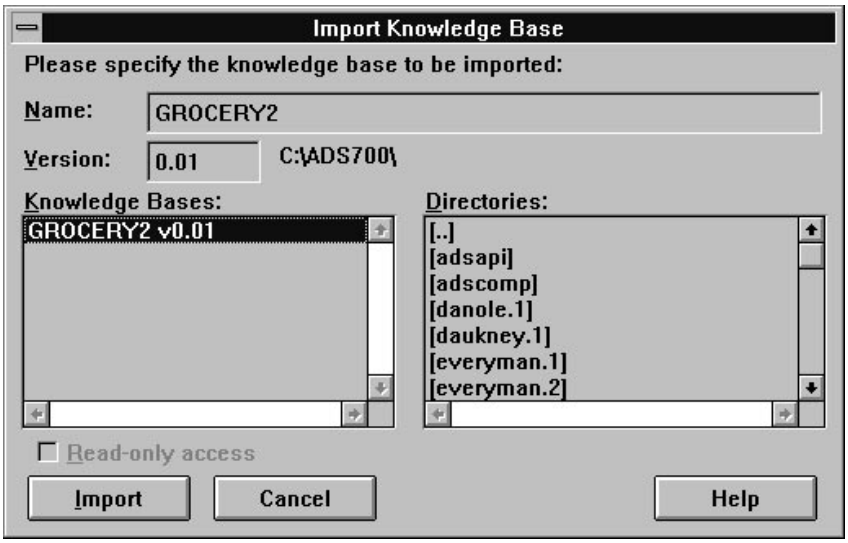
Importing GROCERY

Overview

This section explains how to import and run the sample knowledge base GROCERY. **Importing** a knowledge base opens it from the external import format.

What to do

1 Select File.Import. The Import Knowledge Base window displays.



☑ *Tip:* As you select a choice on the pull-down menu, note that the message bar at the bottom of the screen gives you a short summary of the choice. For help with a pull-down choice, press F1 when that choice is highlighted.

2 Use the Directories field to select the directory where the GROCERY knowledge base is placed—C:\AIONDS700\EXAMPLE\DOC by default. The knowledge bases available for import in this directory are displayed in the Knowledge Bases list box.

3 Double-click GROCERY v0.01 to import the knowledge base.

If you do not have a user name in a loaded Aion DS profile, AionDS prompts you for your name.

📖 *For more information:* See Chapter 10, "Customizing AionDS," to learn about profiles.

As AionDS is importing the knowledge base, the message bar shows you the progress and AionDS displays a message asking you to wait.

Result: After it is imported, the knowledge base title appears in the AionDS title bar. An object list window appears containing the class and state hierarchies. This knowledge base hierarchy window shows icons for the states, classes, and vocabularies in the GROCERY knowledge base.

☑ *Tip:* To open the Knowledge Base Hierarchy window after closing it:

1 Select Object.List. The Select Objects dialog displays.

2 In the Select Objects dialog, select Any in the Type field. Do not enter any text in the Name field.

3 Check External Objects in the Status group. This displays WINLIB and DDELIB.

4 Select KB in the Scope group.

5 Select Hierarchy in the List Format group.

6 Click Select.

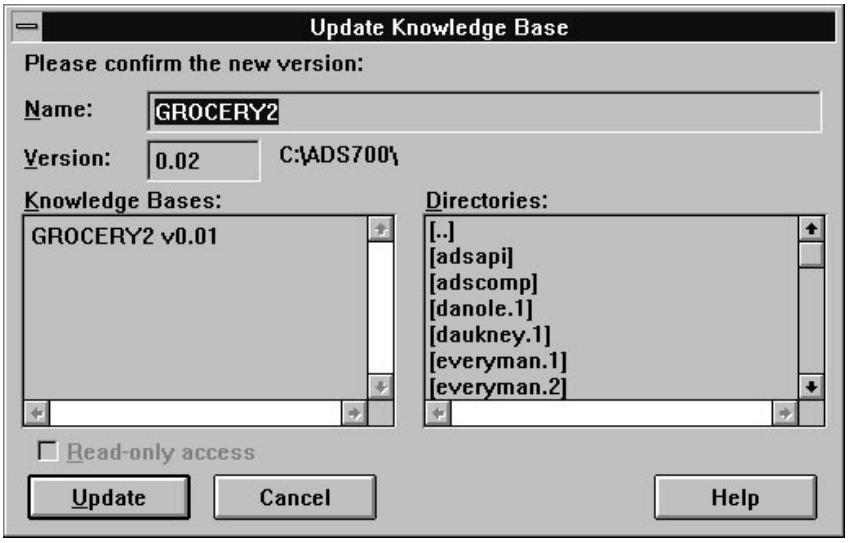
Result: The objects listed in the Knowledge Base Hierarchy window are displayed.

Updating GROCERY

Overview

Before exploring GROCERY, use the Update utility to make a copy of GROCERY. This way, you can make changes and still return to the original if your changes make the copy unusable.

1 Select File.Utilities.Update. The Update Knowledge Base window displays.



📖 *For more information:* See the "Updating a knowledge base" section in Chapter 3, "Working at the Knowledge Base Level," to learn more about using the fields in the Update Knowledge Base window.

2 Note that GROCERY displays in the Name field and 0.02 displays in the Version field. Click Update to make a copy of GROCERY with the incremented version number.

Result: After you select Update, note that the title bar now says AionDS GROCERY v0.02. AionDS has created an updated copy of the knowledge base, closed the original, and opened the updated copy.

Exploring the Knowledge Base Hierarchy window

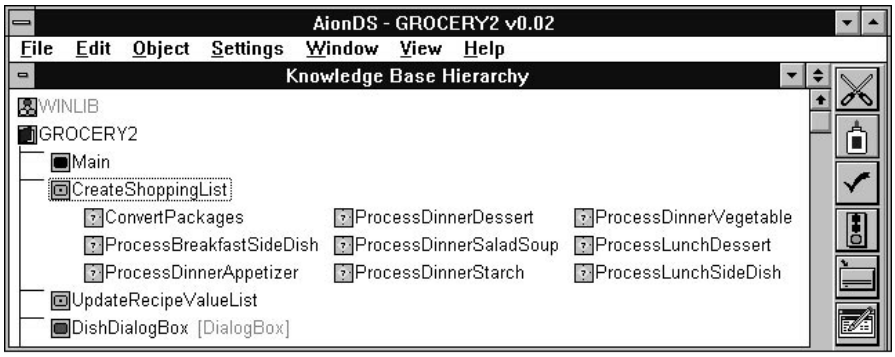
Overview

Before you run this knowledge base, use a few features of the Knowledge Base Hierarchy window described in the following sections. All features of this type of window are discussed in Chapter 4, "Working with Object Lists."

What objects does a state own?

What to do

- To see what objects the CreateShoppingList state owns, hold down CTRL and click the mini-icon for the CreateShoppingList state (not the text CreateShoppingList). The rules CreateShoppingList owns are listed.



What slots and methods does a class have?

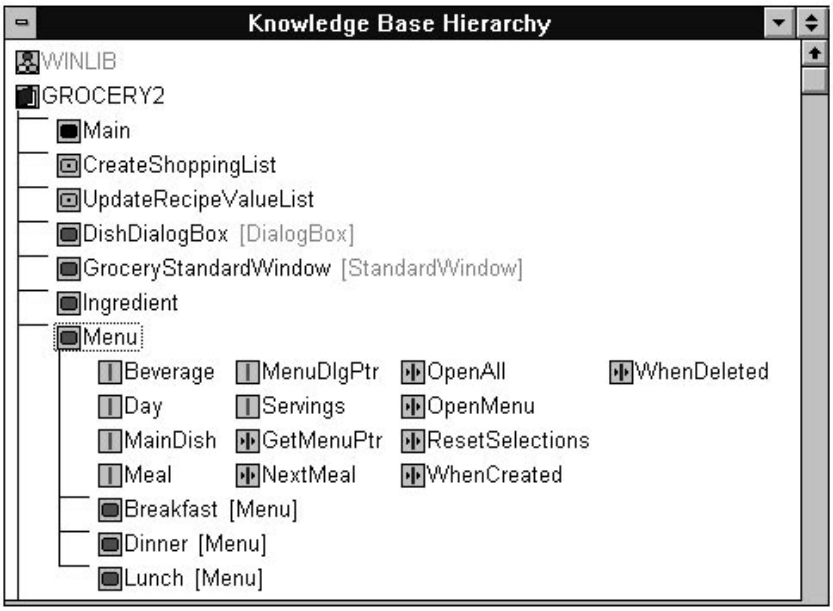
What to do

You just used the mouse to see the objects a state owns. Now, use the keyboard to look at a class hierarchy.

1. Press the arrow keys to move the focus (the dotted box) to the Menu class.
2. Press ALT-V to see the view pull-down menu. Now, use the up and down arrow keys to select a viewing option.
3. Select Slots/Methods, and press ENTER.

   ☯ *Alternative:* Press ALT and click the Menu class mini-icon.

   Result: The Menu class slots and methods display.



Looking at the CreateShoppingList state agenda

What to do

- Double-click the CreateShoppingList state to open an object editor for CreateShoppingList. The property shown by default is the agenda. You will see how to edit different properties later in this tour and in more detail in Chapter 5, “Working with Objects.”

Running GROCERY

Overview

One major advantage of AionDS is that you can run a knowledge base while your editors are still open. You can even explore the knowledge base and make changes to it while it is running (although you cannot save the changes until the knowledge base terminates). The following section shows you how to run a knowledge base.

Run window

What to do

1 Select Run in the button bar (the green traffic light). The Run window displays.

2 In this window, specify the characteristics for your knowledge base execution. For now, use the defaults. In Chapter 3, "Working at the Knowledge Base Level," you'll learn about the options available in this window.

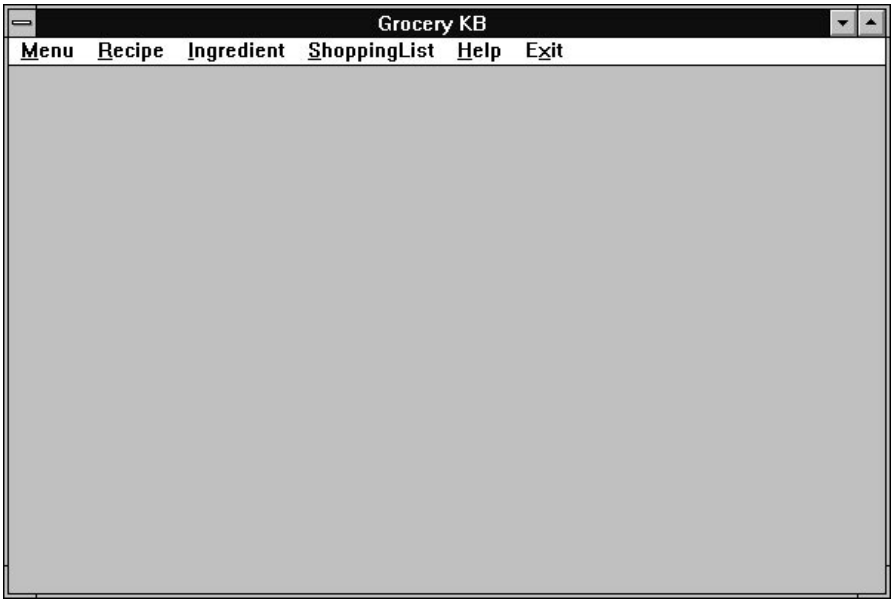
3 Click Run.

Application window

Overview

AionDS displays the application window and provides you with an execution system that mirrors what the user sees when the knowledge base is delivered.

📖 *For more information:* See Chapter 3, “Working at the Knowledge Base Level,” for notes on running a knowledge base on the AionDS desktop.



What to do

1. Select Menu.Open. The Menu Open Selection dialog displays.
2. From the Menu Open Selection dialog, select the day of the week and meal time (breakfast, lunch, or dinner) for which to select the dishes and click OK. The dialog box for that day and meal displays.
3. Select the number of servings (servings must be greater than or equal to 10) to serve for this meal. Select the dishes for this meal. To set up the menu for the next meal time, click Next and repeat this step. If you are done setting up the menu for the week, click OK.
4. The application window displays again. You can display a shopping list that shows the amount of each ingredient needed. The shopping list is based on the menu you created for the week. To display the shopping list, select ShoppingList.Open. The Shopping List window displays.

Changing the knowledge base

Overview

This section contains an exercise that gives you hands-on experience changing a knowledge base. The camp director asks you to modify the GROCERY knowledge base so you can add new recipes to the camp menu.

You and the director decide on the following concepts:

- Add the New menu item to the Recipe pull-down menu. The New menu item displays the Recipe dialog.
- Use the Recipe dialog to add new recipes to the camp menu and specify the required ingredients.

Knowledge base design

Since you are the knowledge base developer, the director has left the exact design and implementation in your expert hands. After a day of designing, you decide upon a knowledge base structure that contains the following:

- the Recipe.New menu item, which opens the Recipe dialog

  Camp personnel use the Recipe dialog to enter the new recipes and select (from a static list) recipe ingredients. When an employee completes and exits the Recipe dialog, the rules in an independent state are executed.
- an independent state with rules to allow new recipes to be selected from the Add Dish dialog
- a database file, linked to the Recipe class, to store new recipes

What to do

To build the knowledge base structure and implement this function, complete the following tasks:

1. Add the New menu item to the Recipe pull-down menu and include the method and function that open the Recipe dialog.
2. Create the Recipe dialog.
3. Create an independent state with rules to allow new recipes to be selected from the Add Dish dialog.
4. Change the Recipe class to create and access a database file that contains a list of recipes.
5. Close all editors and save the changes.
6. Test the new knowledge base.

To find out how the knowledge base works after completing these changes, see the GROCERY2 knowledge base. The following sections give you step-by-step instructions for each task.

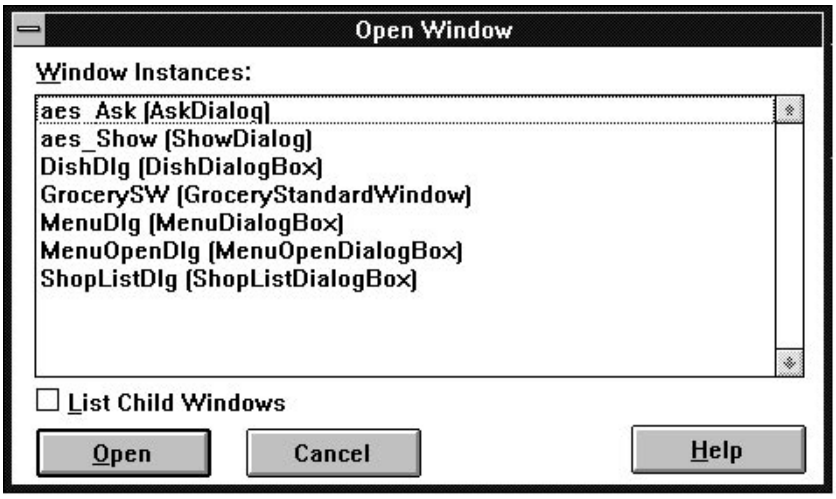
Adding the New menu item

Overview

In this step, you use the Window editor to add the New menu item to the Recipe pull-down menu in the GROCERY knowledge base main window, GrocerySW, and include the method and function that open the Recipe dialog.

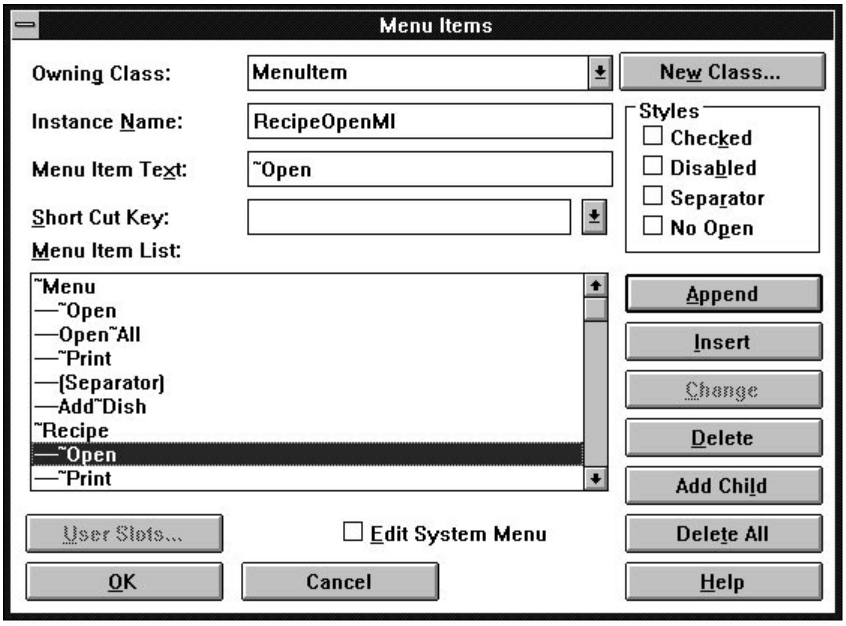
What to do

1. Click the Window editor window to activate it.
2. Open the GrocerySW window. Select Screen.Open Window. The Open Window dialog displays.



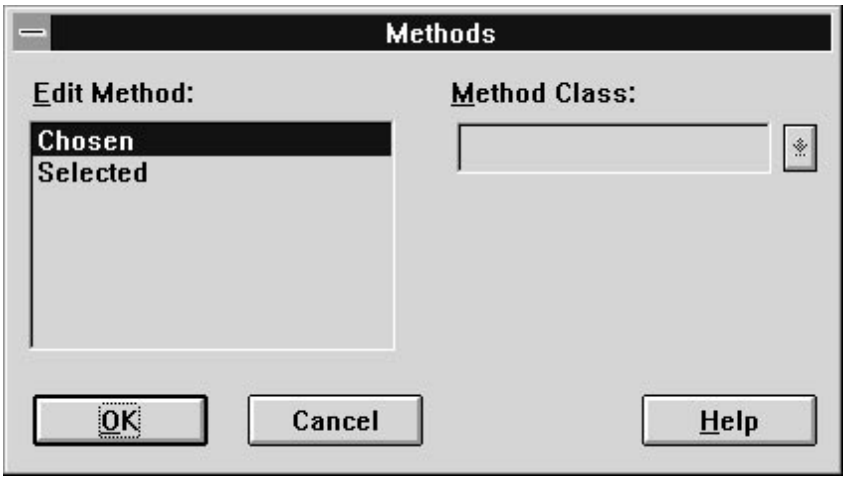
3 Double-click the Grocery KB window instance. The GrocerySW window instance displays.

4 Add the New menu item to the Recipe pull-down menu. Select Screen.Menu Items. The Menu Items dialog displays.

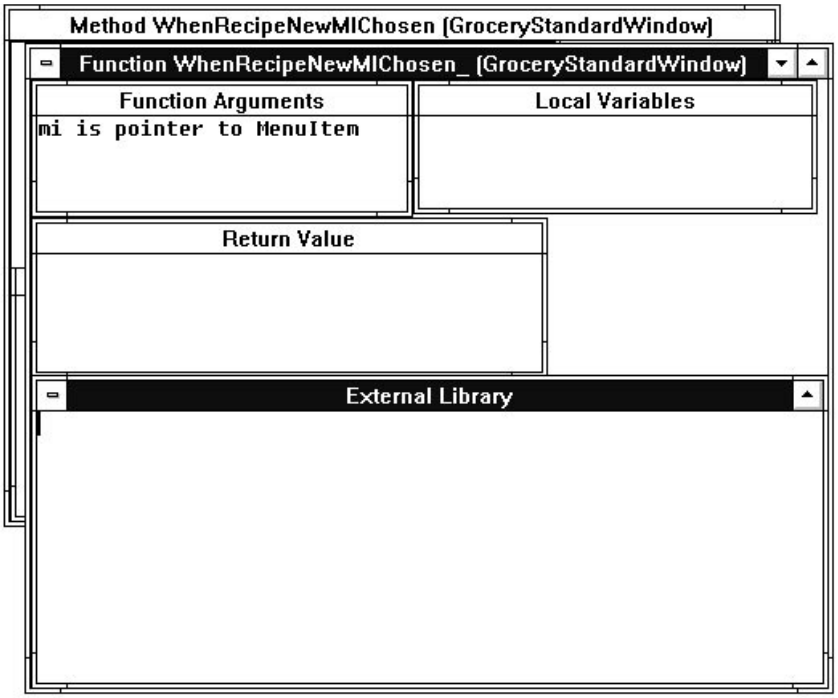


5 Select the Open menu item under ~Recipe in the Menu Item List field. Type over the Instance name, RecipeOpenMI, with the new name, `RecipeNewMI`. Type over the Menu Item Text name, ~Open, with the new name, ~New, and click Insert. Click OK and ~New is inserted before ~Open.

6 Create the WhenRecipeNewMIChosen method and function for the New menu item. This method is executed when the user selects the New menu item. In the GrocerySW window, select Recipe.New. The Methods dialog displays.



7 In the Edit Method group, select Chosen and click OK. The Method and Function dialogs are created and displayed.



8 In the Function Body property, include the following statements:

```
   /*  Goto the Open RecipeDlg or create a new     */
/*  Recipe instance and Open the RecipeDlg      */
/*  for it.                                     */

   if RecipeDlg.IsOpen
then
  RecipeDlg.SetActiveWindow
else
  RecipeDlg.RecipePtr = create (Recipe)
  RecipeNameTW.attach
   (slot(RecipeDlg.RecipePtr->.Name))
  RecipeIngredLB.attach
   (slot(RecipeDlg.RecipePtr->.IngredientsNeeded))
  RecipeDlg.Open
end
```

9 You cannot save the WhenRecipeNewMIChosen function and method until you create the Recipe dialog referenced in the Function Body property.

Creating the Recipe dialog

Overview

In this step, you use the Window editor to create the Recipe dialog as shown in the following figure.

In the Recipe dialog, you need to create the following objects:

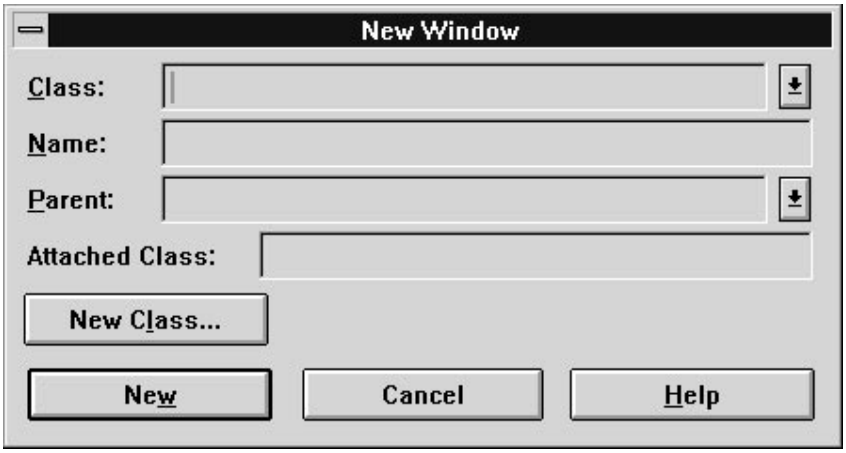
- label windows
- a text window
- a list box
- OK and Cancel push buttons
- RecipePtr slot to hold the new recipe values an employee will create in this dialog

You add a method and function for the OK push button to call the UpdateRecipeValueList state. You also add a method and function for the Cancel push button to delete the Recipe instance.

Creating the Recipe dialog

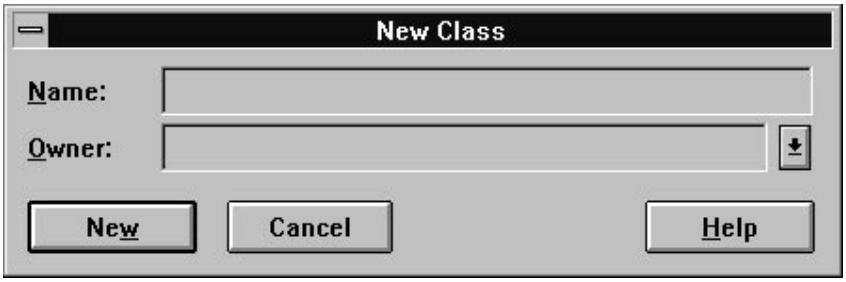
1 Activate the Window editor tool palette by clicking the Window editor window.

2 Select the New dialog box icon from the tool palette and click the cursor (which displays as the dialog box icon) in the screen. The New Window dialog displays.



3 Top-level windows should reside in subclasses so that the class hierarchy can be seen clearly and customized objects are separated from the WindowObject class library.

Create a subclass of DialogBox. Click New Class. The New Class dialog displays.

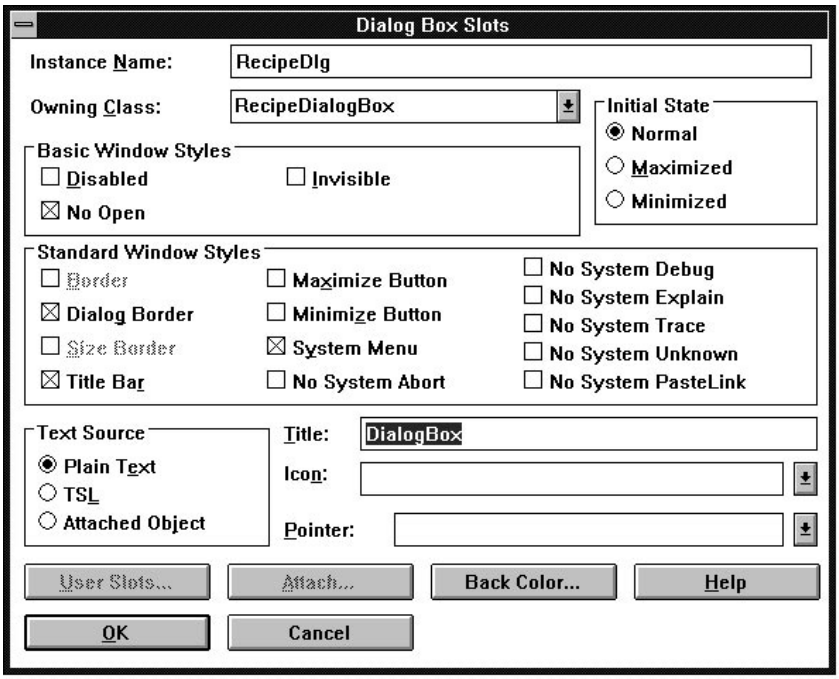


4 Enter `RecipeDialogBox` in the Name field. Enter `DialogBox` in the Owner field. Click New to create the RecipeDialogBox class and return to the New Window dialog.

5 In the New Window dialog, enter `RecipeDlg` in the Name field and click New.

6 Resize the Recipe dialog to the same size as the Recipe dialog figure on page 28. Place the tip of the arrow on the side of the dialog to resize so that the arrow turns into a double-headed arrow. Drag the mouse to pull or push the side in the desired direction.

7 Change the name displayed in the title bar of the RecipeDlg window. Double-click in the RecipeDlg window to display the RecipeDlg window slots.

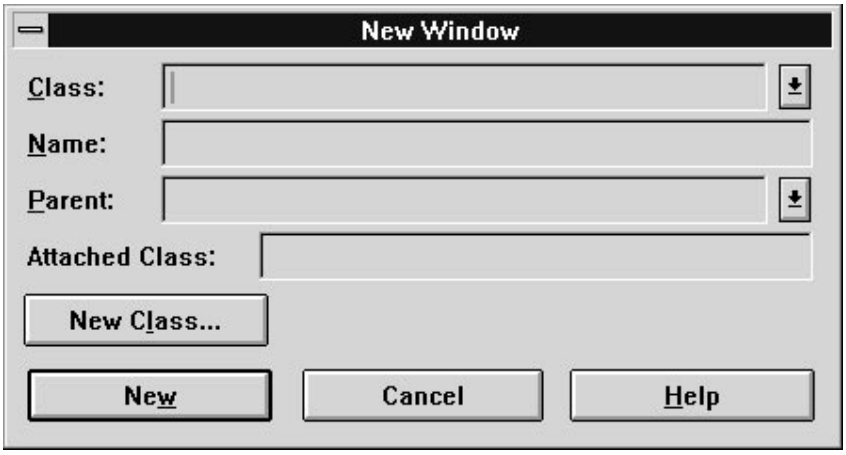


8 Replace the text in the Title field with Recipe. Click OK to return to the Recipe dialog.

Adding label windows

1 Add the Recipe Name and Ingredients Needed label windows. For each, select the Label Window icon from the tool palette and click the cursor (displayed as the Label Window icon) in the Recipe dialog box.

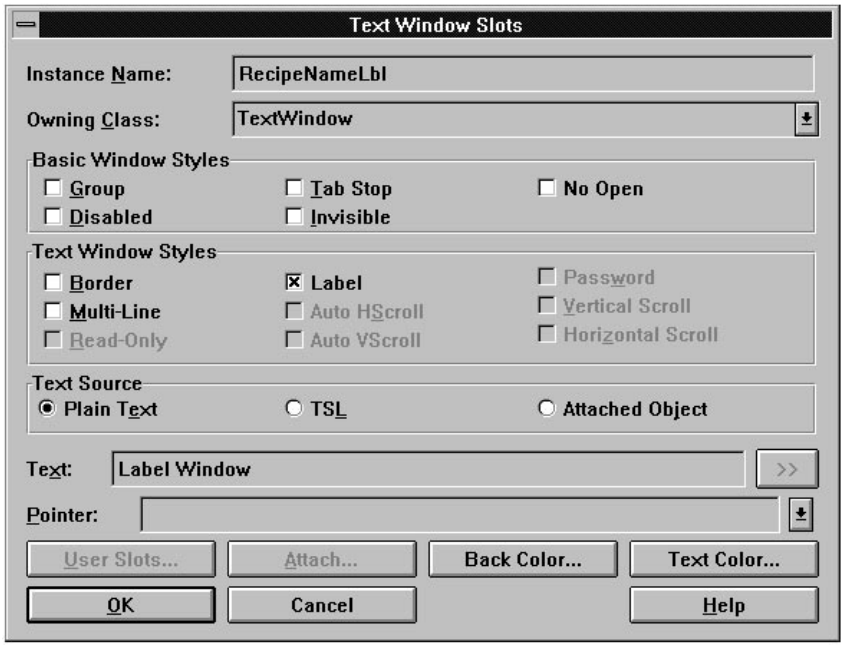
The New Window dialog displays.



2 In the New Window dialog, enter the name of the label window in the Name field and click New to return to the label window.

Name the Recipe Name label window RecipeNameLbl. Name the Ingredients Needed label window RecipeIngredientsLbl.

3 Enter the text displayed in the label window. Double-click in the label window to display the label window slots.



4 In the Text field, enter the text you want displayed in the label window. Click OK to return to the label window.

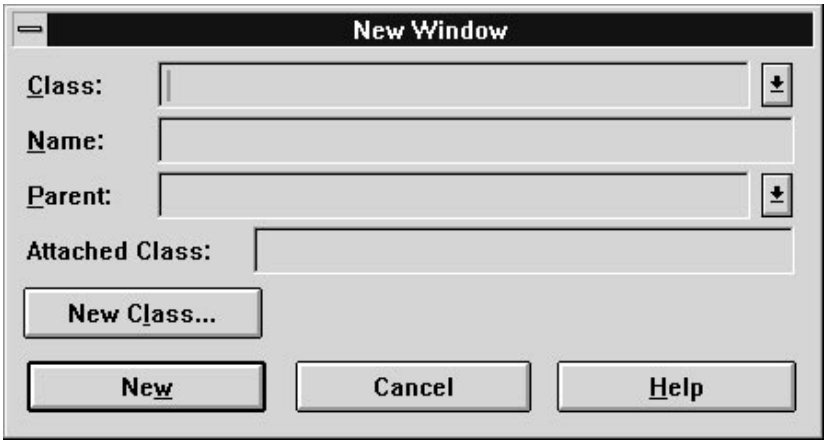
For the Recipe Name label window, enter "`Recipe Name`:". For the Ingredients Needed label window, enter "`Ingredients Needed`:"

5 Resize the label windows to the size shown in the Recipe dialog figure on page 2-28. Place the tip of the arrow on the side of the label window to resize so that the arrow turns into a double-headed arrow. Drag the mouse to pull or push the side in the desired direction.

Adding the Recipe Name text window

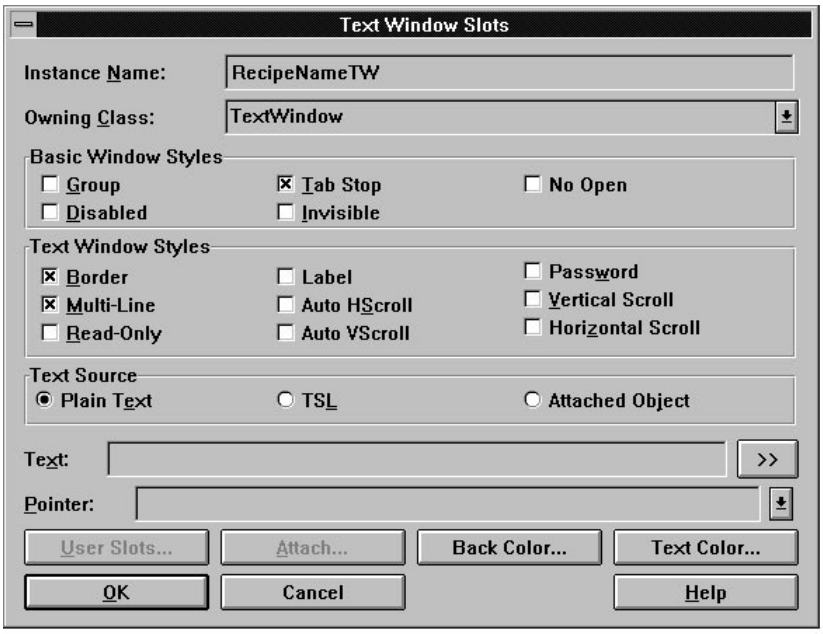
1 Add the Recipe Name text window. Select the Text Window icon from the tool palette and click the cursor (displayed as the Text Window icon) in the Recipe dialog box.

The New Window dialog displays.



2 In the Name field, enter `RecipeNameTW` to match the corresponding name in the New menu item's Function Body property. Click New to return to the Recipe Name text window.

3 You must change some text window styles. Double-click in the Recipe Name text window. The Text Window Slots dialog displays.

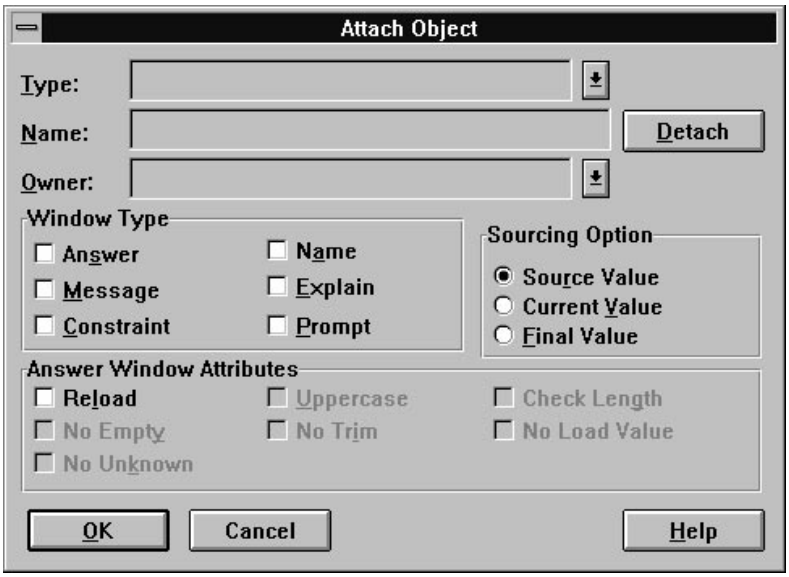


4 Since you only want to create one new recipe at a time, uncheck Multi-Line in the Text Window Styles group.

5 Set up the Recipe Name text window so you can attach it to the Name slot owned by the Recipe class. The function WhenRecipeNewMIChosen_ dynamically attaches the Name slot to the Recipe Name text window.

6 Select Attached Object in the Text Source group.

7 Click Attach. The Attach Object dialog displays.



8 Check Answer in the Window Type group.

9 Check No Empty and No Unknown in the Answer Window Attributes group.

10 Click OK to return to the Text Window Slots dialog.

11 Click OK in the Text Window Slots dialog. You return to the Recipe Name text window.

12 Resize the text window to the size shown in the Recipe dialog figure on page 2-28. Place the tip of the arrow on the side of the text window to resize so that the arrow turns into a double-headed arrow. Drag the mouse to pull or push the side in the desired direction.

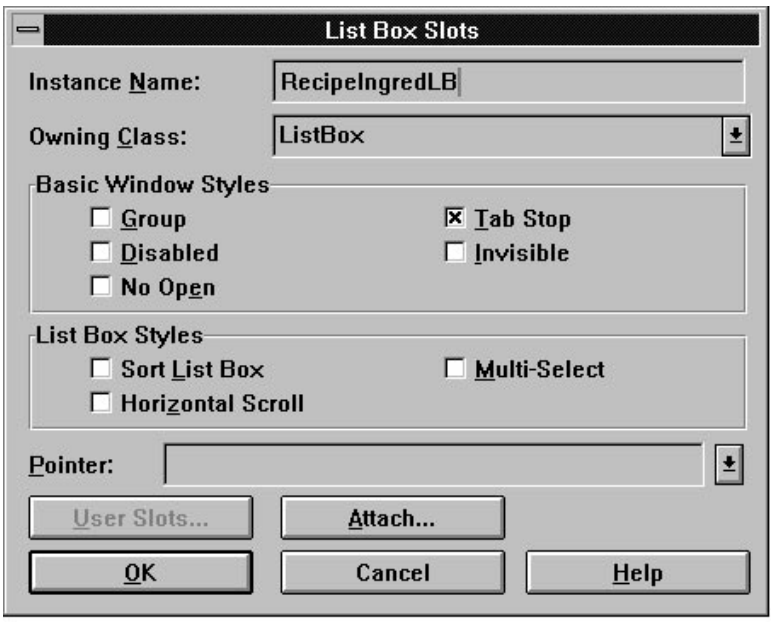
Adding the Ingredients Needed list box

1 Add the Ingredients Needed list box. Select the New List Box icon from the tool palette and click the cursor (displayed as the List Box icon) in the Recipe dialog box.

The New Window dialog displays. The ListBox class displays in the Class field. The RecipeDlg(RecipeDialogBox) window displays in the Parent field. In the Name field, you must name the list box RecipeIngredLB to match the corresponding name in the New menu item's Function Body property.

2 Click New. The Ingredients Needed list box displays.

3 You must change some list box styles. Double-click in the Ingredients Needed list box to display the List Box Slots dialog.



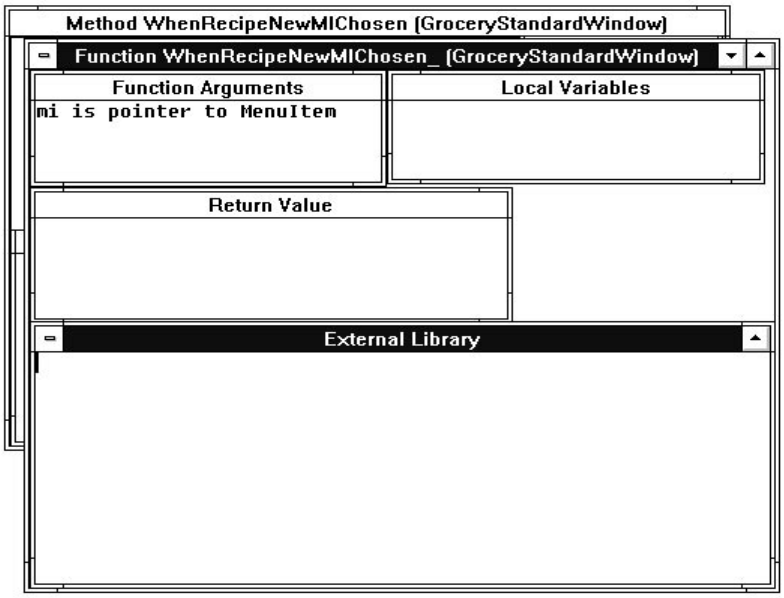
4 Recipes usually contain several ingredients, so check Multi-Select in the List Box Styles group.

5. You must set up the Ingredients Needed list box so you can attach it to the IngredientsNeeded slot owned by the Recipe class. The function WhenRecipeNewMIChosen_ dynamically attaches the IngredientsNeeded slot to the Ingredients Needed list box.
6. Click Attach. The Attach Object dialog displays.
7. Check Answer in the Window Type group.
8. Check No Empty in the Answer Window Attributes group.
9. Click OK to return to the List Box Slots dialog.
10. Click OK in the List Box Slots dialog. You are returned to the Ingredients Needed list box.
11. Resize the list box to the size shown in the Recipe dialog figure on page 2-28. Place the tip of the arrow on the side of the text window to resize so that the arrow turns into a double-headed arrow. Drag the mouse to pull or push the side in the desired direction.

Adding the Cancel push button

1. To add the Cancel push button, select New Push button from the tool palette and click the cursor (displayed as the push button icon) in the Recipe dialog box. It must be named RecipeCancelPB. Resize and display the Cancel push button name by using the same techniques as for the Recipe dialog.

2. Add statements to the function body of the Cancel push button. When you right double-click Cancel, the Methods screen displays. Click OK to create and display the WhenRecipeCancelPBChose Method and Function editors.

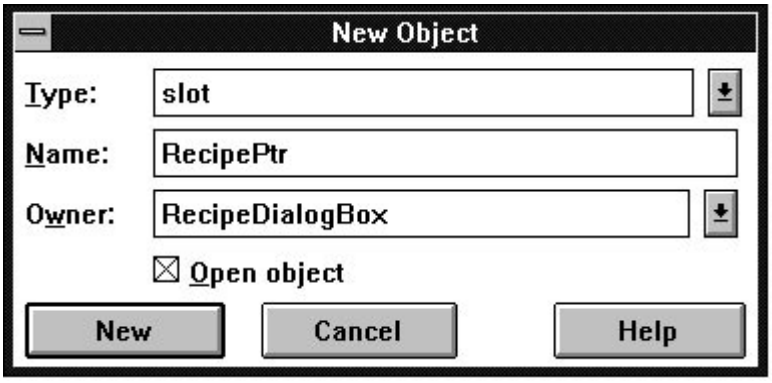


3. Enter the following statement in the Function Body:

```
/*  Delete the Recipe instance and Close the  */
/*  DialogBox.                                */

Delete(RecipePtr->)
close
```

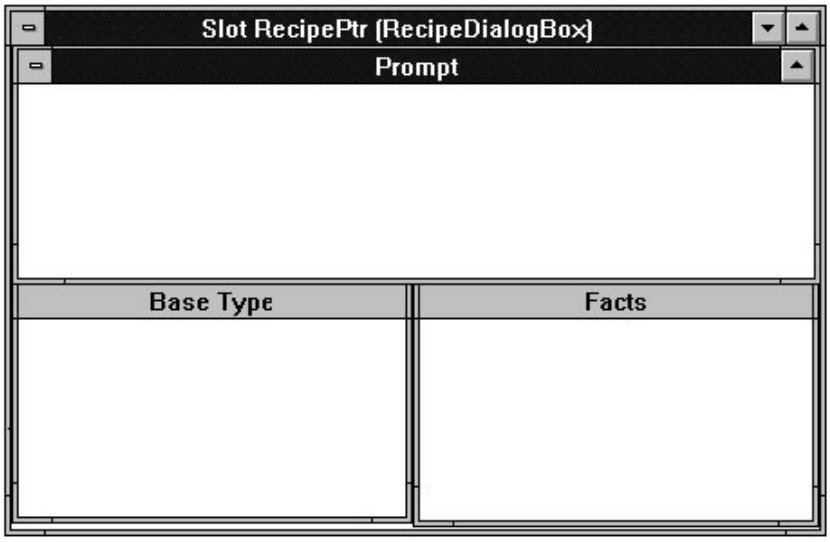
Creating the RecipePtr slot

1 Create a slot named RecipePtr in the RecipeDialogBox class. Select Object.New. The New Object dialog displays.



2 Enter `slot` in the Type field, `RecipePtr` in the Name field, and `RecipeDialogBox` in the Owner field.

3 Click New. AionDS creates the RecipePtr slot and displays the Slot RecipePtr (Recipe) Dialog.



4 Enter the following in the Base Type property:

```
pointer to Recipe
```

Adding the OK push button

1. Add the OK push button. Select New Push button from the tool palette and click the cursor (displayed as the push button icon) in the Recipe dialog box. It must be named RecipeOKPB to match the corresponding name in the New menu item's Function Body property. Resize and display the OK push button name by using the same techniques as for the Recipe dialog.

2. Add statements to the function body of the OK push button to exit to the next statement in the UpdateRecipeValueList state. Double-click the right mouse button to create and display the WhenRecipeOKPBChosen Method and Function editors.

3. Enter the following statement in the Function Body:

```
/*  If the user's answers are successfully  */
/*  assigned to the Recipe slots, update the*/
/*  RecipeValueList, and close RecipeDlg.   */

if AssignAnswers
then
  state(UpdateRecipeValueList)
  close
end
```

4. You cannot save the Recipe dialog until you have created the independent state, UpdateRecipeValueList, which the OK push button function calls.

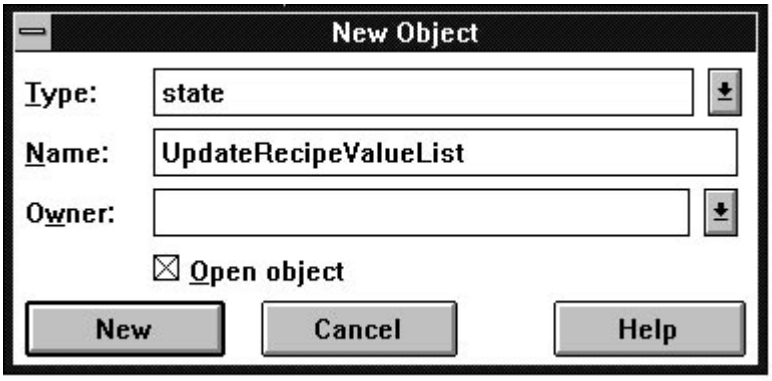
Creating an independent state

Overview

Create an independent state called UpdateRecipeValueList. This state contains the rules to allow new recipes to be selected from the Add Dish dialog.

What to do

1. Activate the Knowledge Base Hierarchy window by selecting it from the Window pull-down menu.
2. Select Object.New to display the New Object window.



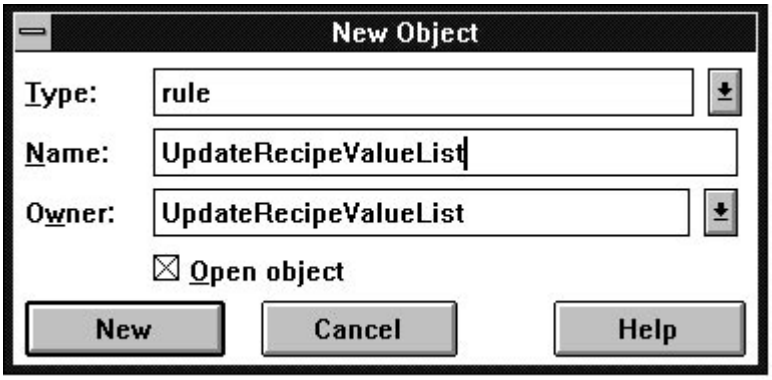
3. In the Type list box, click the down arrow to use the drop-down list and select or enter `state`.
4. Click in (or TAB to) the Name field and enter `UpdateRecipeValueList`.
5. Leave the Owner field blank to create an independent state not owned by any other state.
6. Make sure Open Object is checked. When Open Object is checked, AionDS opens an editor for the object as soon as AionDS creates the object.
7. Click New or press ENTER to create the new state. The state and its Agenda window are created and displayed.

8 In the Agenda window, enter the following:

```
/*  Update the RecipeValueList  */

forwardchain
DishDlg.ResetRecipeValueList
```

9 Save the state by pressing F8. See the new state icon (UpdateRecipeValueList) displayed in the Knowledge Base Hierarchy window.

10 Add a rule as a child of the UpdateRecipeValueList state. Select the UpdateRecipeValueList state in the Knowledge Base Hierarchy window. Select Object.New to display the New Object dialog.



Enter `rule` in the Type field and the name of the rule in the Name field. Click New. The UpdateRecipeValueList state displays in the Owner field, because it was already selected in the knowledge base hierarchy window.

The Rule editor displays.

Enter the following statements in the Rule Definition property:

```
/*  Update RecipeValueList by adding to it   */
/*  Selection Values that are not already    */
/*  included.                                */

ifmatch Recipe with
  not (RecipeValueList includes Recipe.Name)
then
  add Recipe.Name to RecipeValueList
end
```

11 Save the rule by pressing F8.

Changing the Recipe class

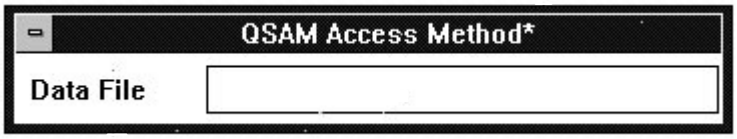
Overview

To create a database file that contains the list of new recipes and link it to the Recipe class:

1. Change the access method to QSAM.
2. Reverse generate the Recipe class definition.
3. Modify the Recipe class definition.
4. Change the Update Mode property.
5. Export the class definition.

What to do

1. Change the access method to QSAM. Double-click the Recipe class name in the Knowledge Base Hierarchy window to open the Recipe class editor. Select Access Method from the Database menu bar. Select QSAM from Access Method. The QSAM Access Method dialog displays.



   In the Data File field, enter `recipe.dat` which is the database file to which you want to link the Recipe class.

2. Reverse generate the Recipe class definition. Select Database.Generate class. This builds the Recipe class definition property.

3. Modify the Recipe class definition. Select Property.Open. The Open Property dialog displays. Double-click the Class Definition property to display the Class Definition property. Change the first line to:

   ```
   file(QSAM, textual) of record
   ```

   By adding the term `textual`, you can view the database file in a more readable format—one record per line—instead of as a single string.

4. Change the Update Mode property to automatic. Select Property.Open to display the Open Property dialog. Double-click Update Mode to display the Update Mode property. Select Automatic.

5. Select Database.Export definition to create the external database file.

Saving and closing all open editors

What to do

1. To close all open editors, select them and press F8.
2. Save the knowledge base to disk by selecting File.Save, or selecting Disk on the button bar.

☑ *Tip:* If parsing errors occur, check the message bar for more information.

Testing the new knowledge base

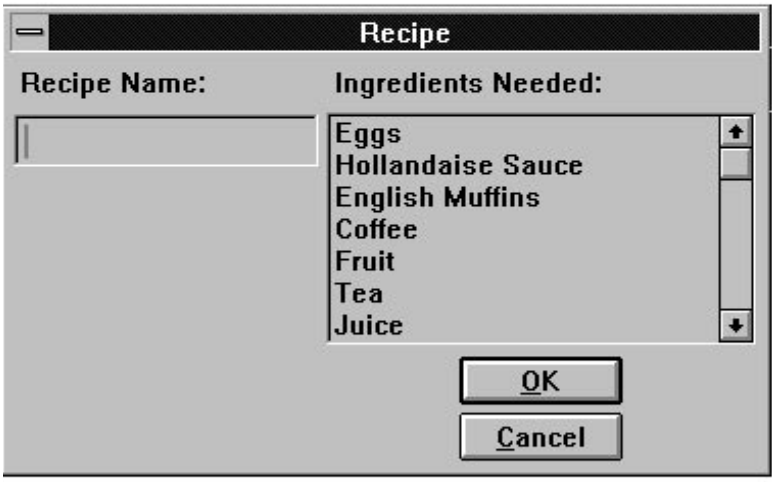
Overview

This section covers testing your new changes. Always test a knowledge base after making major changes to it. If the knowledge base does not execute as expected or even terminates abnormally, use the debugger to identify and solve the problem.

📖 *For more information:* See Chapter 3, "Working at the Knowledge Base Level," for instructions on running the knowledge base with the debugger.

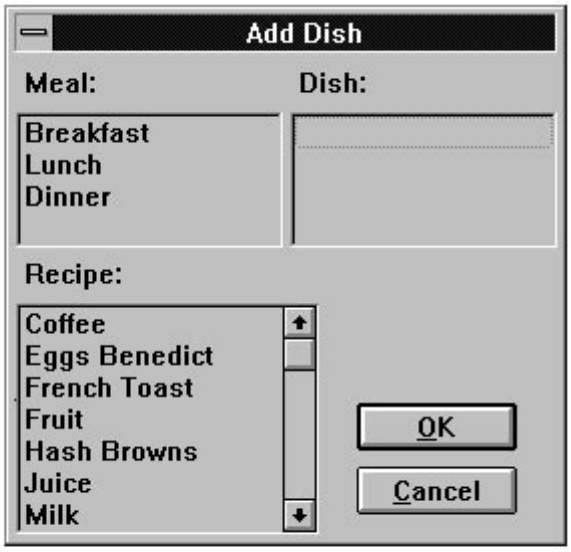
What to do

1. Run the knowledge base by selecting File.Run.
2. Create a new recipe by selecting Recipe.New. The Recipe dialog displays.



Enter a new recipe name in the Recipe Name field and select required ingredients from the Ingredients Needed list box. Click OK.

3 Add the new recipe as a meal in the Menu dialog. Select Menu.AddDish to display the Add Dish dialog.



Select your new recipe from the Recipe list box. Select a meal time and dish for your new recipe by making a selection from the Meal list box and the Dish list box. Click OK.

4 See if your new recipe has been added to the Menu dialog by selecting Menu.Open.

Chapter 3
Working at the Knowledge Base Level

Introduction

This chapter describes how to perform functions affecting a knowledge base. To access knowledge base level functions, use the File menu. This chapter also briefly describes online help.

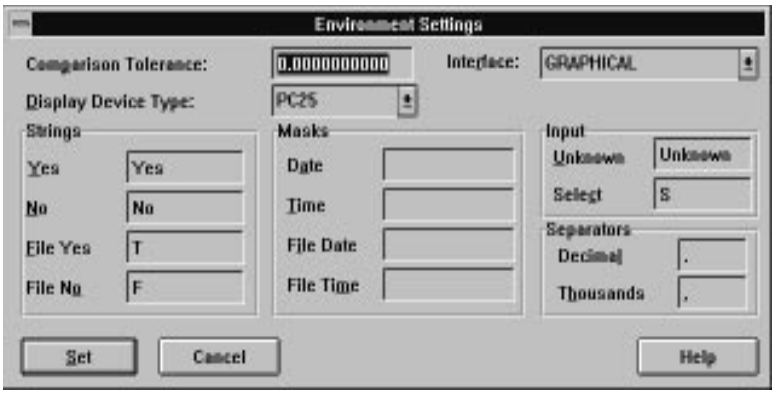
In this chapter

Creating a knowledge base

What to do

To create a knowledge base, set the Interface option in the Environment Settings window to Graphical and use the Create Knowledge Base window.

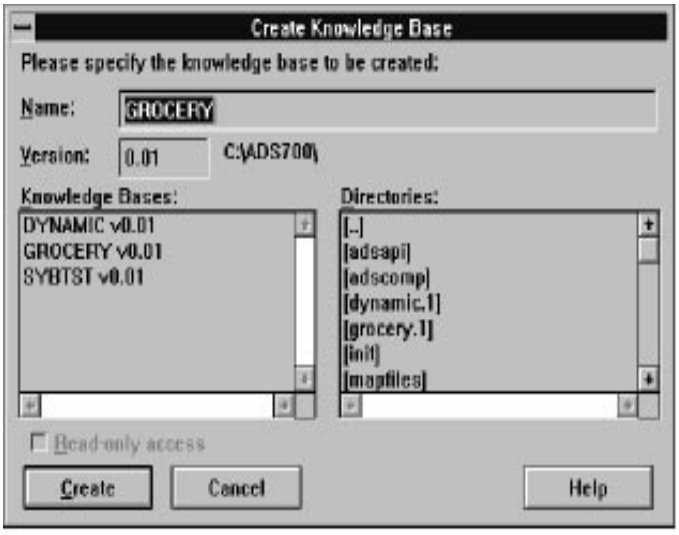
1 To open the Environment Settings window, select Settings.Environment. The Environment Settings window displays.



2 Set the Interface option to Graphical or Character-based, depending on the interface you are using.

📖 *For more information:* See Chapter 10, “Customizing AionDS,” to learn about the other fields in the Environment Settings window.

3 Select File.New. The Create Knowledge Base window displays.



4 If necessary, change the directory in the Directories list box.

5 Enter the name and version of the new knowledge base in the Name and Version fields.

6 Click Create.

Name field

Enter the path and knowledge base name in the Name field.

To change directories, enter the full path name, including a backslash after the last directory.

To go to the root directory, enter a backslash.

Version field

Enter the version number of the knowledge base to open.

If you create a knowledge base and do not enter a version number, the knowledge base enters 0.01 as the default version number. AionDS uses 0.01 to identify Version 1. To work on a new version of a knowledge base, use Copy or Update to make a copy of the knowledge base, and give it a version number.

AionDS increments version numbers by 0.01. For example, the second version of a knowledge base is 0.02.

Current path

The current directory displays to the right of the Version field. When you enter the full path and knowledge base name, AionDS ignores the current path.

By default, the current directory is one of the following:

- from the command line—the directory from which you started AionDS
- from the AionDS icon—the working directory associated with AionDS in its program group

Knowledge Bases and Directories list boxes

To select a directory name from the Directories list box, double-click on the directory name.

All available knowledge bases in the selected directory display in the Knowledge Bases list box.

To select a knowledge base in the Knowledge Bases list box, click on the knowledge base.

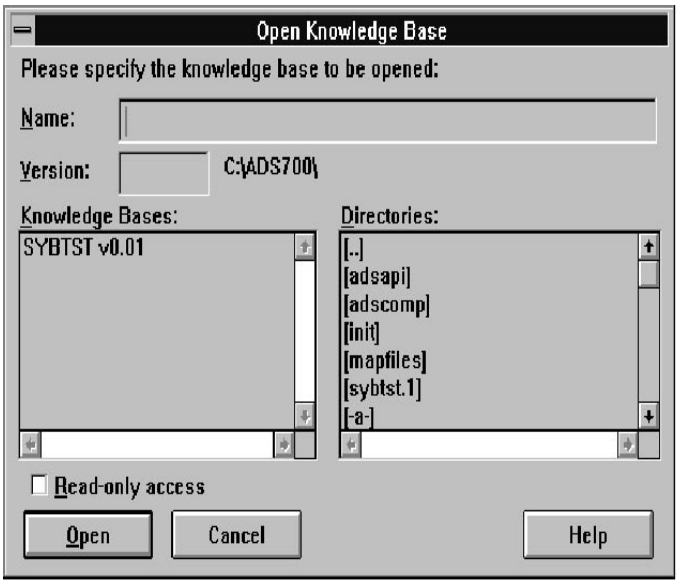
Note: You cannot create a knowledge base with read-only access.

Opening a knowledge base

What to do

To open a knowledge base, use the Open Knowledge Base window. You can open only one knowledge base at a time.

1 Select File.Open. The Open Knowledge Base window displays.



2 If necessary, change the directory in the Directories list box.

3 Enter the name and version of the knowledge base to open.

4 Click Open.

5 If the knowledge base has a password set, AionDS prompts you to enter the password.

Name field

Enter the name of the knowledge base in this field, select a knowledge base name from the Knowledge Bases list box, or enter the full path and knowledge base name.

To change directories, enter the full path name, including a backslash after the last directory.

To access the root directory, enter a backslash.

If the Version field and the Read-only access check box are correct, click Open.

Version field

Enter the version number of the knowledge base to open. If you select a knowledge base from the Knowledge Bases list box, that version number displays in the Version field.

Current path

The current path displays to the right of the Version field.

By default, the current path is one of the following:

- from the command line—the directory from which you started AionDS
- from the AionDS icon—the working directory associated with AionDS in its program group

Knowledge Bases and Directories list boxes

To select a directory name from the Directories list box, double-click on the directory name. All knowledge bases available in the selected directory display in the Knowledge Bases list box.

To select a knowledge base in the Knowledge Bases list box, click on the knowledge base.

Note: If you check Read-only access, you can only browse, not modify, the knowledge base.

Saving a knowledge base

What to do

➢ To save work in progress to disk, select File.Save or click the save icon in the button bar. AionDS saves to disk any changes to knowledge base objects you saved with Object.Save. AionDS also prompts you to save any unsaved objects.

☯ *Alternative:* AionDS prompts you to save your work. To specify prompt intervals, select Settings.Display and specify the interval in minutes in the Auto-Save field. To deactivate auto-save, enter 0.

Save your knowledge base periodically. Using Object.Save to save changes to objects does not update the knowledge base stored on disk. Export your knowledge base periodically. Exporting a knowledge base saves it in an external file format.

🕮 *For more information:* See Chapter 10, "Customizing AionDS," for details about the Display Settings dialog.

✋ *Caution:* If you save objects with syntax errors, AionDS does not prompt you to resolve the errors before saving the knowledge base.

Closing the current knowledge base

What to do

To close a knowledge base:

1. Select File.Close.
2. AionDS prompts you to save objects with unsaved changes.

   To save all changes made to an object, click Save.

   To close a knowledge base without saving the changes, click Discard.

   Result: AionDS clears the knowledge base from memory. Closing a knowledge base does not end an AionDS session.

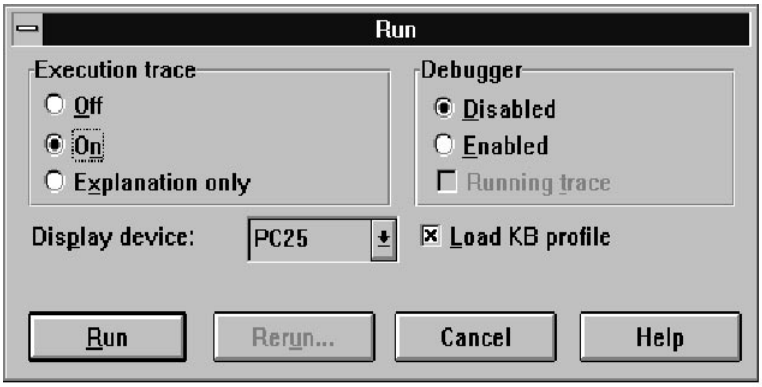
   ☑ *Tip:* You can change the default sizes and positions of object editor windows. The new default sizes and positions are not saved automatically. For more information, see Chapter 10, "Customizing AionDS."

Running a knowledge base

What to do

1 To execute a knowledge base, select File.Run, or select the Run icon in the button bar.

The Run window displays.



📖 *For more information:* See the Run Settings dialog in Chapter 10, "Customizing AionDS," for details about the defaults that control the execution of a knowledge base.

2 Use the execution trace radio buttons to specify whether the trace is on or off, and what type of trace to write.

📖 *For more information:* See Chapter 19, "Trace File," of the *General Reference* to learn more about the execution trace.

3 In the Debugger group, enable or disable the debugger.

📖 *For more information:* See Chapter 9, "Debugging Knowledge Bases," to learn how to use the debugger.

4 Use the Display device field to indicate the type of display the knowledge base execution simulates on the desktop. Select the field to display the following screen types:

Device-Type	Terminal	Rows	Columns
*	all	any	any
PC*	PC	any	any
PC25	PC	25	80
PC43	PC	43	80
PC50	PC	50	80
M*	3270	any	any
M2	3270	24	80
M3	3270	32	80
M4	3270	43	80
M5	3270	27	132

5 When you check Load KB profile, AionDS automatically loads the profile associated with this knowledge base. If you do not check this check box, AionDS does not load the associated profile, even if the profile is available in the knowledge base directory.

6 Click Run to execute the knowledge base. Execution does not begin until the knowledge base is saved. If you try to execute a knowledge base before saving, AionDS prompts you to save the knowledge base with a confirmation pop-up window.

Viewing knowledge base objects

You can switch to other AionDS editor windows while running a knowledge base. Any window you open while running a knowledge base reflects its current state. This means that you can view changes AionDS makes to a running knowledge base.

For example, you may want to view instances created during knowledge base execution.

📖 *For more information:* See Chapter 4, "Working with Object Lists," to learn more about how to view instances and objects.

Editing knowledge base objects

You cannot change objects while a knowledge base is running.

Aborting and restarting knowledge base execution

What to do

- To abort a running knowledge base quickly (for example, one in an infinite loop), select Run.Abort.
- To restart knowledge base execution quickly, select Run.Restart.

Exception: The Restart command is not available in GUIAionDS/Win.

Managing libraries

Overview

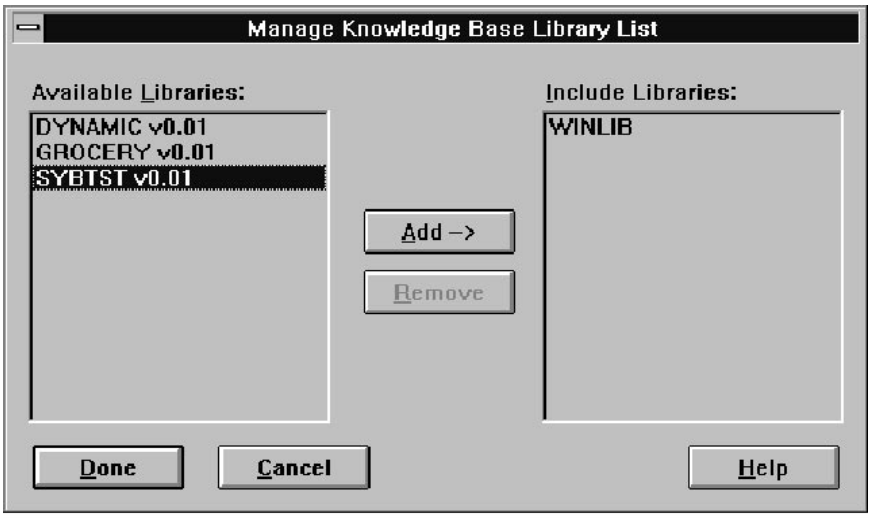
Managing libraries comprises the following tasks:

- adding libraries to knowledge bases
- deleting libraries from knowledge bases
- updating modified libraries to knowledge bases

Adding libraries

What to do

1 Select File.Libraries. The Manage Knowledge Base Library List dialog displays.



2 From the Available Libraries list, select the libraries to add by clicking on each. You can select more than one library.

☑ *Tip:* The Available Libraries list displays the libraries in the AionDS directory. To include a library in a knowledge base, place the library in the AionDS directory or DPATH.

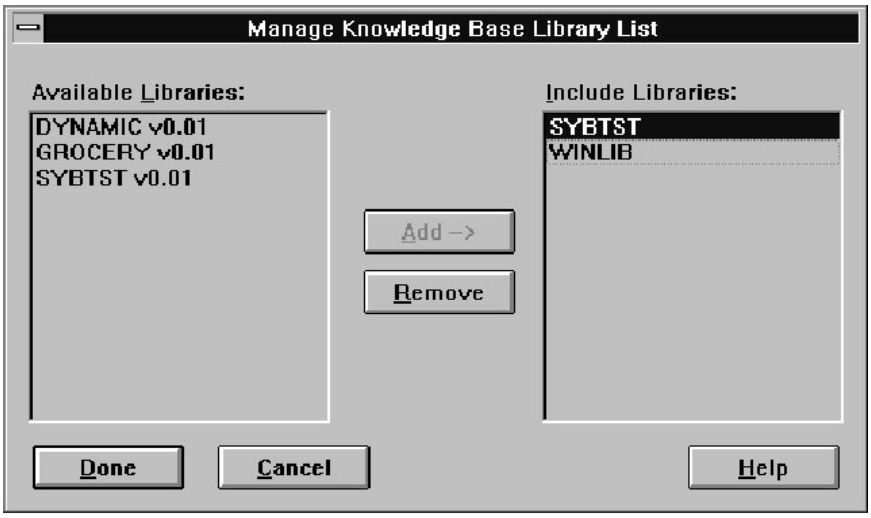
3 Click Add. AionDS adds the selected libraries to the Include Libraries list.

4 Click Done. AionDS imports the knowledge base and adds the libraries found in the Include Libraries list.

Deleting libraries

What to do

1 Select File.Libraries. The Manage Knowledge Base Library List dialog displays.



2 In the Include Libraries list, select the libraries to delete.

3 Click Remove. AionDS deletes the selected libraries from the Include Libraries list.

4 Click Done. AionDS removes the selected libraries from the knowledge base, and the knowledge base is reimported.

Updating modified libraries to knowledge bases

Overview

Changes made to objects in libraries are not automatically updated to the same libraries included in knowledge bases. When you open a knowledge base with modified libraries, AionDS does not update the knowledge base's included libraries. To update the included libraries, import the knowledge base or include the libraries in the knowledge base again.

What to do

To import a knowledge base:

1. Open the knowledge base.
2. Select File.Import to import the knowledge base and update the libraries.

To include libraries again:

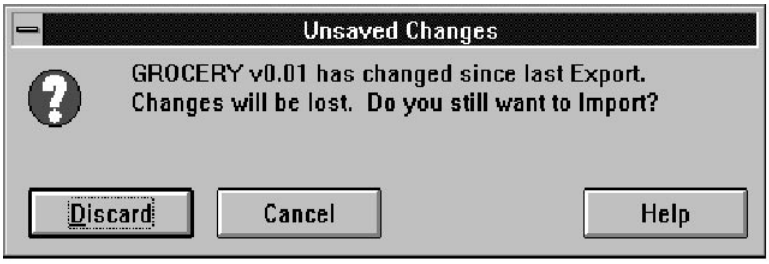
1. Open the knowledge base that includes the modified libraries.
2. Select File.Libraries. The Manage Knowledge Base Library List dialog displays.
3. Click Done. AionDS updates the included libraries.

Importing a knowledge base

Importing an open knowledge base

To have AionDS re-import a knowledge base, select Import when the knowledge base is open.

1 Select File.Import. If you made changes to the knowledge base since the last export, AionDS displays a confirmation window asking if you want to import the knowledge base.



2 To import the knowledge base, click Discard.

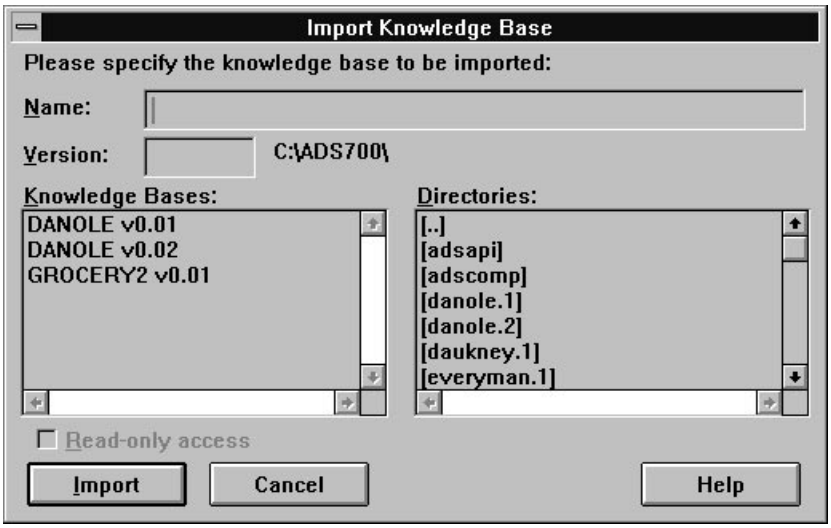
Result: AionDS imports the knowledge base.

3 If you decide not to import the knowledge base, click Cancel.

Result: You return to the knowledge base.

Importing a closed knowledge base

1. Close the open knowledge base.
2. Select File.Import. The Import Knowledge Base window displays.



3. If necessary, change the directory in the Directories list box.
4. Select a knowledge base in the Knowledge Bases list box.
5. Click Import.

Name field

To access a knowledge base, use the name field in one of the following ways:

- enter the name of the knowledge base
- select a name from the knowledge base's list box
- enter the full path and knowledge base name

To change directories, enter the directory path name, including a backslash after the last directory. To go to the root directory, enter a backslash.

If the Version field is correct, click Import.

Version field

To import, enter the version number of the knowledge base. If you select a knowledge base from the Knowledge Bases list box, that version number displays in the Version field.

AionDS increments the version number by 0.01. For example, the second version of a knowledge base is 0.02.

Current path

The current path displays on the right of the Version field. To change the current path, use the Directories list box or the Name field. When you enter the full path and knowledge base name, AionDS ignores the current path.

By default, the current path is one of the following:

- from the command line—the directory from which you started AionDS
- from the AionDS icon—the working directory associated with AionDS in its program group

Knowledge Bases and Directories list boxes

To select a directory name from the Directories list box, double-click on the directory name. The selected directory displays in the Knowledge Bases list box. To include a library in a knowledge base, place the library in the AionDS directory or DPATH.

To select a knowledge base in the Knowledge Bases list box, click on the knowledge base. AionDS includes the latest version of each library.

☞ *Note:* You cannot import a knowledge base with read-only access.

Exporting a knowledge base

What to do

Use Export to generate the external format of a knowledge base. Export your knowledge base frequently; this creates a backup copy. The knowledge base must be open.

- Select File.Export.

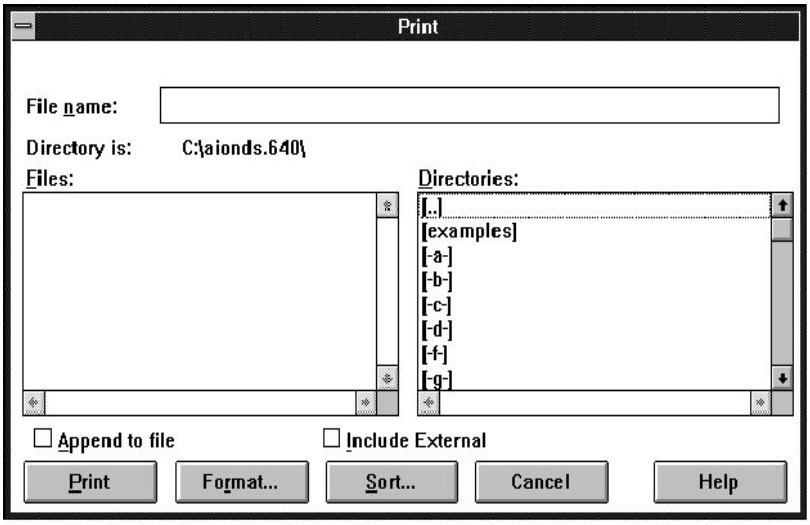
Printing object information

What to do

To print knowledge base object information or save it (in text format) in a file, use the Print window.

1 Select File.Print.

To print selected knowledge base objects, first select objects from an object list window, then select Object.Print.



2 Specify the file to which to save the object information in the File name field. You can enter a path name or use the Directories list box to specify a path.

3 To print out object information, enter the name of your printer (in many installations this is LPT1 or PRN). See your operating system documentation for the exact name of your printer.

4 To append object information to the end of a file, select the file; then select the Append to file check box.

5 To place object information in a new file, enter the path and file name in the File name field and select the Append to file check box.

6 To overwrite the contents of a file with object information, select the file and make sure the Append to file check box is **not** selected.

7 To specify the print order of object information, click Sort. The Print Sort window displays.

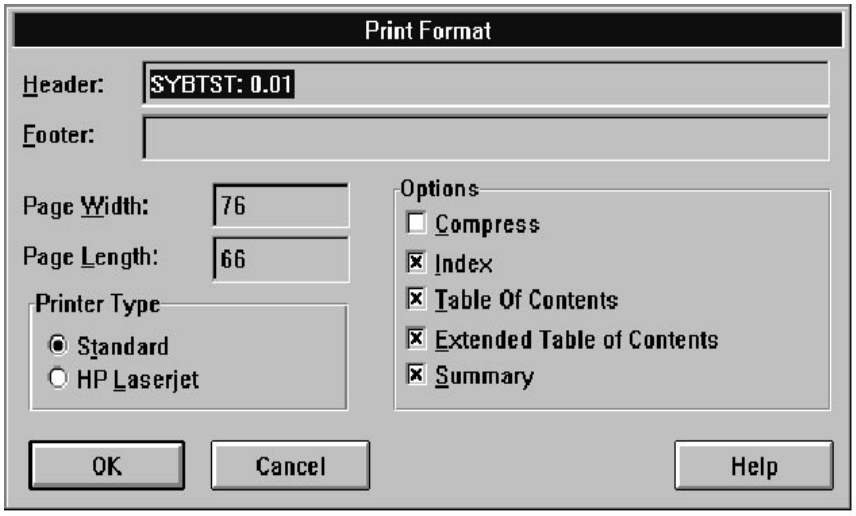
8 Change the print sort fields, then click OK to return to the Print window.

9 To perform any of the following tasks, click Format. The Print Format window displays.

- enter header and footer text for each page
- change page width and length
- indicate printer type
- compress output into minimum number of lines
- add index to end of printout
- add table of contents to end of printout
- add extended table of contents containing entries for all objects to end of printout
- add summary of all objects printed to end of printout

After changing the print format attributes, click OK to return to the Print window.

Print Format window



From this window, you can:

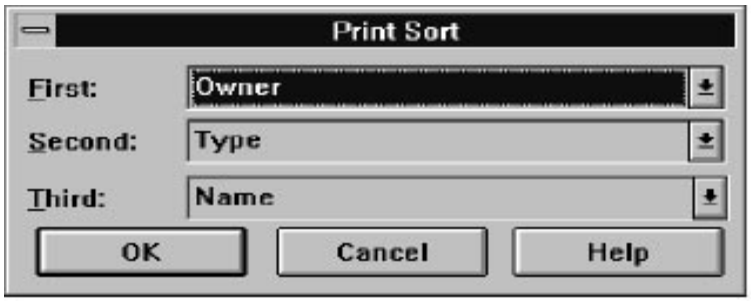
- Enter the header and footer text for each page. The default header is the knowledge base name and version number.
- Change the page width and length. The default page length is 66 lines and the default page width is 76 characters.
- Indicate the printer type.

Use the following five check boxes under Options to modify the way the knowledge base prints:

Check box	If checked ...
Compress	compresses the output into as few lines as possible
Index	adds an index to the end of the printout
Table of Contents	adds a table of contents to the end of the printout
Extended Table of Contents	adds an extended table of contents containing entries for all objects to the end of the printout
Summary	adds a summary of all the objects printed to the end of the printout

Print Sort window

From the Print Sort window, you can select the first, second, and third order in which objects sort for printing.



The Print Sort window has three drop-down list boxes. Using the list boxes, you can sort the knowledge base objects alphabetically by their parent, type, and name. The default is to sort objects alphabetically by parent name, type within parent name, and object name within type. To change this order, change the values in the drop-down list boxes.

Browsing a file

What to do

1. To browse text files, select File.Browse. AionDS displays the Browse window.
2. Select a file from the window. AionDS starts a session of the Browser for that file.

In AionDS/2

AionDS/2 calls the system editor, E.EXE. The browse session is independent of AionDS. The AionDS primary window does not own the system editor.

You can edit files using the system editor.

☑ *Tip:* One way to take advantage of the system editor is to add commentary before printing the browse file.

In AionDS/Win

AionDS/Win calls the AionDS/Win Browser. The AionDS/Win Browser is a child of the primary window called. You cannot edit files using the AionDS/Win Browser.

Tracking changes in a knowledge base

Overview

To track, document, and control changes to a knowledge base, use the Change Management sub-menu. You can use Change Management to incorporate changes made simultaneously by several developers.

Change Management commands

The following table lists each Change Management command and its function:

Command	Function
Check out	creates a working copy of the master knowledge base
Check in	moves the change set from the current knowledge base to the master
Compare	checks for change set changes to the same object property and lists those changes
Apply	makes the changes in the selected change sets to the current knowledge base
List	lists the change sets in the current knowledge base
Open	opens one or more change sets in the current knowledge base
Print	prints modified objects in change sets or saves them (in text format) in a file
Enable/Disable	starts or stops recording changes to the change set

📖 *For more information:* See Chapter 8, "Tracking Changes in a Knowledge Base," to learn about Change Management procedures and menu commands, as well as to see an example of using Change Management.

Knowledge base utilities

Overview

Use the File.Utilities to perform the following functions on an existing knowledge base:

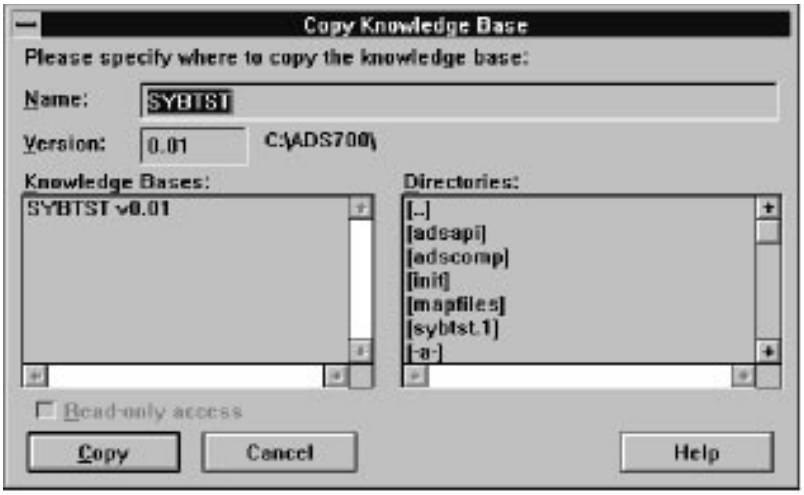
Command	Function	Page
Copy	makes a copy of the knowledge base; you can specify a new name, version, and path	3-28
Update	creates a copy of a knowledge base and increments the version number; you can specify a new path or name	3-31
Delete	deletes a knowledge base from disk	3-33
Move	moves a knowledge base to a new path or renames it in the current directory	3-36
Prepare	strips a knowledge base of any unnecessary text in preparation for a production version of the knowledge base	3-38
Attributes	sets a title for the knowledge base and displays information, such as when the knowledge base was last modified	3-41
Passwords	sets runtime and edit-time passwords	3-42

Copying a knowledge base

What to do

To make a copy of a knowledge base:

1. Open the knowledge base to copy.
2. Select File.Utilities.Copy. AionDS displays the Copy Knowledge Base window.



3. Select the directory in which to save the copy.

   ☯ *Alternative:* You can specify the disk and path name in the Name field.

4. Enter the knowledge base name for the copy. To keep the same knowledge base name for the copy as the original, select the original name in the Knowledge Bases list box.

5 If the knowledge base has a unique name within the directory, let the version number default to the source knowledge base version number; or you can enter a new version number. If the knowledge base name is the same as a knowledge base in a directory, you must specify a unique version number.

6 After entering the information, click Copy. The next time you view a Knowledge Base window, the copy displays in the appropriate directory. The original knowledge base remains open.

Name field

Enter the name of the knowledge base in this field:

Select a knowledge base name from the Knowledge Bases list box, or enter the full path and knowledge base name.

To change directories, enter the path name to the directory, including a backslash after the last directory.

To go to the root directory, enter a backslash.

Check that the Version field is correct, then click Copy.

Version field

Enter the version number of the knowledge base. If you select a knowledge base from the Knowledge Bases list box, that version number displays.

Current path

The current path displays to the right of the Version field. To change the current path, use the Directories list box or the Name field. When you type the full path and knowledge base name, AionDS ignores the current path.

By default, the current path is one of the following:

- from the command line—the directory from which you started AionDS
- from the AionDS icon—the working directory associated with AionDS in its program group

Knowledge Bases and Directories list boxes

To select a directory name from the Directories list box, double-click on the directory name. All knowledge bases in the selected directory display in the Knowledge Bases list box.

To select a knowledge base in the Knowledge Bases list box, click on the knowledge base.

Note: You cannot copy a knowledge base with read-only access.

Updating a knowledge base

Overview

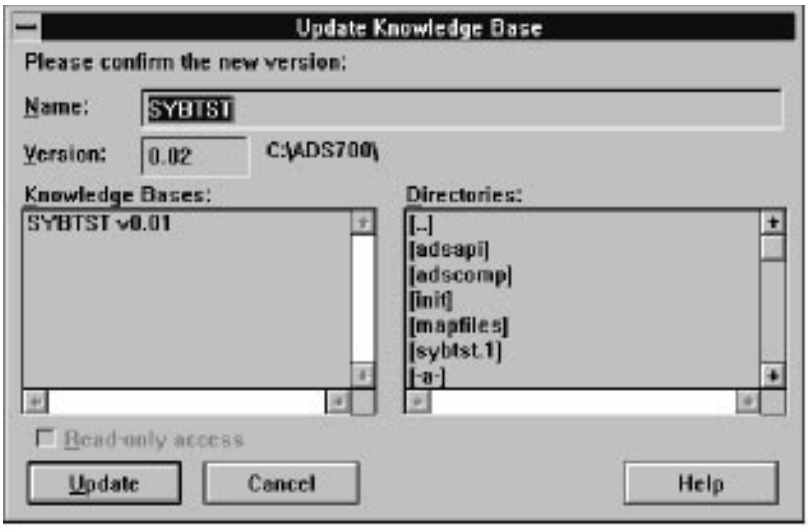
Updating a knowledge base is similar to copying, except that:

- Update closes the source knowledge base (the knowledge base that was open when you chose Update) and opens the new copy.
- The Update Knowledge Base window automatically increments the version field by one number.

Typically, you use defaults to make a copy of a knowledge base.

What to do

1. Open the knowledge base.
2. Select File.Utilities.Update. The Update Knowledge Base window displays.



3. You can change the name, path, and version number.
4. Click Update.

Name field

The name of the current knowledge base displays. To change the name in this field:

- select a name from the Knowledge Bases list box, or
- type the full path and knowledge base name

To change directories, enter the path name to the directory, including a backslash after the last directory.

To go to the root directory, enter a backslash.

Check that the Version field is correct, then click Update.

Version field

AionDS increments the version of the current knowledge base by one number and displays the change in the Name field. You can change the version number. If you select a different knowledge base to update from the Knowledge Bases list box, that version number displays in the Version field.

Current path

The current path displays to the right of the Version field. To change the current path, use the Directories list box or the Name field. When you enter the full path and knowledge base name, AionDS ignores the current path.

By default, the current path is one of the following:

- from the command line—the directory from which you started AionDS
- from the AionDS icon—the working directory associated with AionDS in its program group

Knowledge Bases and Directories list boxes

To select a directory name from the Directories list box, double-click on the directory name. All knowledge bases in the selected directory display in the Knowledge Bases list box.

To select a knowledge base in the Knowledge Bases list box, click on the knowledge base.

Note: You cannot update a knowledge base with read-only access.

Deleting a knowledge base

Overview

To delete the current knowledge base, select Delete with the knowledge base open. AionDS closes and deletes the open knowledge base. If no knowledge base is open, the Delete Knowledge Base window displays.

Deleting an open knowledge base

1 Select File.Utilities.Delete. A confirmation dialog displays.

2 To delete the knowledge base, click Delete.

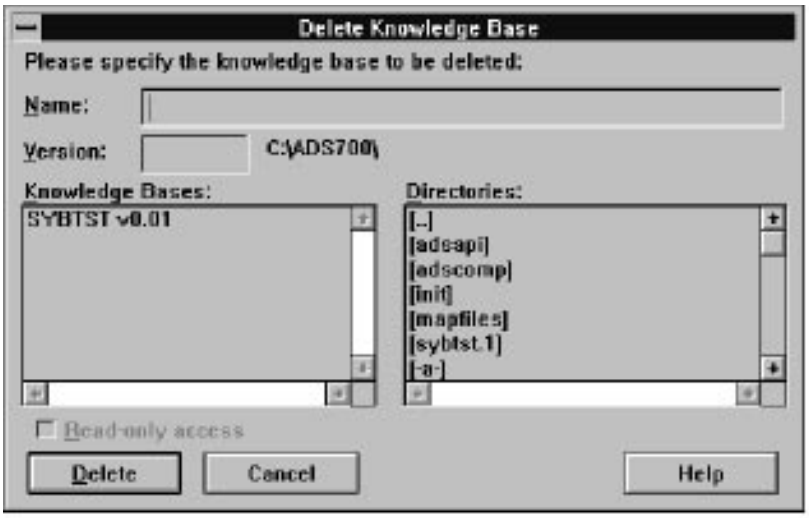
Result: AionDS closes and deletes the open knowledge base from disk.

3 If you do not want to delete the knowledge base, click Cancel.

Result: You return to the open knowledge base.

Deleting an unopened knowledge base

1. To delete an unopened knowledge base from disk, close the open knowledge base.
2. Select File.Utilities.Delete. The Delete Knowledge Base window displays.



3. Use the Directories list box to select the directory in which the knowledge base is located.
4. Select the knowledge base.
5. Click Delete.
6. The confirmation dialog displays.

   Result: If you click Delete on the confirmation dialog, AionDS deletes the selected knowledge base from disk and returns you to the primary window.

   If you click Cancel from the confirmation dialog, AionDS returns you to the Delete Knowledge Base window.

Name field

To enter the name of the knowledge base in this field:

- select a knowledge base name from the Knowledge Bases list box, or
- type the full path and knowledge base name

To change directories, enter the full path name to the directory, including a backslash after the last directory.

To go to the root directory, enter a backslash.

Check that the Version field is correct, then click Delete.

Version field

Enter the version number of the knowledge base to delete. If you select a knowledge base to delete from the Knowledge Bases list box, that version number displays in the Version field.

Current path

The current path displays to the right of the Version field. To change the current path, use the Directories list box or the Name field. When you enter the full path and knowledge base name, AionDS ignores the current path.

By default, the current path is one of the following:

- from the command line—the directory from which you start AionDS
- from the AionDS icon—the working directory associated with AionDS in its program group

Knowledge Bases and Directories list boxes

To select a directory name from the Directories list box, double-click on the directory name. All knowledge bases in the selected directory display in the Knowledge Bases list box.

To select a knowledge base in the Knowledge Bases list box, click on the knowledge base.

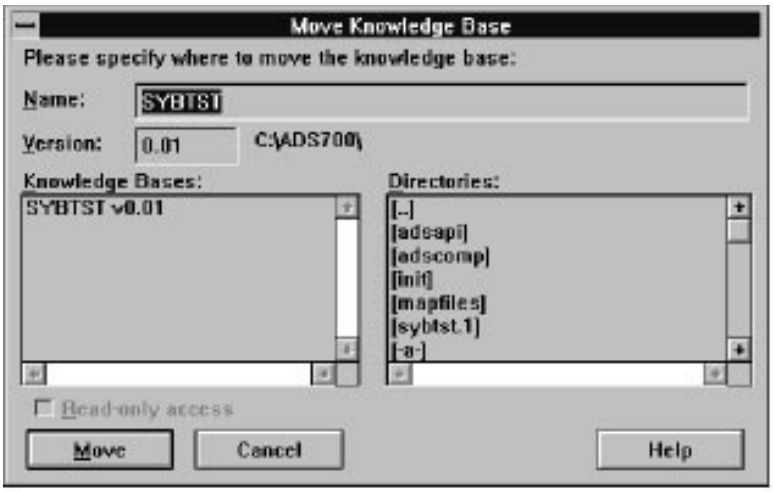
Note: Read-only access does not apply to deleting knowledge bases.

Moving a knowledge base

What to do

To move a knowledge base:

1 Select File.Utilities.Move. The Move Knowledge Base window displays.



2 Enter the destination knowledge base path, name, and version.

If the knowledge base is unique to its directory, you can let the version number default to the current version number, or enter a new version number. If the knowledge base name exists in the destination directory, you must change the knowledge base name or give it a unique version number, or both.

3 After entering the information, click Move. If you are overwriting the same knowledge base, an error message displays. You cannot move a knowledge base onto itself.

Name field

The open knowledge base name displays in the Name field. To change the knowledge base name:

- select a name from the Knowledge Bases list box, or
- enter the full path and name

To change directories, enter the path name to the directory, including a backslash after the last directory.

To go to the root directory, enter a backslash.

Check that the Version field is correct, then click Move.

Version field

The open knowledge base version number displays in the Version field. You can change the version number of the knowledge base to move. If you select a knowledge base to move from the Knowledge Bases list box, that version number displays in the Version field.

Current path

The current path displays to the right of the Version field. To change the current path, use the Directories list box or the Name field. AionDS ignores the current path when you type the full path and knowledge base name.

By default, the current path is one of the following:

- from the command line—the directory from which you started AionDS
- from the AionDS icon—the working directory associated with AionDS in its program group

Knowledge Bases and Directories list boxes

To select a directory name from the Directories list box, double-click on the directory name. All knowledge bases in the selected directory display in the Knowledge Bases list box.

To select a knowledge base in the Knowledge Bases list box, click on the knowledge base.

Note: You cannot move a knowledge base with Read-only access selected.

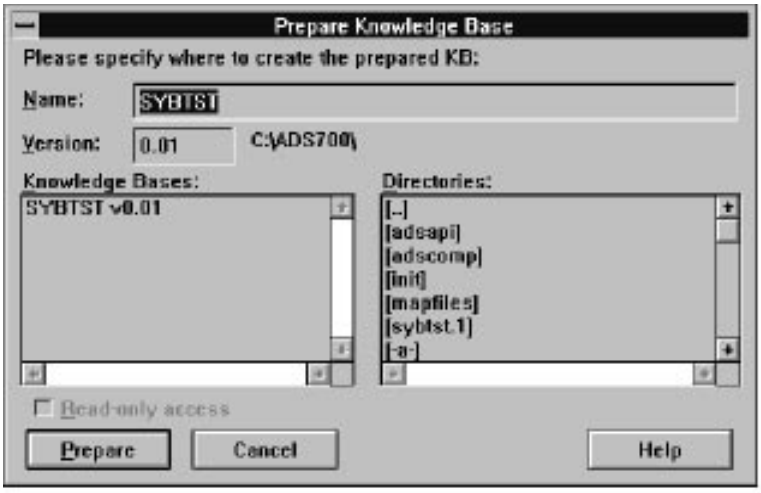
Preparing a knowledge base

What to do

Use Prepare to strip your knowledge base of any text not needed for the production version. To ensure the integrity of your original knowledge base, be sure to change the version number.

To prepare a knowledge base:

1. Open the knowledge base.
2. Select File.Utilities.Prepare. The Prepare Knowledge Base window displays.



3 Change the version number in the Version field.

4 To prepare a different knowledge base, use the Directories list box to select the directory in which the knowledge base is located.

5 Select the knowledge base.

6 Click Prepare.

Result: AionDS prepares and opens the knowledge base. The knowledge base hierarchy window does not display. You can run, but cannot edit, the knowledge base.

Name field

The knowledge base name displays in the Name field. To change the knowledge base name:

- select from the Knowledge Bases list box, or
- enter the full path and name

To change directories, enter the full path name to the directory, including a backslash after the last directory.

To go to the root directory, enter a backslash.

Check that the Version field is correct, then click Prepare.

Version field

The open knowledge base's version number displays in the Version field. To use Prepare , you must change the version number of the knowledge base. If you select a knowledge base from the Knowledge Bases list box, the corresponding version number displays in the Version field.

AionDS increments the version number by 0.01. For example, the second version of a knowledge base is 0.02.

Current path

The current path displays to the right of the Version field. To change the current path, use the Directories list box or the Name field. When you enter the full path and knowledge base name, AionDS ignores the current path.

By default, the current path is one of the following:

- from the command line—the directory from which you started AionDS
- from the AionDS icon—the working directory associated with AionDS in its program group

Knowledge Bases and Directories list boxes

To select a directory name from the Directories list box, double-click on the directory name. All knowledge bases in the selected directory display in the Knowledge Bases list box.

To select a knowledge base in the Knowledge Bases list box, click on the knowledge base.

Note: You cannot prepare a knowledge base with Read-only access selected.

Setting knowledge base attributes

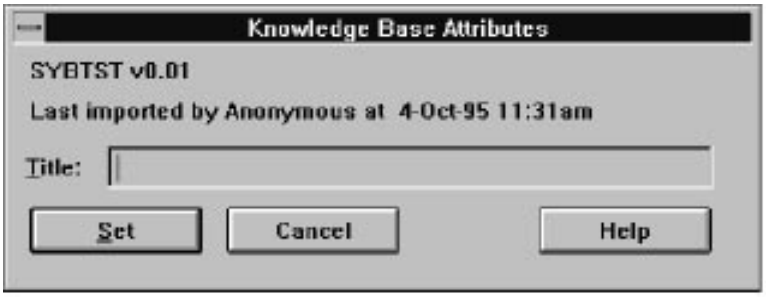
Overview

Use Attributes to assign a title to, and display information about, the knowledge base.

What to do

To assign a title to a knowledge base:

1. Open the knowledge base.
2. Select File.Utilities.Attributes. The Knowledge Base Attributes window displays.



3. To assign a title to a knowledge base, type a title in the Title text entry field. AionDS limits the title to 30 characters, including spaces.
4. Click Set.

Setting a password

Overview

You can set the following three types of passwords:

- Owner

  If you use an Owner password to open a knowledge base, you can change an object's authorization and set passwords.

- User

  If you use a User password to open a knowledge base, you cannot change an object's authorization, set passwords, see hidden objects, or modify read-only objects. To make an object read-only, you must change the object's authorization.

- Runtime

  If a knowledge base has a Runtime password, you cannot run a knowledge base without the password.

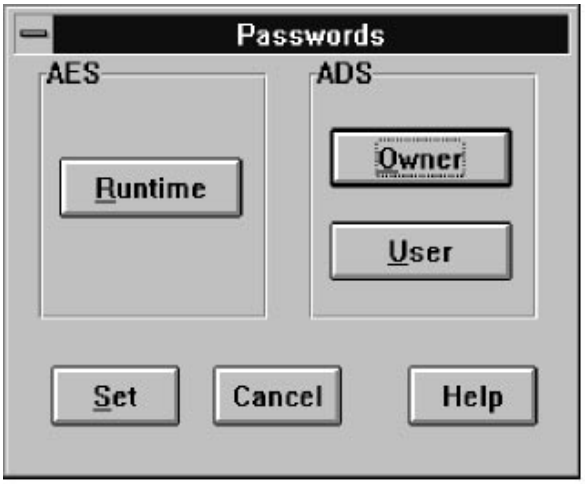
📖 *For more information:* See Chapter 5, "Working with Objects," to learn how to change an object's authorization.

What to do

To assign an Owner, User, or Runtime password:

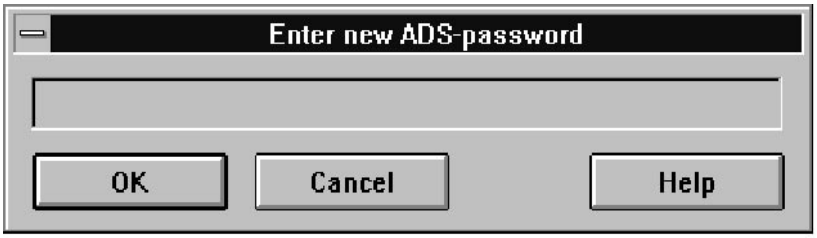
1. Open the knowledge base.
2. Select File.Utilities. The Utility sub-menu displays.

3 Select Passwords. AionDS displays the Passwords window.

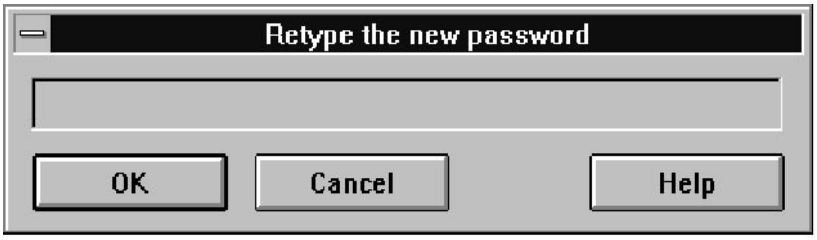


4 Click Owner, User, or Runtime. AionDS displays the Enter new ADS-password or Enter new AES-password window.

Caution: Always assign an owner password. Without an Owner password, the User password is not set.



5 Enter the new password and click OK. The Retype the new password window displays.



6 Re-enter the password. AionDS confirms your password and displays the Passwords dialog.

Caution: If you insert spaces while entering the password, the spaces become part of the password.

7 Click Set. If you don't want to set a password, click Cancel.

☑ *Tip:* You can enter a password for the owner, user, and runtime, and then click Set.

Organizing windows with the Window pull-down menu

Overview

You can control how AionDS places the Window editor and object list, Object editor, and Screen editor windows within the AionDS primary window with the Window pull-down menu. The Property pull-down menu controls the Property windows within an Object editor. The Screen pull-down menu controls the Windows within the Screen editor or Window editor.

When the Interface field in the Environment Settings window is set to character, AionDS/2 uses the Screen editor. When the Interface field in the Environment Settings window is set to graphical, AionDS uses the Window editor.

📖 *For more information:,* see Appendix A, "Building and Running Character-based Applications in AionDS," to learn how to use the Screen editor.

Select the Window menu to display the three choices:

Tile	displays the Windows editor and all open object list, object editor, and screen editor windows side-by-side and top-to-bottom within the AionDS primary window
Cascade	displays the Windows editor and all open object list, object editor, and screen editor windows one on top of another, with only the title bars showing
Close All	closes the Windows editor and all open object list, object editor, and screen editor windows. AionDS presents confirmation pop-ups for each window where you have made, but not saved, changes.

AionDS online help

Overview

There are many places to access online help in AionDS. The following table summarizes those places:

From	How?	Function
a window	press F1 or the Help push button	help window describing the window with hypertext links to help
a pull-down menu	when a pull-down menu option is highlighted, press F1	help window describing the purpose of the action
the Help icon in the button bar	click on the question mark help icon in the button bar	help window that lets you select from a list of common tasks
the Help pull-down	select Help from the menu bar	choose from the help options displayed in the menu

Chapter 4
Working with Object Lists

Introduction

This chapter shows you how to view and manipulate knowledge base objects through Object List windows. An **Object List** window displays collections of knowledge base objects selected according to your criteria. When you open a knowledge base, the first Object List window displayed is the knowledge base hierarchy window.

In this chapter

Object listing actions

Summary

Use the following two options in the Object pull-down menu to control creation of Object List windows:

Option	Function
List	opens an Object List window. Selecting List displays the Select Objects window. From the Select Objects window, you can select the type of objects to display in the Object List window
Collect	opens an Object List window containing objects selected from another Object List window or a property window that is a system-supplied list (such as a Used By property). Select the objects to put into another window. Selecting Collect displays the Collect Objects window

The controls in the Window pull-down menu also determine the layout of object lists, Object editors, and Window editor windows within the AionDS primary window.

Selecting objects for an Object List window

Overview

To view collections of related knowledge base objects, use an Object List window. From an Object List window you can organize your work, view relations between objects, and navigate through the knowledge base.

This section explains how to display knowledge base objects. Once you display a collection of objects, you can change the Object List window format and further narrow the list of objects.

What to do

➢ To open an Object List window, select Object.List.

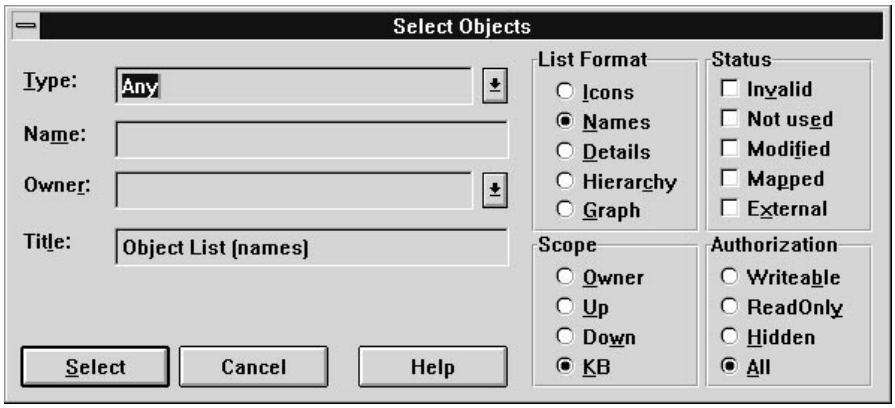
☯ *Alternative:* Select the list button from the button bar. The list button looks like the following:

AionDS displays the Select Objects window. From this window, indicate which objects you intend to list and the initial format.

Fields on the Select Objects window

Overview

Use the Select Objects window to specify characteristics of objects to display in an Object List window.



Summary

The following is a summary of window fields and controls:

Field/Control	Function
Type	drop-down list box indicating types of knowledge base objects to list
Name	text entry field to enter explicitly the name, or name with wild card, of objects to display in the object list
Owner	drop-down list box indicating the name of the context from which the objects can come. This combo box works in conjunction with the Scope radio buttons
Title	specifies text that displays in the title bar of the Object List window

Continued

Continued

Field/Control	Function
List Format	comprises five radio buttons controlling the format of the object list
Status	group of check boxes that lets you specify the status of objects to display in the Object List window
Scope	comprises three radio buttons, and works in conjunction with the Owner field. These buttons control where AionDS looks for objects to place in the object list.
Authorization	comprises four radio buttons that limit the objects that a user can view or change

Type field

Use the Type field to indicate the type of knowledge base objects to include in the object list.

Selecting Any

When you first open the Select Objects window, the default selection in the Type field is `Any`. `Any` indicates that the object list contains objects of any type that match the other specifications in this window. If you select `Any`, the object list is not constrained by the type of knowledge base object. To constrain the types of objects to include in the Object List window, select a type from the Types list box.

Selecting a single object type

Enter the full name of the object category, or enough of the first few letters (usually two) to indicate a unique object category. You can also select an object category from the drop-down list.

Selecting more than one object type

You can enter any type of object. Separate each object category with a space. The field scrolls horizontally to 255 characters, if you need more room. To select more than one type of object to list, enter the full name of the object or enter enough letters to indicate a unique object type. For example, use MD for method; ME for message; or IN for instance.

Name field

This is an optional field. Use the Name field to enter the name of the knowledge base object or objects to list. To list a number of objects with similar names, use the asterisk as a wild card. For example, enter:

```
st*
```

in the Name field to list all objects starting with the letters st.

To list objects ending with TRX, enter:

```
*TRX
```

🔔 *Reminder:* The Name field is not case-sensitive, so `TRX` is the same as `trx`. Leaving the Name field empty is equivalent to typing * in the field.

Owner field

Use the Owner field to indicate the name of the context that owns all objects placed in the Object List window. The owner you specify is used in conjunction with the Scope radio buttons to indicate how to constrain the list of objects.

Title field

Use the Title field to give the Object List window a title. With a useful name in the object list title bar, you can identify collections of objects quickly when you have more than one collection open.

The default title bar for an Object List window is the format of the Object List window. For example, the default title bar for an Object List window by icon is:

```
Object List (Icon)
```

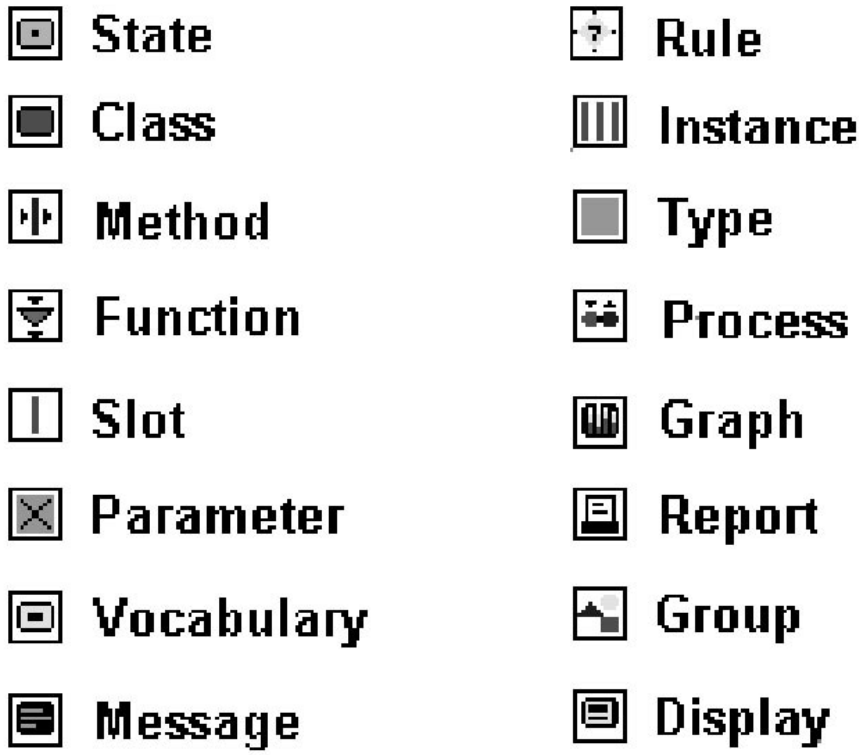
List Format radio buttons

The following five List Format radio buttons indicate the format of the Object List window:

- icons
- names
- details
- hierarchy
- graph

Objects as icons

In an Object List window, objects display as the following icons:



Status check boxes

You can specify the following characteristics of the objects to list. When you display an object list in a details view, AionDS displays the status of each object in the Flag column as follows:

Status	Flag	Description
Invalid	I	invalid knowledge base objects
Not Used	N	knowledge base objects with an empty Used by property; those not used by any other object in the knowledge base
Modified		no flag; unsaved, modified knowledge base objects
Mapped	M	mapped slots to a specific field in an external database
External	X	library objects (for example, objects in WinLib). AionDS adds library objects to the object list.

☑ *Tip:* If the List Object window does not display the flag column, expand the window.

Listing Authorization status

When you display an object list in detail view, AionDS displays any hidden or read-only authorization status.

Status	Flag	Description
Hidden	H	objects hidden from view which cannot be changed when the knowledge base is opened with a user password
Read-only	R	read-only objects which cannot be changed when the knowledge base is opened with a user password

Scope radio buttons

The Scope radio buttons define the scope AionDS uses when creating an Object List window.

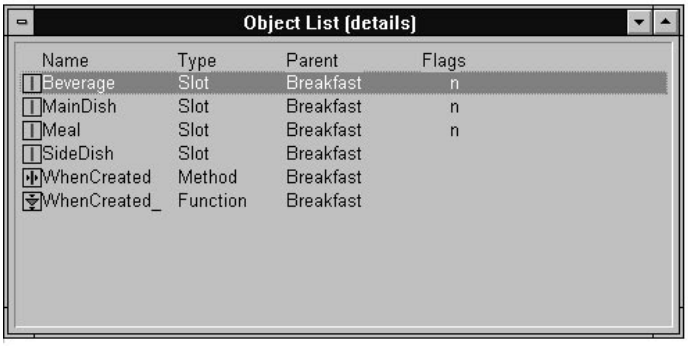
Scope	Function lists:
Owner	object name in the Owner field; any objects owned by this object display in the object list
Up	object name in the Owner field. Any objects owned by the following display in the object list, including objects in the Owner field and **ancestors** of the Owner field object. Ancestors are the owner, the owner's owner, and so on all the way up the hierarchy.
Down	object name in the Owner field. Any objects owned by the following display in the object list, including the object in the Owner field and the **descendants** of the Owner field object. Descendants are objects below the owner in the hierarchy.
KB	objects in the entire knowledge base, restricted by any constraints specified in the object selection window

Overview

To explore Owner and Scope, look at the GROCERY knowledge base. As shown in the partial hierarchy in the following example, GROCERY has a Menu class with three subclasses called Breakfast, Dinner, and Lunch.

Owner

Suppose you select Details from the List Format group box, Owner in the Scope group box, and All in the Authorization group box. You then select `Any` in the Type field and the Breakfast class in the Owner field. As shown in the following example, the object list only displays the knowledge base objects owned by the Breakfast class.

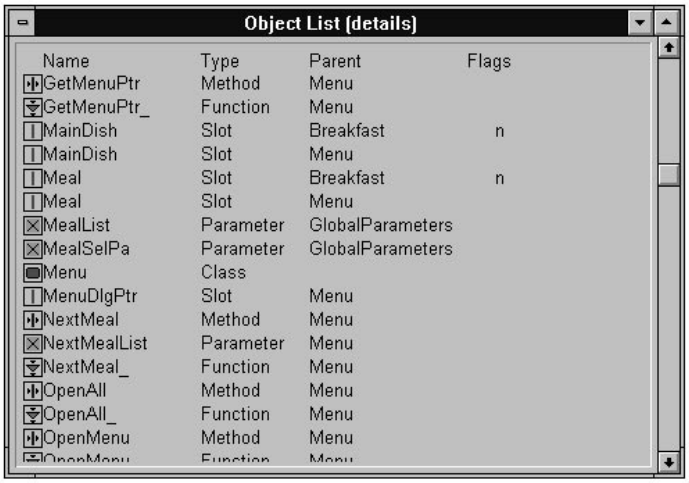


Up

Suppose you select Up in the Scope group box, and the Breakfast class is in the Owner field, and `Any` is in the Type field. The following objects display in the Object List window:

- objects owned by the Breakfast class
- the Breakfast class
- objects owned by classes above the Breakfast class
- classes above the Breakfast class

The following example shows the object list.



KB

If you select KB, objects from the entire knowledge base can display regardless of the contents of the Owner field. Any constraints specified in the Select Objects window affect the contents of the object list. Leaving the Owner field blank is equivalent to selecting KB. If you select KB, AionDS removes the contents of the Owner field from view.

Authorization radio buttons

The Authorization radio buttons display lists of selected objects with different levels of access:

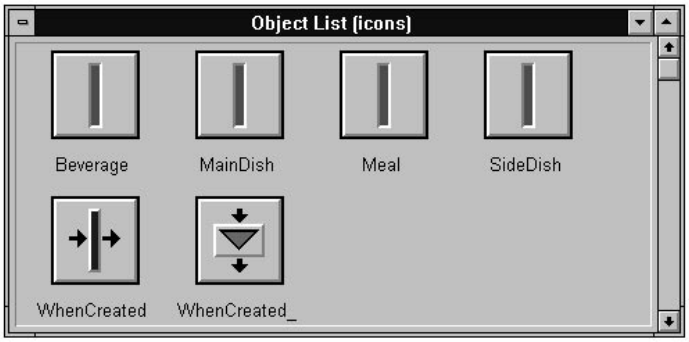
Authorization	Function lists:
Writeable	read/write objects
Read Only	read-only objects; cannot be changed
Hidden	objects hidden from view that cannot be changed; not available with the User password
All	all objects the user is authorized to view

📖 *For more information:* See Chapter 5, "Working with Objects."

Object listing by icon

Overview

If you select Icons in the Select Objects window, objects display as icons in an Object List window. Each object in the Object List window is represented by an icon; the object name displays below each icon.



To change the order of the icons, select a sort characteristic (by name, by type, or by owner) from the View pull-down menu.

Object listing by name

Overview

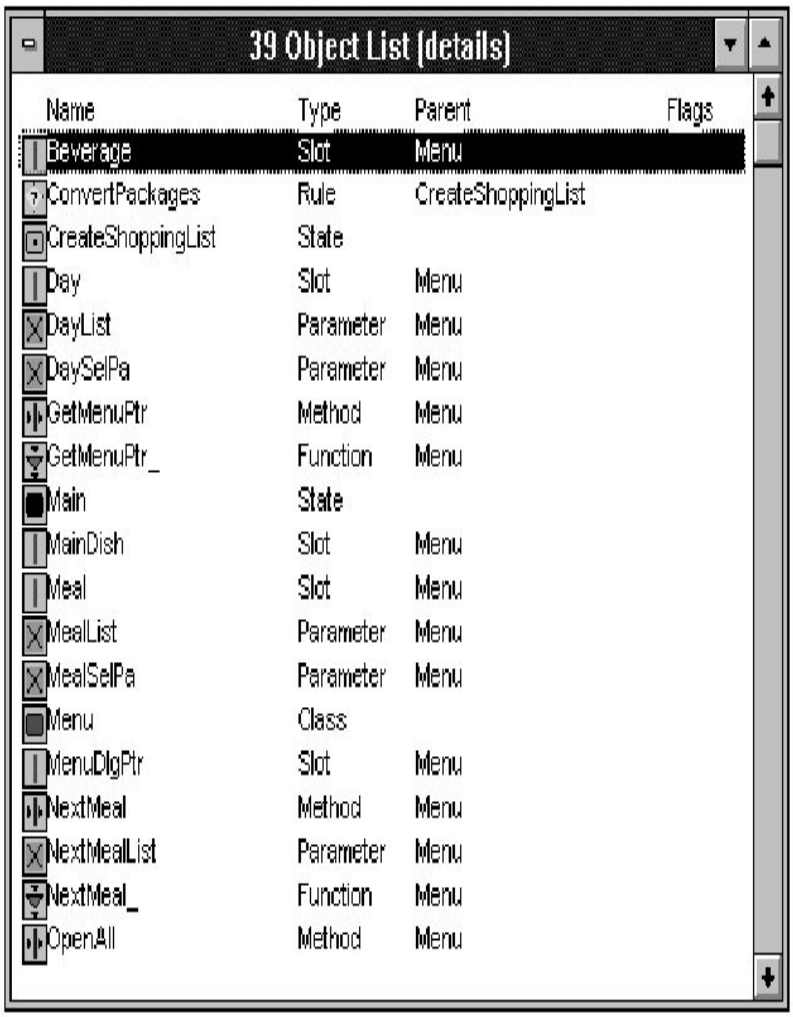
If you select Names on the Select Objects window, each object displays in an Object List window with a mini-icon and the object name.

To sort the Object List window, select a sort characteristic (by name, by type, or by parent) from the View pull-down menu.

Object listing by detail

Overview

If you select Details from the Select Object window, objects display in an Object List window as a list with a number of columns: name, type, parent, and flags.



To sort the Object List window, select a characteristic (by name, by type, or by parent) from the View pull-down menu.

📖 *For more information:* See “Status check boxes” on page 4-8 for a list of flags.

Object listing by hierarchy

Overview

This is the same object list format as the knowledge base hierarchy window displayed when you first open or create a knowledge base. If you select Hierarchy on the Select Object window, objects display in an Object List window by hierarchy.

If you select Hierarchy, the valid types in the Types field are Any, State, Class, and Vocabulary (in this case, Any is the same as entering State, Class, and Vocabulary in the Type field). AionDS displays a hierarchical outline of the knowledge base contexts, as in the following:

Contexts are states, classes, and vocabularies. These are the only objects that can own other objects. The outline shows the relationship between parent and child objects.

You can expand and collapse branches of the hierarchy in the object hierarchy window. You can display objects owned by a context.

📖 *For more information:* Select Help.Keys for a summary of Object List window keyboard options.

Expanding and collapsing branches

Select a branch or branches, then select View, followed by one of the following menu choices:

- Expand One Level (press PLUS on the number keypad)
- Expand Branch (press ASTERISK on the number keypad)
- Collapse branch (press MINUS on the number keypad)
- Expand All (press CTRL-ASTERISK on the number keypad)

☯ *Alternative:* To expand a branch that has children that are not showing, or collapse a branch that has children showing, click on that branch's mini-icon in the outline.

Listing owned objects

From an object hierarchy window, you can list all objects owned by a context. Select the context or contexts, then select View.Owned Objects. The objects owned by the context display in the hierarchy.

☯ *Alternative:* Point to the context icon or mini-icon and press CTRL-CLICK or CTRL-PLUS (on the number keypad).

Concealing owned objects

To conceal owned objects, select View.No Objects.

☯ *Alternative:* Point to the context icon or mini-icon and press CTRL-CLICK or CTRL-MINUS (on the number keypad).

Listing slots and methods

You can list all slots and methods in a class from an object hierarchy window. Select the class or classes, then select View.Slots/Methods. The class slots and methods display in the hierarchy.

☯ *Alternative:* Point to the class icon or mini-icon and press ALT-CLICK or ALT-PLUS (on the number keypad).

Concealing slots and methods

To conceal slots and methods, select View.No Objects.

☯ *Alternative:* To conceal slots and methods for a single class, press ALT-CLICK or ALT-MINUS (on the number keypad).

Listing instances

From an object hierarchy window, you can list all instances in a class. Select the class or classes, then select View.Instances. The class instances display in the hierarchy.

☯ *Alternative:* Point to the class icon or mini-icon and press SHIFT-CLICK or SHIFT-PLUS (on the number keypad).

Concealing instances

To conceal instances, select View.No Objects.

☯ *Alternative:* To conceal instances for a single class, press ALT-CLICK or SHIFT-MINUS (on the number keypad).

Listing parents

To list an object's parent in the Knowledge Base Hierarchy window, select View.Show Parent. The object's parents display in the hierarchy.

Listing a knowledge base hierarchy

To list a knowledge base object hierarchy displaying a library or class structure, select one of the following:

- View.Hierarchy.Library to view the library hierarchy
- View.Hierarchy.Class to view the class hierarchy

Moving around

Use the mouse to move the selection from object to object, or use the following keys to move the focus. The **focus** is the dotted box (or a raised

box in three-dimensional mode) that surrounds a selection before you highlight it by clicking or pressing the space bar.

Up and Down arrow	moves the focus from context to context up and down the knowledge base hierarchy
Right arrow	moves the focus into the set of owned objects or the set of class member objects. Once the focus is in a set, the right arrow moves through each object in the set
Left arrow	moves the focus up to an object's parent

Object listing by graph

Overview

The most powerful way to display an Object List window is by graph. Object listing by graph shows the relationships between knowledge base objects. If you select View.Graph, AionDS presents an Object List window by graph of the objects meeting the criteria.

Possible relationships

You can graph relationships between objects with the View pull-down menu. An Object List window must be active when you select the object or objects. The Object List window displays the following graphical relationships:

Property	Function
No relation	resets graph to show no relationship for selected objects
Uses	displays objects that the selected object uses. These are the same objects you can observe in an object's Uses property window. The selected object uses the graphed objects.
Used By	displays objects the selected object is used by. These are the same objects you can observe in an object's Used By property window. The graphed objects use the selected object.
Owns	selects a context, displaying the objects the selected context owns. The selected object owns the graphed objects.
Owned By	selects any object, and shows the context that owns the selected object

continued

continued

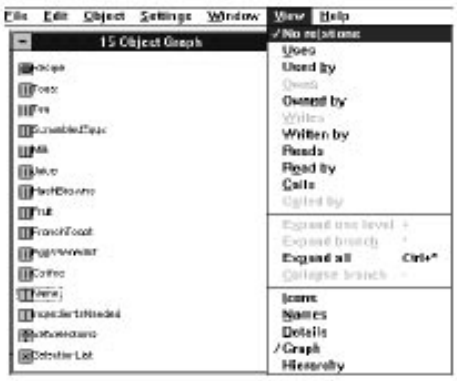
Property	Function
Writes	selects a function, rule, state, or process. When the selected object executes, the slots and parameters to which the selected object can set a value are displayed. The selected object writes or sets a value to the graphed objects.
Reads	selects an object. When the selected object executes, the Object List window displays the slots, parameters, and instances from which the selected object reads values. The selected object reads or uses the values of the graphed objects.
Read By	selects a slot, parameter, or instance. The Object List window displays the objects that read values in the selected object. The selected object is read by or used by the values of the graphed objects.
Calls	selects an object. The Object List window displays the objects that the selected object calls, including function calls, message passing, or agenda statements (message, report, graph, state). The selected object calls the graphed objects.
Called by	selects an object. The Object List window displays the objects that call the selected object.

☑ *Tip:* Object list by graph does not give an overview of an entire knowledge base, but views relationships between portions of the knowledge base. Keep the relationships as simple as possible. To analyze multiple relationships, use multiple object lists (by graph). In addition, you can use collapse nodes to simplify graphs.

What to do

1. Select the Graph list format from the Select Objects window. Objects display as a list of mini-icons in an object graph window.
2. Select the objects to graph.
3. Select a relationship to the graph from the View pull-down menu.

   When you select knowledge base objects in an object graph window, the View pull-down menu displays the possible relationships.



AionDS grays any relationships that you cannot graph for the selected objects.

Review

To review how to open and change an object list by graph window, open a window like the one in the previous example using the following steps:

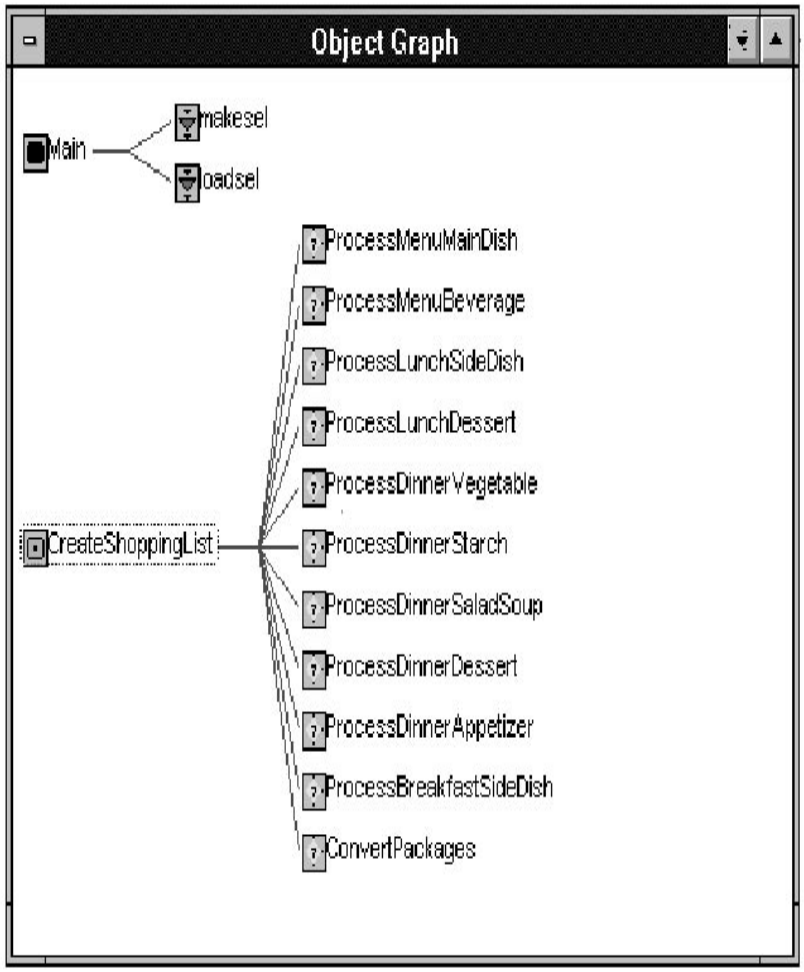
1. Open a knowledge base.
2. Select the List Objects button from the button bar.
3. In the Select Objects window, indicate the states in graph format.

4 Select the states to graph.

5 Select the relationship to the graph from the View pull-down menu.

Owns relationship

To look at the Owns relationship for the selected states, select View.Owns. In the following example, note how the object list displays the objects that each selected state owns.

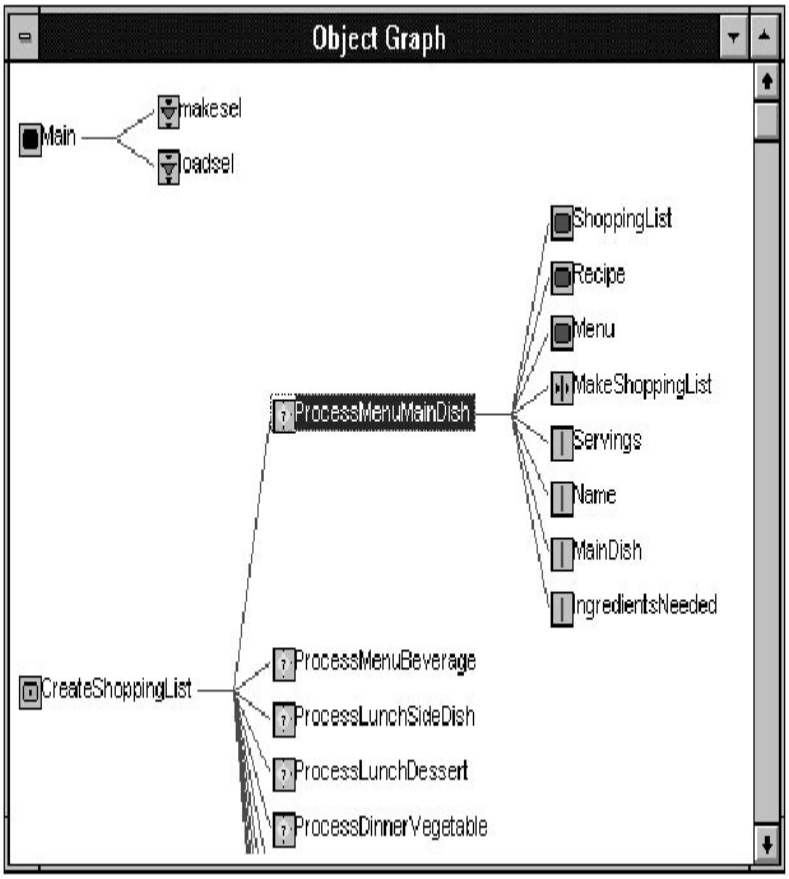


The Main state owns the makesel and loadsel functions. The CreateShoppingList class owns 11 rules.

Uses relationship

You discovered that the CreateShoppingList state owns a number of rules. How can you find out what objects a rule uses?

To see what objects the ProcessMenuMainDish rule uses, select the rule and then select View.Uses. As the following example shows, the Object List window displays all objects the selected rule uses.



Tips

- The number of relationships you explore at a time controls the complexity of the graph. To explore many relationships at once, collapse some branches of the graph to make the screen easier to read.
- The Object List window by graph can scroll horizontally and vertically.
- Sometimes there are no results when you select a graph. For example, if you select an *empty* class and choose View.Owns, no graph displays because the class does not own anything.

Changing the format of an Object List window

Overview

You can change the format of the objects displayed in an Object List window. When any Object List window is open, you can change the window to any of the other four formats.

What to do

➢ To view objects in a different format, select a new format from the View pull-down menu.

☯ *Alternative:* You can also select another format for an Object List window when you collect objects.

Collecting objects into another Object List window

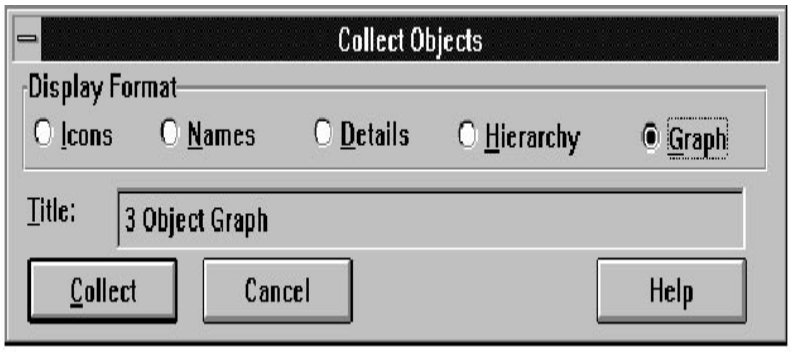
Overview

To view specific types of objects from an existing Object List window, collect the objects into another Object List window.

For example, if you open an Object List window for all states in a knowledge base, then add a new state, the Object List window reflects the new states. If you collect STATE1, STATE2, and STATE3 into an Object List window, the contents of this window do not change (unless you delete a collected object from the knowledge base).

What to do

1. To select objects to collect from an existing Object List window, press SHIFT-CLICK on the object in the collected Object List window.
2. Select Object.Collect. AionDS displays the Collect Objects window.



3. Specify the format of your new Object List window by selecting a radio button from the Display Format group.
4. Enter the title of the collected Object List window.
5. Click Collect.

Chapter 5
Working with Objects

Introduction

This chapter discusses the following topics:

- Working with objects: how to create, open, import, export, cut, copy, paste, move, search and replace, set authorization for, and print objects.
- Working with object properties: how to open, cut, copy, paste, move, search for, and replace text in properties.

In this chapter

Continued

Continued

Working with included library objects

Overview

You modify objects in a library the same way you modify objects in a knowledge base—you open the library as a knowledge base. Included library objects can own knowledge base objects, so when you modify objects in a library, you might also need to modify the knowledge bases in which the library is included.

Creating objects owned by library objects

You can create instances and subclasses of included library classes, and you can create substates of included library states. These objects are placed in the knowledge base, not in the included library.

Caution: When the parent of a knowledge base object is an included library object that has been deleted or renamed, AionDS deletes the knowledge base object because AionDS cannot find the parent of the knowledge base object.

Library object restrictions

The following restrictions apply to libraries:

- A library can contain up to 65,535 objects.
- You cannot change included library objects from within the knowledge base.

Creating an object

Overview

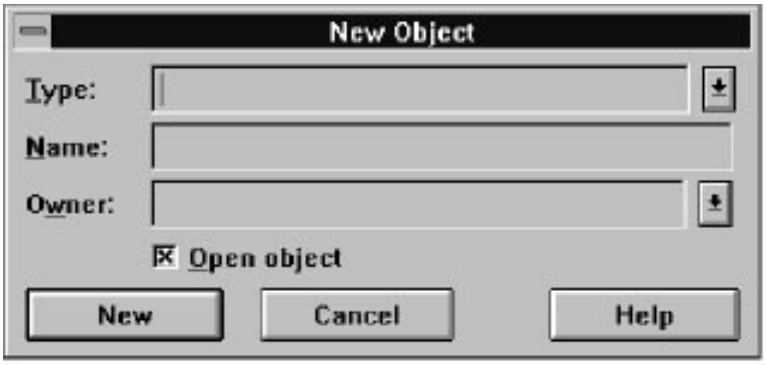
To create objects, you use object editors to define and develop objects. You can also create objects by copying existing objects.

📖 *For more information:* See the "Copying objects" section.

What to do

To create a knowledge base object, use the New Object window.

1 Select Object.New. The New Object window displays.



2 Specify the object type, name, and owner in this window.

   🔔 *Reminder:* Included library objects can own instances, substates, and subclasses.

3 Specify whether to open the object editor by checking or unchecking Open object.

4 Click New.

Type field

Use the Type field to indicate the type of knowledge base object to create. You can enter the full text of the object type, select a type from the drop-down list, or enter the two-letter abbreviation of the object type.

📖 *For more information:* See the *General Reference* and online help for a list of object type abbreviations.

Name field

Use the Name field to enter the name of the knowledge base object to create. The object name can be up to 24 characters long, starting with an alphabetic character and containing only alphanumeric characters and underscores. No other characters (such as spaces, dashes, or ampersands) are allowed.

Owner field

Use the Owner field to indicate the context to own the new object. The default owner is the context currently selected when you select Object.New. The default owner displays in the Owner field.

If no context is selected, or to change the default owner, select from the pull-down menu. You can also enter the entire context name in the Owner field.

Open object check box

After AionDS creates the object, check Open object to open an object editor for the new object. If you do not check Open object, AionDS creates the object, but does not open the editor for that object.

The Open object selection stays the same until changed. For example, if you uncheck Open object when you create an object, the next time the New Object window displays, Open object remains unchecked.

New push button

After entering all the information in the New Object window, click New. If you checked Open object, AionDS creates the object and displays the object editor.

Saving an object

What to do

➢ To check the syntax of an object and add its changes to the knowledge base, select Object.Save.

☯ *Alternative:* To save an object, press F7 while its editor is active.

Result

AionDS checks the syntax of the object. If there are no syntax errors:

- AionDS saves the object
- the following smile icon displays:

- AionDS deletes the asterisk next to the object name in the title bar

If there is a syntax error:

- AionDS does not save the object
- a frown icon displays, pointing to the syntax error:

- an edit error message displays in the message bar
- an exclamation point displays next to the object name in the object editor title bar
- an exclamation point displays in front of the object name in the object hierarchy window to indicate the object is invalid

Saving an object with errors

1. You can save an object with syntax errors by double-clicking on the object editor's system menu. AionDS displays an Errors confirmation dialog.
2. To close and save the object, click Continue.

Result: AionDS closes and saves the object with the syntax errors.

Caution: You are not prompted to resolve these syntax errors before saving the knowledge base.

Changing and deleting objects in a library

What to do

1. Open the library as a knowledge base.
2. Select the object from the knowledge base.
3. Change or remove the library object.
4. Add the library back into the knowledge base.

Caution: Changing or deleting library objects with owned or child objects in the knowledge base may produce unexpected results in the knowledge bases in which the library object is included.

Effects

Caution: If you delete or rename a library object, the following changes are made when the library is updated to a knowledge base:

- The library object's owned objects and child objects in the knowledge base are deleted.
- Any KDL, TDL, and TSL references to the included library object and the object's owned objects are invalidated.

When the parent of a knowledge base object is an included library object that has been deleted or renamed, AionDS deletes the knowledge base object because AionDS cannot find the parent of the knowledge base object.

Opening object editors

Overview

This section discusses the different methods you can use to open an object editor for an existing object. If you don't know the exact name, type, and owner of the object to open, use the Open Object window to list the objects that match what you know about the object.

Opening an object editor

1 Select an object in an Object List window.

2 Select Object.Open.

Result: The object's editor displays.

☯ *Alternatives:* To open an object editor from an Object List window, you can also:

- double-click an object icon or the name next to the icon
- select an object and press ENTER
- select an object and click the following edit icon from the button bar:

☯ *Alternative:* To open an object editor from a text window, select an object name and double-click on it, or press F3. AionDS opens that object's editor. If the object name is not unique, a list box displays. A **text window** is an object property displayed in Text Substitution Language (TSL) in which you can enter text.

Opening more than one object editor

You can open editors for more than one object by doing the following:

1. Select the objects to open from an Object List window. Hold down SHIFT and click each object.
2. Select Object.Open.

Result: AionDS opens the editors for all selected objects.

☯ *Alternatives:* To open more than one Object editor, you can select the objects to open and do one of the following:

- press ENTER
- press F3
- click the edit icon from the button bar

The Open Object window

If you don't know the exact name, type, and owner, use the Open Object window to list the objects that match what you know about the object. To display the desired object editor:

1. Do not select an object.
2. Select Object.Open.

   ☯ *Alternative:* Press ENTER or F3, or click the edit icon on the button bar.

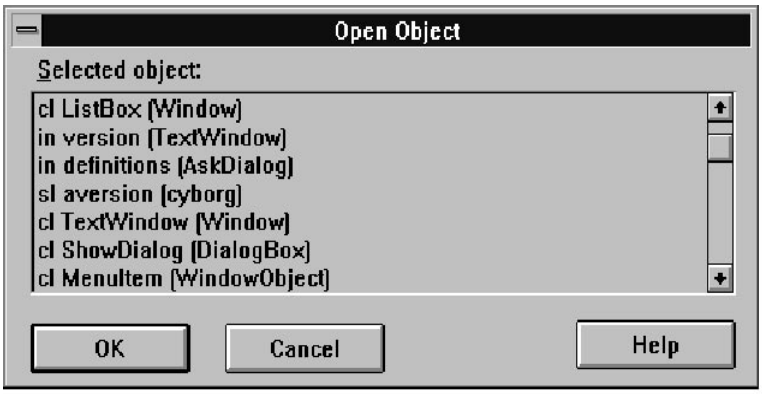
   Result: The Open Object window displays.

3 Specify the type, name, and owner of the object to open in the Type, Name, and Owner fields.

Field	Description
Type	selects the object type. If you do not know the object type, select Any. Any selects all object types matching the criteria in the Name and Owner fields.
Name	enters the object name. If you only know the first few letters of the object's name, substitute the wild card symbol (*) for the remaining letters.
Owner	selects the object's parent. If you do not know the object's parent, leave this field blank.

4 Click Open.

5 If you do not specify the object's type, name, or owner, another dialog displays with a list of objects that match the specified criteria.



6 Select the desired object from the Selected object field.

Result: The specified Object editor displays.

Going to another object

Overview

You may need to edit an object related to the object you are currently editing. Use Jump to open an editor quickly for a related object. Jumping to another Object editor does not close the editor from which you jumped.

The Jump feature lets you view a list of objects related to the active object and edit one of them.

Related objects

You can select from the following relationships:

Properties	The To list box lists:
Uses	objects to which the From object explicitly refers
Used By	objects in which the From object is referred to explicitly
Owns	objects which the From object owns. This option is available only when a context is in the From field.
Owned By	the context which owns the From object.

Continued

Continued

Properties	The To list box lists:
Up	If the From object is not a state or class: objects of the same name and type owned by any context above the From object in the hierarchy. If the From object is a state or class: objects of the same type above the From object in the knowledge base hierarchy.
Down	If the From object is not a state or class: objects of the same name and type owned by any context below the From object in the hierarchy. If the From object is a state or class: objects of the same type below the From object in the knowledge base hierarchy.

What to do

1 From an active Object editor, select Object.Jump. The Jump window displays:

2 Indicate the relationship using the Relation radio buttons.

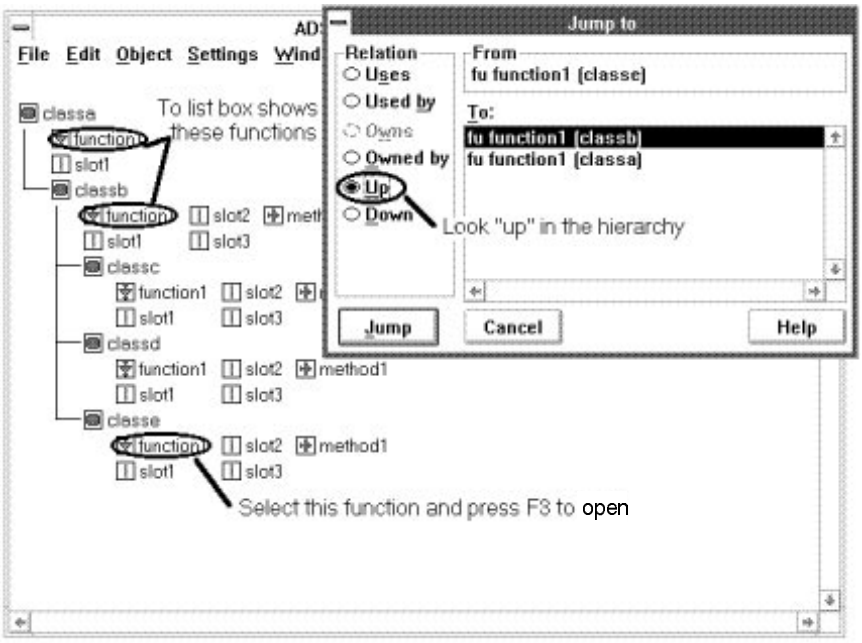
These buttons define the relationship between the active Object editor shown in the From field (the object where you opened Jump) and the objects that appear in the To list box.

3 To select an object to edit in the To list box, double-click an entry in the list box, or select an entry, then select the Jump push button.

From the Jump window you can select the object in the To list box to which you want to jump.

Example

The following example shows objects that appear in the To list box if you open Jump from a function at the bottom of a class hierarchy.



Closing and saving an object

What to do

To check the syntax of an object, add its changes to the knowledge base, close the object's editor, and select Object.Close (or press F8). AionDS saves the object in the open knowledge base and closes the object editor. The knowledge base is not written to disk. To save the knowledge base to disk, select File.Save.

What happens

If there are no syntax errors, AionDS displays a smile icon.

If there is a syntax error, AionDS displays a frown icon that points to the problem. An edit error message appears in the message bar. AionDS also displays an exclamation point next to the name of the object.

Closing an object without saving

What to do

➢ To close an Object editor window without saving any changes, select Close on the system menu icon, located in the top left-hand corner of the property window.

☯ *Alternatives:*

- double-click on the system menu icon
- select Window.Close all
- press CTRL-F4

What happens

AionDS displays a confirmation window that prompts you to save the object's changes. You can save or discard the changes, or cancel the confirmation window and return to the Object editor.

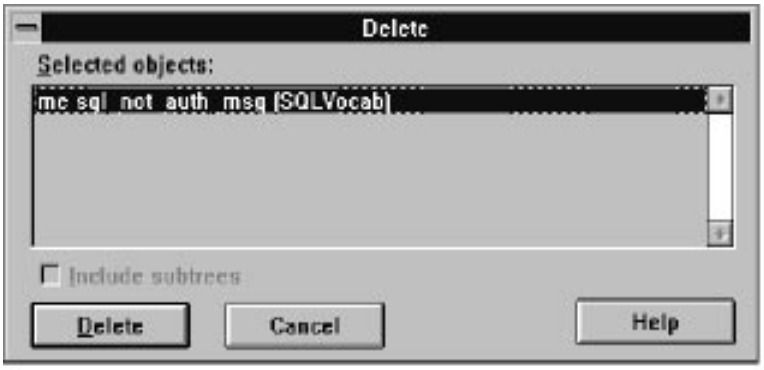
Deleting objects

What to do

To delete knowledge base objects, use the Delete window.

1 Select the object or objects.

2 Select Edit.Delete object. The Delete window displays:



3 Click Delete to confirm the removal of the object from the knowledge base. If the object is a context and you want to remove the other objects below it in the hierarchy, check Include subtrees .

To delete a graphical instance, use the Window editor.

✋ *Caution:* If you delete or rename a library object, the following changes are made when the library is updated to a knowledge base:

- The library object's owned objects and child objects in the knowledge base are deleted.
- Any KDL, TDL, and TSL references to the included library object and the object's owned objects are invalidated.

📖 *For more information:* See Chapter 6, "Working with the Window Editor," to learn how to delete graphical instances.

Cutting to the clipboard

To place a copy of selected objects on the clipboard, select Edit.Cut object, or click the scissors button in the button bar. Using Cut is safer than using Delete because you can Paste the object back.

Cutting and pasting objects

What to do

To cut and paste objects:

1. Using the mouse or keyboard, select the object or objects to cut or copy.

   If you select icons or mini-icons from any Object List window, you can extend the selection across multiple objects.

   If you select an object by highlighting its Object editor window, you can select only one object.

   ✋ *Caution:* When you make changes to objects in a knowledge base that has been included as a library in other knowledge bases, you might need to modify the knowledge bases in which the library is included to accommodate your changes.

2. **To cut an object,** select Edit.Cut object. Cut object removes the selected object or objects to the clipboard.

   ☯ *Alternative:* Select the cut button from the button bar.

   AionDS displays the Cut window. Use this window to confirm the selected object to cut. Use the list box in this window to deselect an object.

   To copy an object, select Edit.Copy object. This copies the selected object or objects into the clipboard.

   AionDS displays the Copy window. Use this window to confirm the selected objects to copy. Use the list box in this window to deselect an object.

   ☑ *Tip:* Depending on the current selection, the cut, copy, and paste actions in the Edit pull-down menu change. If you select a knowledge base object, the Edit menu has a Cut object choice. If you select text, the Edit menu has a Cut text, Copy text, or Delete text choice.

3. If the knowledge base object or objects are contexts, the Include subtrees check box is available. Check Include subtrees to place all contexts below the selected context in the clipboard.

4. Select an object to be the parent of the knowledge base object.

5 Select Edit.Paste.

☯ *Alternative:* Select the paste button (the glue bottle icon) from the button bar.

AionDS displays the Paste window.

6 If the selected object is a valid owner for the objects in the clipboard, the Target Owner field contains the object's name. If the selected owner is not a valid owner for the objects in the clipboard, the Target Owner field is blank. Enter a valid parent from the drop-down list in the Target Owner field.

If the name is already in use in that context, AionDS prompts you for a new object name.

7 If you check Open object list, AionDS displays a Collect Objects window with the target object in the Owner field. This lets you open a new Object List window for the pasted objects.

🕮 *For more information:* See Chapter 4, "Working with Object Lists," for a description of Object List windows and the Select Objects window.

Cutting and pasting between knowledge bases

Overview

You can cut and paste knowledge bases, attached library text, or objects between knowledge bases. The contents of the clipboard remain after you close a knowledge base. You can cut text, objects, or windows from one knowledge base, close the knowledge base and open another, then copy clipboard contents to the open knowledge base.

✋ *Caution:* Library objects can own other objects in a knowledge base. Cutting library objects may produce unexpected results in the knowledge bases in which the library objects are included. From within a knowledge base, you cannot cut included library objects.

Pasting the same information into several knowledge bases

As long as you do not copy over current information (text, objects, or windows) in the clipboard with new information you can continue to paste the same information into several knowledge bases by using Edit.Cut or Edit.Copy.

📖 *For more information:* See Chapter 3, "Working at the Knowledge Base Level," for information on how to cut and paste attached library objects between knowledge bases.

When is information in the clipboard discarded?

The clipboard continues to hold AionDS objects and windows after you close a knowledge base, but not after you close AionDS. The clipboard continues to hold text even after you close AionDS.

Directly manipulating objects

Overview

Using an object list and the mouse, you can move knowledge base or library objects from one place to another, or make identical copies of objects through **direct manipulation**. Moving objects deletes them from their original context and moves them to the new context. **Cloning** objects copies the identical objects to another context in the knowledge base. In other words, moving is cut and paste in one step, while cloning is copy and paste in one step.

To move graphical instances, use the Window editor.

What to do

To move or clone knowledge base objects with direct manipulation:

1. Select one or more knowledge base objects from any Object List window.
2. **To move**, hold down the right mouse button.

   Caution: Library objects can own other objects in a knowledge base. Moving library objects may produce unexpected results in the knowledge bases in which the library objects are included. From within a knowledge base, you cannot move included library objects.

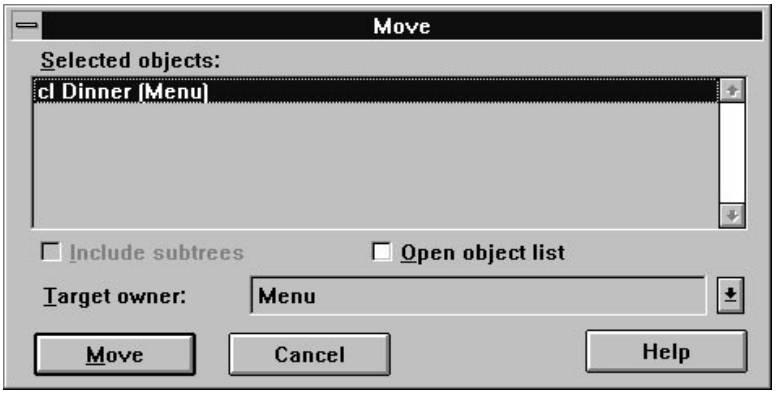
   To clone, press CTRL and the right mouse button.

If you select one object to move or clone, the object's icon displays in place of the mouse pointer.

If you select two or more objects to move or clone, the selected objects display in the Move window or Clone window.

3 Move the icon to the object. A raised bar indicates the new owner.

4 A Move window or Clone window displays.



Use this window to confirm the move or clone procedure for the selected objects. Use the list box in this window to select and deselect objects.

5 If the knowledge base object or objects are contexts, Include subtrees is available. To place the contexts below the selected context in the clipboard, check Include subtrees.

6 If you check Open object list, AionDS displays a Select Objects window with the target object in the Owner field. You can immediately open a new Object List window in the context where the object was pasted or cloned.

📖 *For more information:* See Chapter 4, "Working with Object Lists," for a description of Object List windows and the Select Objects window.

How you can tell an object has changed

Overview

After an object has changed, AionDS displays an asterisk next to the object name in the title bar of the object editor window. If you minimize the Object editor, the asterisk appears next to the object icon's name. AionDS deletes the asterisk when you file or save the object.

Which properties have changed?

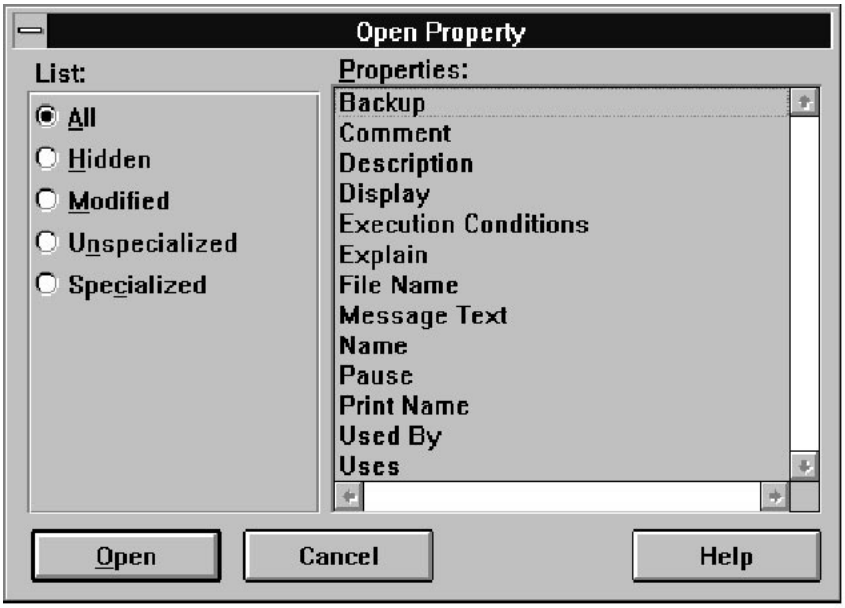
After changing a property, AionDS displays an asterisk next to the property name in the title bar of the Property editor window. AionDS deletes the asterisk when you save or close the object. All properties are not always displayed.

What to do

To check all properties of an object for changes since the last save or close, use the Open Property window.

1 Open the editor for the object.

2 Select Property.Open. The Open Property window displays.



3 Select the Modified radio button. The Properties list box displays the changed properties in the object.

Is an object invalid?

An exclamation point displays next to an invalid object's name in:

- the title bar of the object's editor
- the object's minimized icon

In the Object List window, the invalid status displays in the Flags column as an `in`.

Specializing class slots and methods

Overview

Child classes inherit the slots and methods of their parent class. A slot or method copied from a parent class is an **inherited** slot or method. A class can use, but cannot modify, an inherited slot or method. To modify these objects, specialize an inherited slot or method.

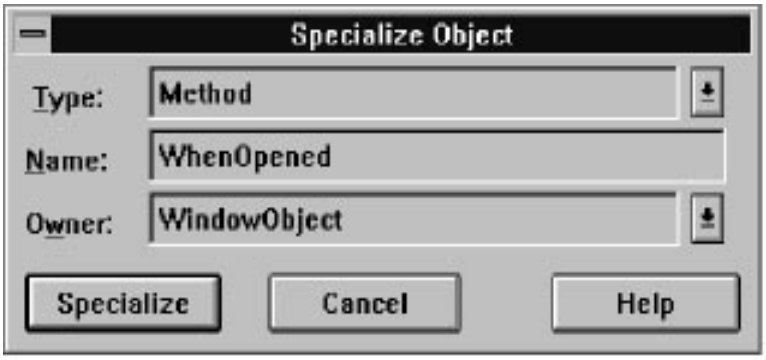
☑ *Tip:* You can specialize a library method, but you cannot specialize a library slot.

What to do

Use the Specialize Object window to specify the slot or method name, the type of object to specialize, and the name of the class that owns the slot or method.

1 Select the class.

2 Select Object.Specialize. The Specialize Object window displays.



3 Specify the object type, a method or slot, in the Type field.

4 Specify the name of the object in the Name field.

☯ *Alternative:*

1 Select a slot or method before selecting Specialize. The object name displays in the Specialize Object window.

2 Specify the class in the Owner field, then perform step 5.

5 Click Specialize. The slot or method editor displays. From one of these editors, you can specialize one or more properties of the slot or method.

📖 *For more information:* See pages 39-40 to learn how to specialize and unspecialize properties.

Unspecializing class slots and methods

Overview

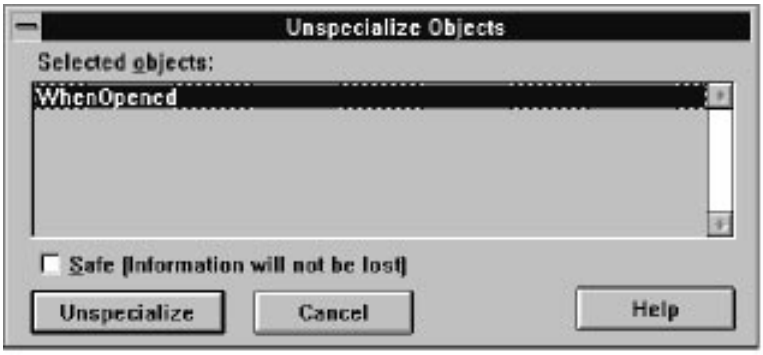
You can unspecialize an object you no longer need. The object returns to its original state unmodified. You cannot specialize or unspecialize a slot owned by a library.

What to do

Use the Unspecialize Object window to unspecialize slots or methods.

☑ *Tip:* You can unspecialize more than one object at a time by selecting multiple objects in the List Object window.

1 Select the specialized slots and methods. The Unspecialize Objects window displays.



2 Highlight the specialized slots and methods by clicking the slot or method name.

3. To unspecialize only the slots or methods that are identical to the slots and methods in the parent class, check Safe. The changed slots and methods containing modified properties from the slots and methods in the parent class do not change. The Safe option protects slots and methods:
   - with specialized properties
   - bound to a local object
   - that are invalid
4. Click Unspecialize.

Result: AionDS deletes the highlighted slots and methods in the Selected Objects list box.

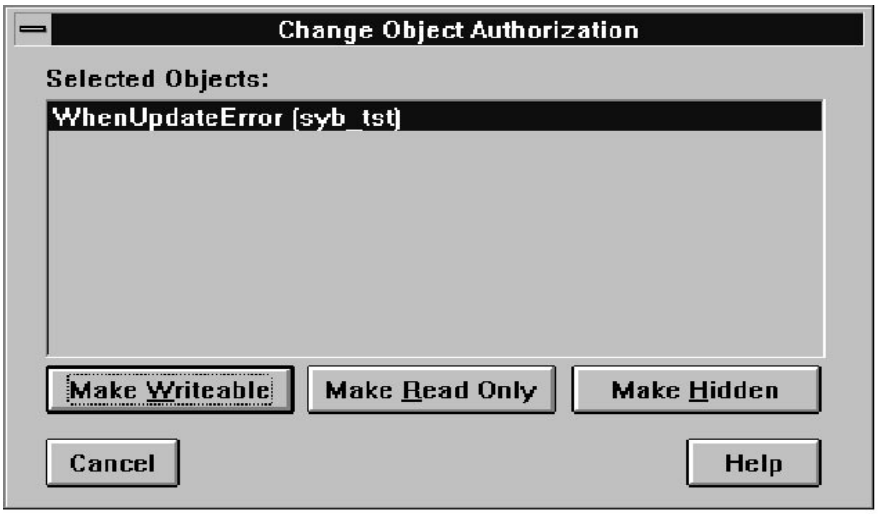
Changing object authorization

Overview

You can protect objects from change or you can hide information. If you use an owner password to open a knowledge base, you can change an object's authorization. You can make an object writeable, read-only, or hidden.

What to do

1. From a knowledge base hierarchy window or an Object List window, select one or more objects. To select multiple objects, press SHIFT and click each object.
2. Select Object.Change Authorization. The Change Object Authorization window displays.



3 Click one of the following Authorization push buttons:

Make Writeable	object is write/read-only
Make Read Only	object is read-only
Make Hidden	object is hidden

4 Confirm the authorization by displaying the objects in a detailed object list.

📖 *For more information:* See Chapter 4, "Working with Object Lists," to learn how to view objects.

Editing an object

Overview

To edit an object, enter information into the object's properties. This section explains how to select and work with AionDS properties.

From within a knowledge base, you cannot change included library objects.

Summary

The Property pull-down menu has the following commands:

Command	Description
Open	selects particular properties to open within an object editor window. AionDS displays the Object Properties window.
Specialize/Unspecialize	specializes an inherited property. If you specialize a property, you can change the contents. If you do not specialize a property, it remains locked. You must specialize a property to make changes.
Choices	displays a pop-up window with a list of valid options for the selected property.
Tile	arranges all open property windows in the active object editor to appear side-by-side and top-to-bottom within the object editor window.
Cascade	layers all open property windows in the active object editor one on top of another, with only the title bar showing.
Close All	closes all open property windows in the active object editor. Close All does not save or discard the contents of any property window, it just hides the property window.

Properties that appear by default

Overview

An object editor appears as a window within the AionDS primary window. In turn, an object editor can own a number of property windows. Each window manipulates a particular property or properties of a knowledge base object.

When open, each object editor displays properties specified in the defaults (until you change the defaults). In most editors, one to three property windows automatically open by default. You can change the default property position and window size.

📖 *For more information:* The procedure for changing these defaults is described in Chapter 10, “Customizing AionDS.”

Opening property windows

Overview

You cannot make changes to a property until its property window is the current window. By default, AionDS opens commonly used properties for each knowledge base object. The rest of the properties are hidden.

What to do

To open hidden property windows:

1 Select Property.Open.

☯ *Alternatives:*

- click open
- press F4

2 In the Open Property window, constrain the list of properties by selecting a List radio button. By selecting a different radio button, you can display a different set of properties in the Properties list box.

Radio button	Display
All	lists all properties available in the selected object, whether their windows are open or closed
Hidden	lists all properties available in the selected object, but does not display them at the bottom of the Properties pull-down menu
Modified	lists all properties modified since the last time you filed the knowledge base object
Unspecialized	lists all properties inherited from a parent and not yet specialized
Specialized	lists all properties inherited from a parent class and specialized

📖 *For more information:* See "Specializing and unspecializing object properties" on page 5-42.

3 From the Properties list box, select the properties to open. You can select more than one.

4 Click Open.

What happens

AionDS opens windows for the selected properties. You can change information in the object properties. To change inherited properties, specialize them.

📖 *For more information:* See "Specializing and unspecializing object properties" on page 5-42.

Some property windows represent multiple properties

Some property windows consist of operating system controls representing more than one property. The Property List window displays the names of individual properties. In some cases, more than one entry in the Properties list box opens the same property window.

Property window names display on the Property pull-down menu

The dynamic list at the bottom of the Property pull-down menu shows all open properties, and the selected property is checked.

Closing property windows

What to do

To close a property window, double-click on the system menu icon of the property window. ☯ *Alternative:* Press SHIFT-F4.

To close all open property windows in the active object editor, select Property.Close All.

What happens

When you close a property window, changes are not saved or discarded. You have hidden the property window. If you reopen that property window, the changes you made remain current.

You cannot save individual properties. To save changes to object properties, save the object itself.

Making open property windows active

What to do

You can change information only in an active property window. To make a property window active, click within its window. Sometimes an object property completely covers another and you cannot click on the covered property.

In this case, make the property window active by:

- cycling through open property windows of the active object editor by pressing SHIFT-F6
- selecting an open property from the window list at the bottom of the Property pull-down menu

Setting scroll bars in property windows

What to do

Property windows that are text-editing windows have scroll bars. To turn them on and off, select or deselect Horiz. scroll bar or Vert. scroll bar from the property's system menu.

Specializing and unspecializing object properties

What to do

To specialize an unspecialized property in a Class, Slot, or Method:

- activate the unspecialized property of an object editor
- select Property.Specialize

To unspecialize a specialized property:

- activate the specialized property of an object editor
- select Property.Unspecialize

📖 *For more information:* See pages 26-27 for information on specializing and unspecializing class slots and methods.

Specializing inherited properties

If a property is inherited from a class above it in the knowledge base hierarchy, the default color of the text in the property window is blue (or whatever color you set for this type of text in the operating system). The displayed default color means you cannot edit this property until you specialize it.

To determine if a property is specialized, use the Open Property window in the Specialized list of properties.

After you select Property.Specialize, the text in the property window becomes black (or whatever color you set for this type of text in the operating system). The object is now specialized, and you can change the contents of the property windows.

Unspecialize inherited values

To return an object to its original state, select Unspecialize from the Open Property window. AionDS removes the specialized input and returns the object to its original, unspecialized state.

You can toggle back and forth between specialized and unspecialized property contents by repeatedly selecting Property.Specialize/Unspecialize.

🔔 *Reminder:* An asterisk displays after the property name. The change has not been saved; you must save the objects to write the changes to the knowledge base.

List specialized properties

In the Open Property window, there are two radio buttons: Specialized and Unspecialized. Select a radio button to display the specialized or unspecialized properties of the current object.

Determining which objects in a class are specialized

To determine which objects in a Class are specialized, open the Slot or Method properties from the Class editor. If the method or slot is specialized, the current class is the designated owner.

Reorganizing the property windows in an object editor

Overview

You can reorganize the open property windows in an object editor window. Select one of the following options from the Property pull-down menu:

Command	Action causes all open property windows to:
Tile	appear side-by-side and top-to-bottom within the object editor window
Cascade	be layered one on top of another, with only the title bar showing
Close All	close; does not save or discard the contents of any property window

Do not confuse these options with similar options in the Window pull-down menu.

In the Property pull-down menu, these options act on property windows within an object editor.

In the Window pull-down menu, these options act on object editor windows, Object List windows, and screen editor windows within the AionDS primary window.

Cutting and pasting text

What to do

To cut and paste text:

1. Select the text to cut or copy with the mouse or keyboard.

   ✋ *Caution:* Library objects can own other objects in a knowledge base. Cutting and pasting library objects may produce unexpected results in the knowledge bases in which the library objects are included. From within a knowledge base, you cannot delete or change included library objects.

2. **To remove** the selected text to the clipboard, select Edit.Cut text.

   To copy the selected text to the clipboard, select Edit.Copy.

   ☑ *Tip:* Depending on the current selection, the cut, copy, and paste actions in the Edit pull-down menu change. If you select text, the Edit menu has a text choice. If you select a knowledge base object, the Edit menu has a Cut object choice.

3. Place the cursor where you want to paste the clipboard text.

4. Select Edit.Paste.

   ☯ *Alternative:*

   - press SHIFT-INS
   - select the paste button from the button bar

☯ *Alternative:* Use the cut and paste buttons on the button bar instead of selecting Cut text and Paste from the menu bar.

Cutting and pasting text between applications

Overview

AionDS shares the clipboard with other applications. AionDS can place text in the clipboard, while any other application that supports the clipboard can paste the information into its application files. AionDS can paste text placed on the clipboard by another application into each knowledge base.

Caution: Library objects can own other objects in a knowledge base. Cutting and pasting library objects may produce unexpected results in the knowledge bases in which the library objects are included. From within a knowledge base, you cannot delete or change included library objects.

Example

The following gives some examples of how you can take advantage of the clipboard function:

- using a word processor to write reports, messages, and text for other knowledge base objects while using the clipboard to copy that text into knowledge base objects
- copying help topics that contain syntax examples to the clipboard while pasting the examples into knowledge base objects. (Select Services.Copy from any help window to copy the current help topic into the clipboard)

Searching for and replacing text

Overview

You can search and replace any text residing in the active property, all selected objects, or the entire knowledge base. To search external library objects (for example, objects in WINLIB), check Include External. Otherwise, only objects that you created are searched.

Caution: Library objects can own other objects in a knowledge base. Replacing text in library objects may produce unexpected results in the knowledge bases in which the library objects are included. From within a knowledge base, you cannot replace text in included library objects.

Search for text

To search for text:

1. Select the property or object in which to search for the text.
2. Select Edit.Search.
3. Enter the text for which to search.
4. Select the location to search in the Scope field:

Scope Field	Searches
Active Prop	only the active property
Object(s)	only selected objects
KB	the entire knowledge base

5. For a list of objects in which the text was found, check List.
6. To search external library objects, check Include External. Otherwise, AionDS only searches objects you created.
7. Click OK. AionDS highlights the found object, and the Continue Search dialog box displays.

8 Click one of the following push buttons:

OK	to continue searching for the next occurrence of the text
Cancel	to end the search

Replacing text

To replace text:

1 Select the property or objects to search.

2 Select Edit.Replace.

3 Enter the text to replace and the new text.

4 Select the location to search in the Scope field:

Scope field	Searches
Active Prop	only the active property
Object(s)	only selected objects
KB	the entire knowledge base

5 For a list of objects in which the text was found, check List.

6 Click OK. Text to be replaced is highlighted and the Confirm Replace dialog box displays.

7 Click one of the following push buttons:

Push button	Function
Replace	to replace the highlighted text with the new text and continue to the next occurrence of the text
Ignore	to continue to the next occurrence of the text without replacing the highlighted text
Cancel	to end the search and replace, click Cancel

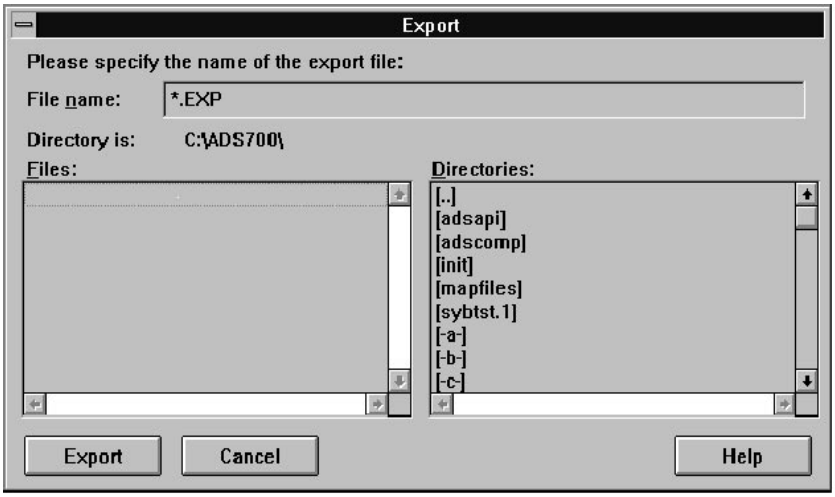
Exporting objects to external files

What to do

To export an object to an external file, use the Export window.

1. Select an object to export.
2. Select Object.Export. The Export window displays.

3 Click Export. Another Export window displays.



4 Enter the name of the new export file.

5 In the Directories list box, specify the directory in which to save the export file.

6 Click Export.

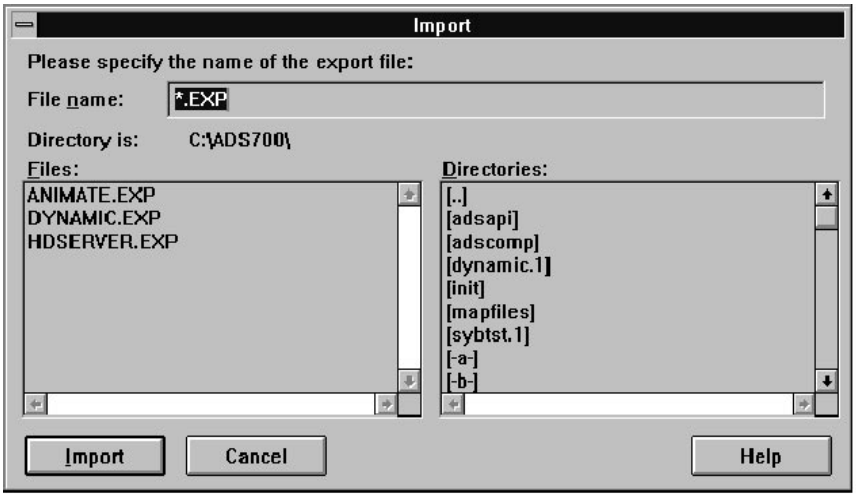
Result: The object exports to an export file in the specified directory.

Importing objects from external files

What to do

To import an object from an external file, use the Import window.

1. Select the object under which the imported object is to exist.
2. Select Object.Import. The Import window displays.



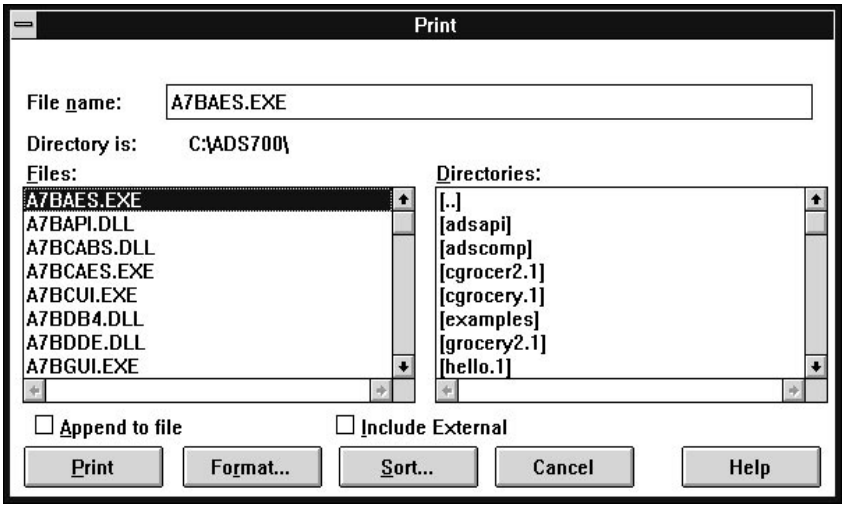
3. If necessary, change the directory in the Directories list box.
4. Select an export file in the Files list box.
5. Click Import.

Result: The selected export file is imported into the knowledge base and placed under the selected object in the knowledge base hierarchy.

Printing selected objects

What to do

1 To print or save information about selected objects to a file, select Object.Print after selecting objects from an Object List window.



2 Use the File name field to specify the file in which to save the object information. You can enter a path name or use the Directories list box to specify a path.

3 To print out the object information, you can enter the name of your printer, which in many installations is LPT1 or PRN. See your operating system documentation to determine the exact name of your printer.

4 To append the object information to the end of a file, select the file, then check Append to file.

5 To overwrite the contents of a file with the object information, select the file and make sure Append to file is unchecked.

6 To arrange the order in which the object information prints, click Sort. The Print Format window displays.

When you have changed the print format attributes, click OK to return to the Print window.

7 To perform any of the following tasks, click Format.

- enter header and footer text for each page
- change page width and length
- indicate printer type
- compress output into as few lines as possible
- add an index to the end of the printout
- add a table of contents to the end of the printout
- add an extended table of contents containing entries for all objects to the end of the printout
- add a summary of all objects printed to the end of the printout

When you complete changing the print format attributes, click OK to return to the Print window.

8 Click Print to print the information of the selected objects or save it to a file.

📖 *For more information:* See Chapter 3, "Working at the Knowledge Base Level," to learn about arranging the print order of the selected objects and the format of the print output.

Making a state the entry state

Overview

The entry state is the state in which a knowledge base begins executing. Each knowledge base has one entry state, which is named Main by default. Main is represented by the following icon:

What to do

To make another state the entry state:

1. Select the state.
2. Select Object.Set entry state.

Chapter 6
Working with the Window Editor

Introduction

This chapter shows you how to build a graphical user interface using the Window editor. You use the Window editor to build the main window, other new windows, dialog boxes, controls, OLE controls and objects, and the DDE system.

In this chapter

Continued

Continued

Continued

Continued

Overview

Knowledge base's application window

When you start a knowledge base, the knowledge base's entry state executes. Use the entry state to open the application window from which to access all knowledge base functions. The knowledge base application window serves the same purpose as the AionDS primary window.

From the application window, you can open other windows to perform specific functions. For example, you can select File.Open in a graphics program to open a dialog from which you can select a file to open and modify.

Other windows and dialogs

AionDS knowledge bases distinguish between the following types of windows:

- top-level windows
- control windows
- desktop window

This distinction is based on the way AionDS opens the windows.

Top-level windows

Top-level windows are opened directly by knowledge base logic. There are two types of top-level windows:

- **Standard windows** have menu bars and allow the user to choose between a number of application features or functions.
- Generally, **Dialog boxes** do not have menu bars and exist to prompt the user for specific types of information. Dialog boxes open in response to user selections from a standard window menu bar or other dialog boxes.

Control windows

Control windows, or controls, display in top-level windows. Controls allow users to select options or enter information. Controls include such interface elements as push buttons, menu items, radio buttons, text windows, and scroll bars. Controls open when the top-level window in which they display opens.

Desktop window

The parent window for top-level windows is the Desktop window, which represents the physical screen. AionDS automatically creates and opens the Desktop window when required by the system. This usually occurs when a standard window starts.

Top-level windows created without a parent are children of the Desktop window. The Desktop window opens child windows.

You can customize the Desktop window for error processing as well as define the initial size of standard windows and dialog boxes as a fraction of the Desktop window's size.

📖 *For more information:* See the *Building GUI Applications: Reference* for details about customizing Desktop windows.

Procedure for building the graphical user interface

Overview

This section contains general procedures for building the graphical user interface—windows, dialogs, and controls.

Building the graphical interface

To build the graphical user interface:

- create the knowledge base
- build the knowledge base's application window
- build other windows and dialogs

Procedure for building new windows, dialogs, and controls

Overview

To build new windows, dialogs, and controls, first open the Window editor. To build new windows you use:

Feature	Function
New Window dialog OR tool palette	creating new windows, dialogs, and controls
Slots dialog	customizing each window, dialog, and control
mouse OR cursor keys	adjusting the window, dialog, and controls to the desired size

You can build the following controls:

- label windows
- text windows
- list boxes
- column list boxes
- combo boxes
- push buttons
- radio buttons
- check boxes
- group boxes
- tool bars
- tools
- scroll bars
- pull-down menus
- bitmap windows
- hot regions
- icon windows
- OCX controls
- OLE objects

Each window's slots dialog contains attributes you can modify. Although each window, dialog, and control has its own unique slots dialog, many attributes are the same. In the slots dialog for each window, dialog, and control, you can customize most of the following attributes:

- instance name
- owning class
- title or text displayed in the window, dialog, or control
- window, dialog, or control styles
- user slot values
- parameters, slots, and messages attached to windows, dialogs, or controls
- mouse pointers
- minimize icons

You can add customized logic to the following controls:

- list boxes
- multi-column list boxes
- combo boxes
- push buttons
- radio buttons
- check boxes
- tools
- icon window
- hot region
- scroll bars
- pull-down menus
- OCX controls

📖 *For more information:* See the *Building GUI Applications: Tutorial* and *Building GUI Applications: Reference* for details about windows, dialogs, and controls.

What to do

To build a new window or dialog and add controls:

1. Open the Window editor.
2. Create a new window or dialog.
3. Customize the window or dialog using the slots dialog.
4. Use the mouse or cursor keys to adjust the window or dialog to the desired size.
5. Add controls to the window and adjust the controls to the desired size.
6. Customize the controls using the slots dialog.
7. Add logic to the controls.

📖 *For more information:* See “Selecting methods and adding functions” on page 6-97.

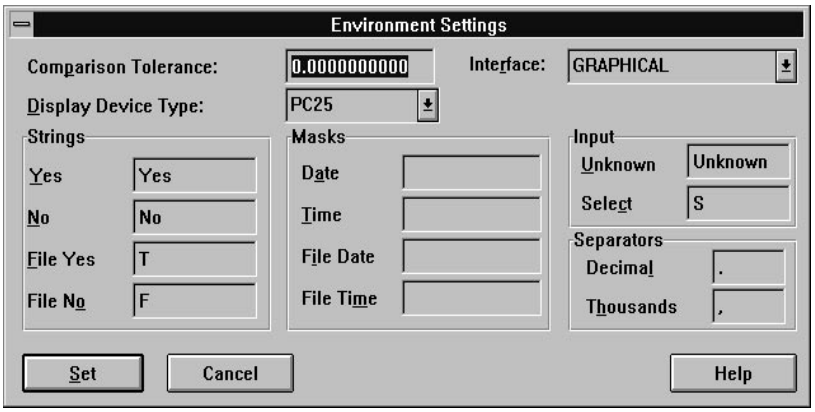
Creating a knowledge base

Overview

To create a knowledge base with a graphical user interface, set the AionDS development environment to graphical.

What to do

1 Select Settings.Environment. The Environment Settings window displays.



2 Specify `graphical` in the Interface field.

3 Click Set.

Result: The AionDS development environment is set to graphical mode.

4 Create a knowledge base. Select File.New. The Create Knowledge Base window displays.

5 After entering the correct information in the window, click New.

Result: The following actions occur:

- AionDS creates a knowledge base.
- The knowledge base hierarchy window displays.
- The entry state, WindowObject class library, and DDE class hierarchy are loaded.

📖 *For more information:* See Chapter 3, "Working at the Knowledge Base Level," for details about the Create Knowledge Base window.

Opening the Window editor

Overview

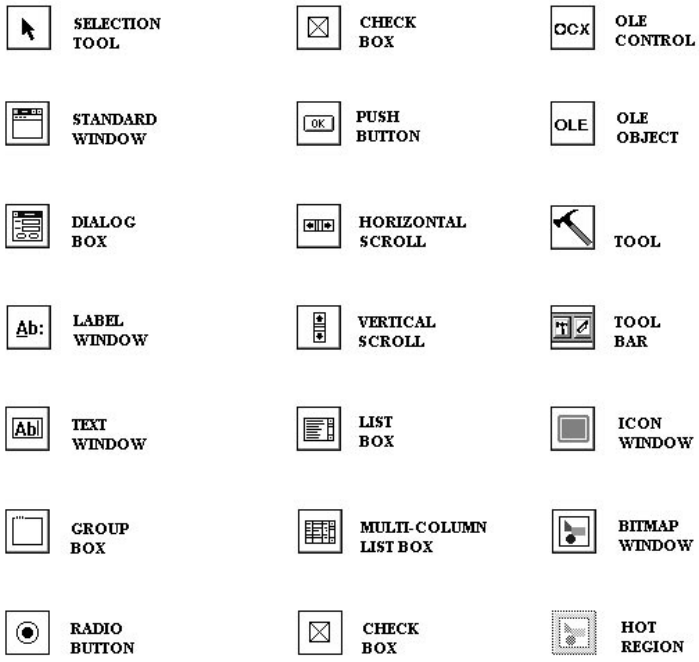
To build windows for the knowledge base, open the Window editor.

What to do

➢ To open the Window editor, select Object.Screen.

Result: The Window editor, Position dialog, and tool palette display. AionDS adds the Screen pull-down menu.

Tool palette



The tool palette consists of buttons you click to create the different windows and controls.

Screen pull-down menu

The Screen pull-down menu contains the following options:

Option	Opens
New Window	New Window dialog to create a new window.
Open Window	Open Window dialog to open an existing window.
Slots	slots dialog of the current window. From the slots dialog, you can change the attributes of the window.
Methods	Methods dialog of the current window. From the Methods dialog, you can select a method and add logic for the window.
Menu Items	Menu Items dialog to build menu items.
Order Children	Order Children dialog to create tabbing sequences and organize groups of child windows.
New Image	New Image dialog to create a new image instance.
Open Image	Open Image dialog to display the slots dialog of an existing image.
New Hot Region	New Hot Region dialog to create a new hot region instance.
Edit Hot Region	Edit Hot Region dialog to select the hot region to display.
Grid	Grid dialog to specify the attributes of the grid.
Position	Position dialog that displays the position and size of the current window.
Install OCX	Insert Object dialog to install an OCX control.
Insert OLE Object	Insert Object dialog to insert an OLE object.
DDE	dialogs used to build static DDE systems. This menu item contains the Conversations, Links, and Topics submenus.

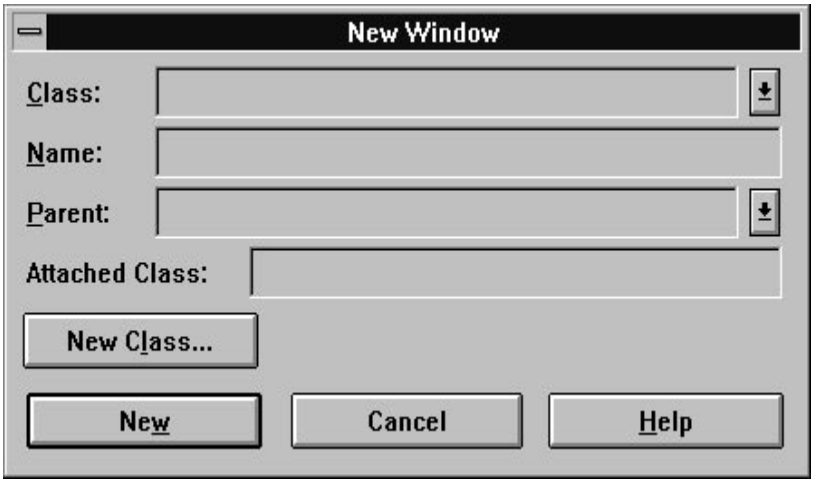
Creating windows, dialogs, and controls

Overview

To create windows, dialogs, and controls, use the New Window dialog. Controls must reside in the parent, top-level windows, or dialogs.

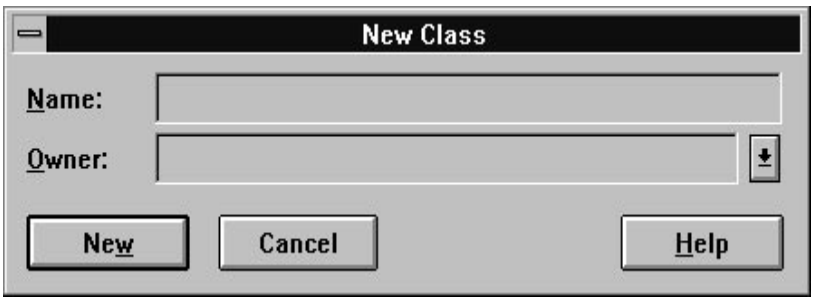
What to do

1 Select Screen.New Window. The New Window dialog displays.



2 In the Class field, select the class for the new window, dialog, or control. The new window, dialog, or control is an instance of this class. The class can be a top-level window or dialog, or a control.

3 To create a new class for the window, dialog, or control instance, click New Class. The New Class dialog displays.



Top-level windows, which are standard windows and dialog box windows, should reside in subclasses so that the class hierarchy can be seen clearly and objects are separated from the WindowObject class library. You must subclass the standard window and dialog box classes of the WINLIB before creating instances.

4 In the Name field, enter a name for the new class.

5 In the Owner field, select the parent class of the new class.

6 Click New to return to the New Window dialog.

7 In the Name field, type an instance name for the new window, dialog, or control.

8 In the Parent field, specify the parent window if you want to make that window a top-level window. This creates the window on the Desktop.

📖 *For more information:* You can also customize aes_Desktop. See the *Building GUI Applications: Reference* for details about customizing aes_Desktop.

To specify a control, specify the window or dialog in which to display the control as the parent.

☑ *Tip:* If a window or dialog currently is selected, that window or dialog displays in the Parent field.

9 Click New. The new window, dialog, or control displays.

Opening windows and dialogs

Overview

To open existing windows and dialogs, use the Open Window dialog.

What to do

1 Select Screen.Open Window. The Open Window dialog displays.

2 Select the window to open in the Window Instances list.

3 If the window to open is a child window and the parent is unknown, check List Child Windows. The child windows of each window display.

4 Click Open.

Result: The selected window opens. If the selected window is a child window, the parent window and all associated child windows open.

Resizing and moving windows, dialogs, and controls

Overview

After you create a window, dialog, or control, you can use the mouse to adjust the size of the window, dialog, or control. You can also use the Position dialog and the Grid command to help resize the window, dialog, or control.

Resizing

1. To resize a window, dialog, or control, place the tip of the arrow on the side, top, or bottom of the window, dialog, or control until the arrow turns into a double-headed arrow (called a resize handle).
2. Drag the mouse to pull or push the side, top, or bottom in the desired direction.

   Result: The side, top, or bottom of the window, dialog, or control is pulled or pushed in the desired direction.

Moving

1. To move a window, dialog, or control, place the tip of the arrow inside the window, dialog, or control.
2. Click and hold down the mouse button. The arrow turns into a four-headed arrow.
3. Drag the window, dialog, or control to the desired place.

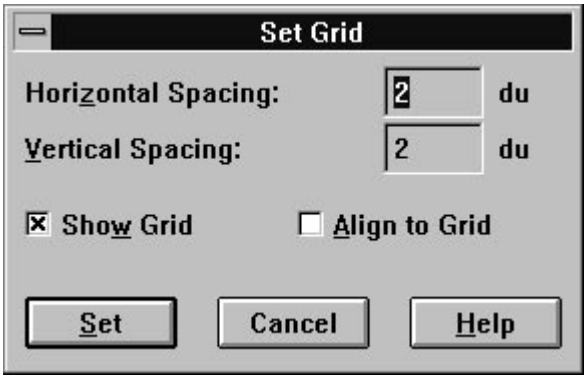
Grid

Use the Grid command to display and customize a grid of regularly spaced dots in a window or dialog. You can use this grid to position and size your controls accurately.

Horizontal and vertical spacing is measured in dialog units (du).

What to do

1 Select Screen.Grid. The Set Grid dialog displays.



2 In the Horizontal Spacing and Vertical Spacing fields, enter a numeric value to increase or decrease the distance between dots. The default value is 2.

☑ *Tip:* The higher the number you enter in the spacing fields, the higher the number of dots and the smaller the distance between dots.

3 Check Show Grid to display the grid.

4 Check Align to Grid to place the borders of controls flush with the grid—that is, the controls can be moved and resized only in increments of dialog units.

Position dialog

Use the Position dialog to display the position and size of a window, dialog, or control.

What to do

- If the Position dialog is not displayed, select Screen.Position. The Position dialog displays.

☑ *Tip:* If you cannot see the Position dialog in the Window editor, and the Position command is checked, the Position dialog is outside the Window editor borders. You can enlarge the Window editor to display the Position dialog.

The Position dialog fields contain the following information:

Name	The name of the currently selected window.
X Coordinate	If a top-level window currently is open in the Window editor, this field contains the location of the left border of the currently selected window from the left border of the screen. The location is given in dialog units for all instances of windows except for Hot Region windows. For Hot Region windows, the location is given in pixels.
Y Coordinate	If a top-level window is currently open in the Window editor, this field contains the location of the top border of the currently selected window from the top of the screen. The location is given in dialog units for all instances of windows except for Hot Region windows. For Hot Region windows, the location is given in pixels.

Width	If a top-level window is currently open in the Window editor, this field displays the width of the currently selected window. The width is given in dialog units for all instances of windows except for Hot Region windows. For Hot Region windows, the width is given in pixels. For Icon windows and Vertical Scroll Bar windows, this field is blank and cannot be modified.
Height	If a top-level window is currently open in the Window editor, this field displays the height of the currently selected window. The height is given in dialog units for all instances of windows except for Hot Region windows. For Hot Region windows, the height is given in pixels. For Icon and Horizontal Scroll Bar windows, this field is blank and cannot be modified.

Copying, cutting, pasting, and deleting windows

Overview

You can copy or move a window from one window into another by using Edit.Cut GUI window, Edit.Copy GUI window, and Edit.Paste GUI window. You can also delete a GUI window by using Delete.GUI window.

Copying a child window into another window

1. Select the child window.
2. Select Edit.Copy. The child window is copied to the clipboard.
3. Select Screen.Open Window.
4. Open the window to which you want to copy the child window.
5. Select Edit.Paste.

   Result: The child window is pasted into the selected window.

Moving a child window into another window

1. Select the child window.
2. Select Edit.Cut. The child window is cut from the parent window and placed on the clipboard.
3. Select Screen.Open Window.
4. Open the window into which you want to place the child window.
5. Select Edit.Paste.

   Result: The child window is pasted into the selected window.

Deleting windows

1. Select the window to delete.
2. Select Edit.Delete GUI window. If the window being deleted has child windows, a confirmation dialog displays.
3. To delete the window and all child windows, click Delete.

   Result: The window and all its child windows are deleted.
4. If you do not want to delete the window and all its child windows, click Cancel.

   Result: You return to the open window and the window and its child windows are not deleted.

Building standard windows

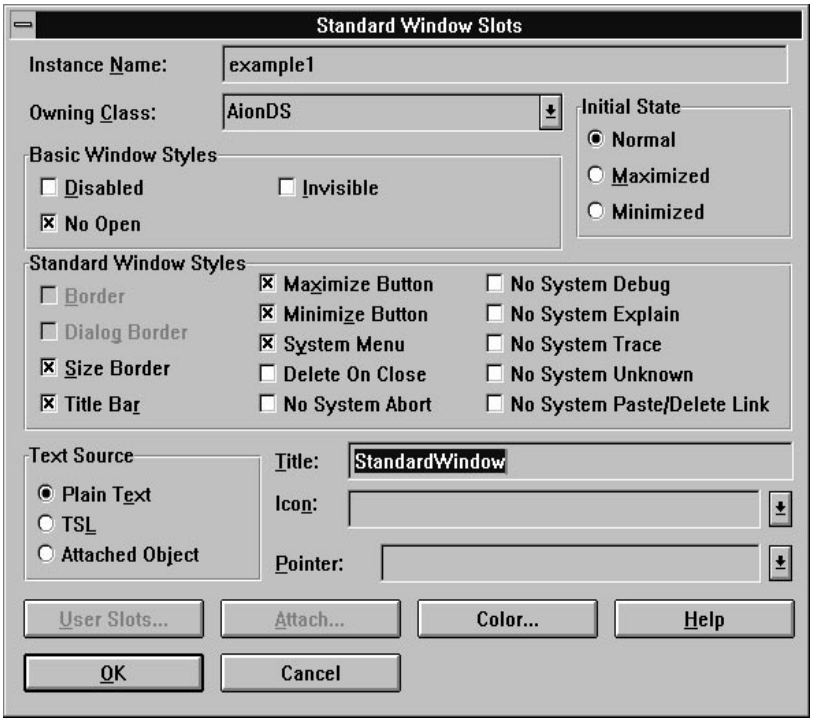
Overview

A **standard window** is a major piece of the user interface. A standard window logically and visually groups together controls, such as menu items and push buttons, in a single window. Users can input complex information by choosing combinations of values and textual input with controls. A standard window usually displays possible actions in drop-down menus. You must make a subclass of the WINLIB StandardWindow class in the knowledge base before creating instances.

What to do

1. Subclass the standard window.
2. Resize and position the new window.

3 Double-click on the standard window. The Standard Window Slots dialog displays.



4 To rename the window instance, enter a new name in the Instance Name field. AionDS automatically updates parent and child links with the new window instance name.

5 You can assign a window instance to another class or subclass by typing a new class name in the Owning Class Field. AionDS automatically updates parent and child links with the new window instance name.

6 In the Initial State group, specify how to display the window when opened.

7 In the Basic Window Styles group, specify whether the window is initially enabled or visible and whether it is opened automatically when its parent window is opened.

8 In the Standard Window Styles group, define the display properties of a standard window, such as whether the border of your window is resizeable, whether the user of your application can maximize or minimize your window, and how many options are displayed in the pull-down system menu.

📖 *For more information:* See the *Building GUI Applications: Reference* for details about the different styles.

9 In the Text Source group, specify the source of the text displayed in the window at runtime.

10 In the Title field, change the text to the name that you want displayed in the title bar.

11 In the Icon field, specify an icon to display when the user minimizes the window. If this field is left blank, a default icon is supplied.

☑ *Tip:* If the Minimize Button style is not selected, the window cannot be minimized.

12 In the Pointer field, select the mouse pointer shape that you want to display when the mouse pointer enters the window.

📖 *For more information:* See "Changing the mouse pointer in a window" on page 6-113.

13 To modify the values of slots owned by the window's parent class, click the User Slots dialog and enter the appropriate information.

📖 *For more information:* See "Changing user slot values" on page 6-101.

14 To attach an object to the standard window, click Attach to display the Attach Object dialog and enter the appropriate information.

📖 *For more information:* See "Attaching parameters, slots, or messages to windows" on page 6-99.

15 Click Back Color to select the background color for the standard window.

📖 *For more information:* See "Changing window background and text colors" on page 6-115.

16 Click OK. The changes are applied and the window displays.

17 Add controls to the window and adjust the controls to the desired size.

18 Customize the controls using the slots dialog.

19 Select methods and add functions to the controls.

📖 *For more information:* See “Selecting methods and adding functions” on page 6-97.

20 Press F8 to save and close the window.

Building an application window

Overview

Most graphical knowledge bases begin with an application window. When a user starts the knowledge base, the application window displays first. The application window has a set of initial options in pull-down menus, tool palettes, or icons. To build an application window, you create a standard window, build the menus, tool palettes, or icons displayed in it, and write the logic executed when a user selects a menu item, tool, or icon.

To open the application window when the knowledge base is started, set up the entry state agenda to send the predefined OpenApp method to the application window.

What to do

To build an application window:

1. Build a standard window with a new class. The application window must be a top-level window.
2. Add controls including menu items with corresponding methods and functions.

   📖 *For more information:* See "Selecting methods and adding functions" on page 6-97.
3. Customize the controls using the slots dialog.
4. Press F8 to save and close the window.
5. Modify the entry state agenda to open the application window. Double-click on the entry state name in the knowledge base hierarchy window to open the editor for the entry state.

6 In the Agenda, on the first line type the following statement:

```
ApplicationWindow.OpenApp
```

where *ApplicationWindow* is the name of the application window instance.

7 Press F8 to check the statement syntax, then save and close the entry state.

Result: This statement is saved as the first line in the entry state agenda and is the first statement executed when the knowledge base is started. The OpenApp method turns control over to the open application window. At this point the application is ready to receive user events such as mouse clicks and menu selections.

Building ask, show and other dialog boxes

Overview

A **dialog box** is very similar to a standard window. Like a standard window, a dialog box groups logically-related controls in a single dialog. The Dialog Box class is actually a subclass of the Standard Window class. You must make a subclass of the WINLIB DialogBox and its subclasses, AskDialog and ShowDialog, in the knowledge base before creating customized instances.

While standard windows often display actions in pull-down menus, a dialog box typically displays only one set of values and represents the users' options with push buttons instead of menus.

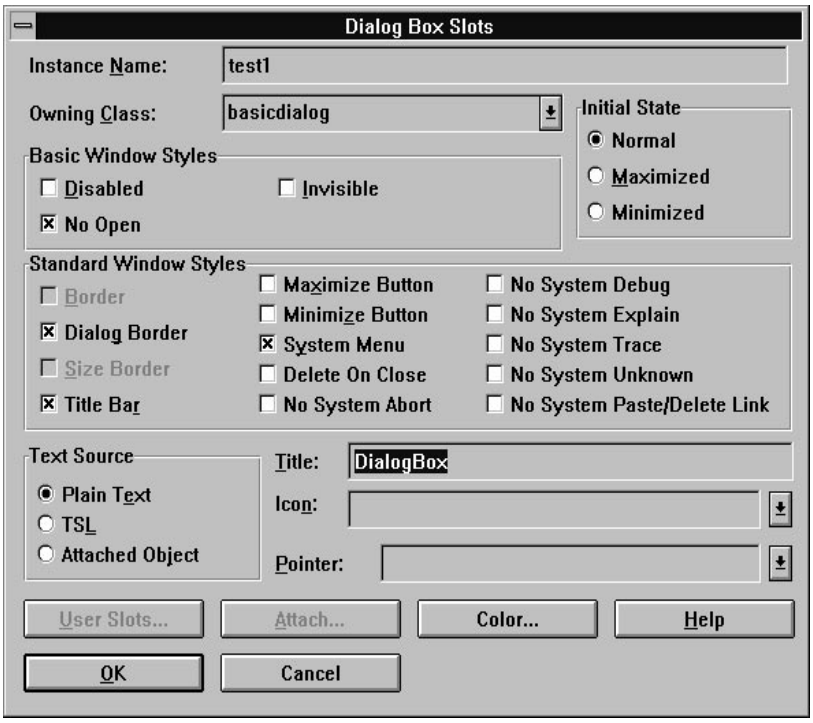
An **ask dialog** is a specialized dialog box that the application displays when the inference engine is in control of processing. A **show dialog** is a specialized dialog box that the inference engine displays when it is in control of processing.

📖 *For more information:* See the *Building GUI Applications: Reference* to learn more about ask dialog and show dialog.

To build dialog boxes, use the Window editor.

What to do

1. Create and resize a dialog box with a new class.
2. Double-click in the dialog box. The Dialog Box Slots dialog displays.



3 To rename the dialog box instance, enter a new name in the Instance Name field. AionDS automatically updates parent and child links with the new window instance name.

4 You can assign a dialog box instance to another dialog box class or subclass by typing a new class or subclass name in the Owning Class field. AionDS automatically updates parent and child links with the new dialog instance name.

5 In the Initial State group, specify how to display the dialog box when it is opened.

6 In the Basic Window Styles group, specify whether the dialog box is initially enabled or visible and whether it is opened automatically when its parent window opens.

📖 *For more information:* See the *Building GUI Applications: Reference* for details about the different styles.

7 In the Standard Window Styles group, define the display properties of a dialog box, such as whether the border of your dialog box is resizeable, whether the user of your application can maximize or minimize your dialog box, and how many options display in the pull-down system menu.

📖 *For more information:* See the *Building GUI Applications: Reference* for details about the different styles.

8 In the Text Source group, specify the source of the text that displays in the dialog box at runtime.

If you select the Plain radio button, text from the Title field displays in the window's title bar.

If you select the TSL radio button, you can enter TSL text into the Title field.

If you select the Attached Object radio button, attach an object with the desired text to display in the title bar.

9 To attach an object to the dialog box, click Attach to display the Attach Object dialog and enter the appropriate information.

📖 *For more information:* See "Attaching parameters, slots, or messages to windows" on page 6-99.

10 To modify the values of slots owned by the dialog box's parent class, click the User Slots dialog and enter the appropriate information.

📖 *For more information:* See "Changing user slot values" on page 6-101.

11 In the Title field, change the text to the name to display in the title bar.

12 In the Icon field, specify an icon to be displayed when the user minimizes the dialog box. If this field is left blank, a default icon is supplied.

☑ *Tip:* If the Minimize Button style is not selected, the dialog box cannot be minimized.

13 In the Pointer field, select the mouse pointer shape to display when the mouse pointer enters the dialog box.

📖 *For more information:* See "Changing the mouse pointer in a window" on page 6-113.

14 Click Back Color to select the background color for the dialog box.

📖 *For more information:* See "Changing window background and text colors" on page 6-115.

15 Click OK. The changes are applied and the dialog box displays.

16 Add controls to the dialog box and adjust the controls to the desired size.

17 Customize the controls using the slots dialog.

18 Select methods and add functions to the controls.

📖 *For more information:* See "Selecting methods and adding functions" on page 6-97.

19 Press F8 to save and close the dialog box.

Building label and text windows

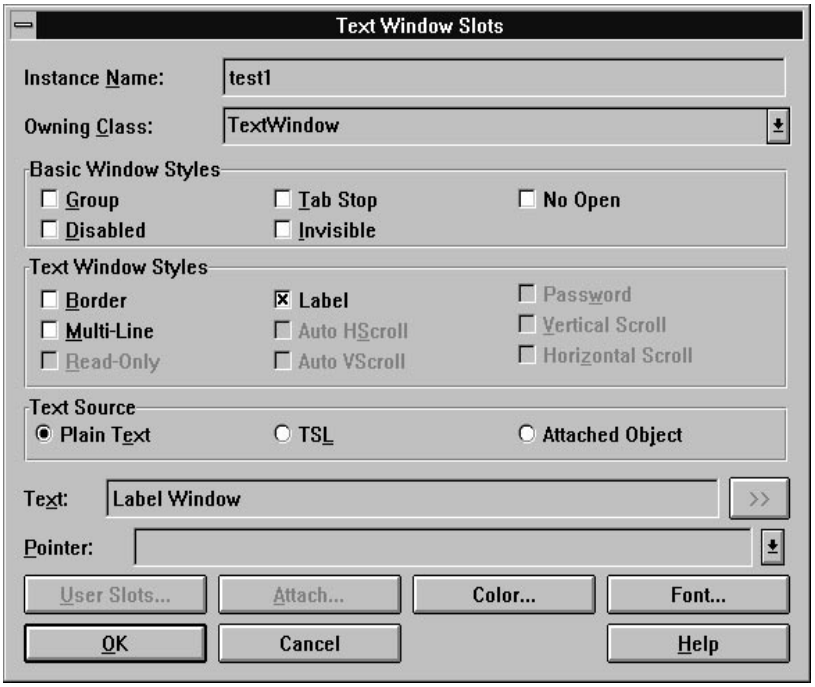
Overview

A **text window** can display text or accept text input from end users. A **label window** is a type of text window that only displays text. Text windows can be used in many different ways. When creating a text window, be sure to select a style that supports the way the text window is used in the application.

What to do

1 Create and resize a label or text window.

2 Double-click on the label or text window. The Text Window Slots dialog displays.



3 To rename the label or text window instance, enter a new name in the Instance Name field. AionDS automatically updates parent and child links with the new label or text window instance name.

4 You can assign the label or text window instance to another label or text window subclass by selecting a new subclass name in the Owning Class field. AionDS automatically updates parent and child links with the new label or text window instance name.

5 In the Basic Window Styles group, you can specify the label or text window as part of a group by selecting on the Group check box. To specify the Tab Stop style for the label or text window, select the Tab Stop check box.

📖 *For more information:* See the *Building GUI Applications: Reference* for details about the different styles.

6 In the Text Window Styles group, define the display properties of a label or text window, such as multi-line, label, and horizontal and vertical scroll bars.

📖 *For more information:* See the *Building GUI Applications: Reference* for details about the different styles.

7 In the Text Source group, specify the source of the text that displays in the label or text window at run time.

If you select the Plain radio button, text from the Text field displays in the label or text window.

If you select the TSL radio button, you can enter TSL text into the Text field.

If you select the Attached Object radio button, attach an object with the desired text to display in the label or text window.

8 To attach an object to the label or text window, click Attach to display the Attach Object dialog and enter the appropriate information.

📖 *For more information:* See “Attaching parameters, slots, or messages to windows” on page 6-99.

9 In the Text field, change the text to the name that you want displayed in the label or text window. You can also add mnemonics to a label window.

📖 *For more information:* See “Adding mnemonics to controls” on page 6-106.

10 In the Pointer field, select the mouse pointer shape to display when the mouse pointer enters the label or text window.

📖 *For more information:* See "Changing the mouse pointer in a window" on page 6-113.

11 To modify the values of slots owned by the label or text window's parent class, click the User Slots dialog and enter the appropriate information.

📖 *For more information:* See "Changing user slot values" on page 6-101.

12 Click Back Color to select the background color for the label or text window.

📖 *For more information:* See "Changing window background and text colors" on page 6-115.

13 Click Text Color to select the color for the text in the label or text window.

📖 *For more information:* See "Changing window background and text colors" on page 6-115.

14 Click OK. The changes are applied and the label or text window displays.

Building list boxes

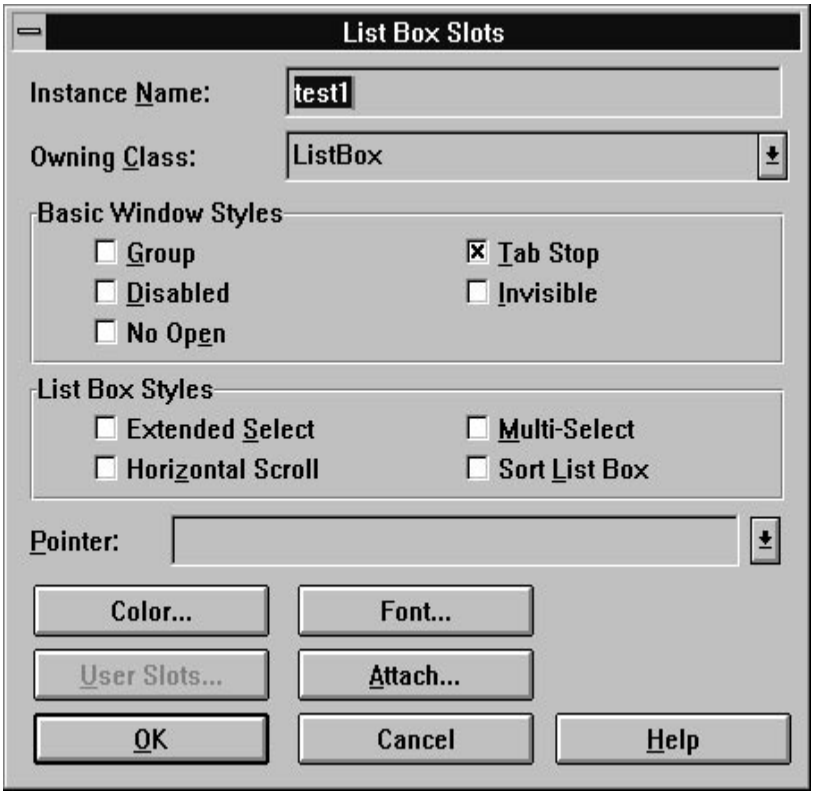
Overview

A **list box** displays a list of text items for selection. You can define the list items in an attached object.

What to do

1 Create and resize a list box.

2 Double-click on the list box. The List Box Slots dialog displays.



3 To rename a list box instance, enter a new name in the Instance Name field. AionDS automatically updates parent and child links with the new list box instance name.

4. You can assign a list box instance to another list box subclass by selecting a new subclass name in the Owning Class field. AionDS automatically updates parent and child links with the new list box instance name.

5. In the Basic Window Styles group, you can specify the list box as part of a group by clicking on the Group check box. You can also specify the Tab Stop styles for the list box by selecting the Tab Stop check box.

   📖 *For more information:* See the *Building GUI Applications: Reference* for details about the different styles.

6. In the List Box Styles group, define the display properties of the list box, such as allowing multiple selections and sorting the list alphabetically.

   📖 *For more information:* See the *Building GUI Applications: Reference* for details about the different styles.

7. To attach an object to the list box, click Attach to display the Attach Object dialog and enter the appropriate information.

   📖 *For more information:* See "Attaching parameters, slots, or messages to windows" on page 6-99.

8. In the Pointer field, select the mouse pointer shape you want displayed when the mouse pointer enters the list box.

   📖 *For more information:* See "Changing the mouse pointer in a window" on page 6-113.

9. To modify the values of slots owned by the list box's parent class, click the User Slots dialog and enter the appropriate information.

   📖 *For more information:* See "Changing user slot values" on page 6-101.

10. Click OK. The changes are applied and the list box displays.

11. Select a method and add a function to the list box.

    📖 *For more information:* See "Selecting methods and adding functions" on page 6-97.

Building multi-column list boxes

Introduction

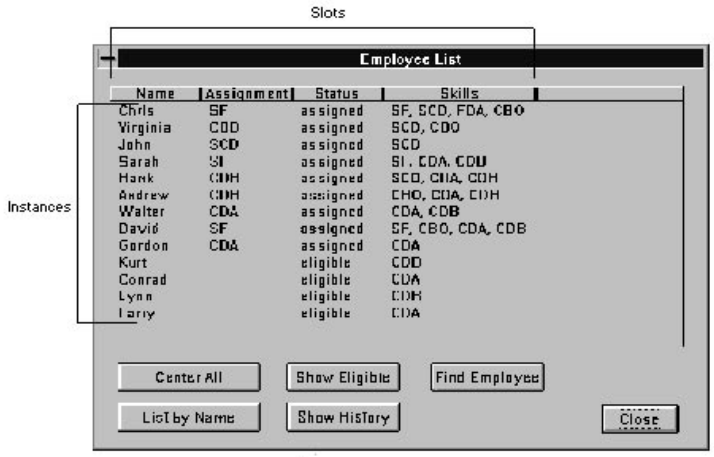
A multi-column list box displays instances in a class. The columns display the slots within the class, while the rows display the instance data.

During runtime you can select rows, but you cannot enter or change the text in the rows.

Components and appearance

The column headings form an optional title bar. The column text can be left-, right-, or center-aligned. In the following example, all columns are left aligned. You can adjust the column widths at runtime. AionDS adds scroll bars as needed.

The following figure is an example of a multi-column list box.



Creating a multi-column list box

Overview

Create an instance of ColumnLB and attach the instance to a **data class**. A data class contains the information displayed in the ColumnLB class. AionDS uses the term “data class” to avoid confusion with the ColumnLB class. AionDS displays the data class’ instance data at runtime.

You can select specific slots to display at runtime.

What to do

1. Create and resize a multi-column list box.
2. Double-click on the multi-column list box. The Columnar List Box Slots dialog displays.

3 To rename a multi-column list box instance, enter a new name in the Instance Name field. AionDS automatically updates parent and child links with the new multi-column list box instance name.

4 You can assign the multi-column list box instance to another multi-column list box subclass by selecting a new class name in the Owning Class field. AionDS automatically updates parent and child links with the new multi-column list box instance name.

5 In the Basic Window Styles group, you specify the multi-column list box as part of a group byselecting the Group check box. You can also specify the Tab Stop styles by selecting the Tab Stop check box.

📖 *For more information:* See the *Building GUI Applications: Reference* for details about the different styles.

6 In the List Box Styles group, you specify how to select items in the multi-column list box.

Extended Select	more than one row by using the mouse with the CTRL and SHIFT keys
Multi-Select	any number of items on the list at one time

7 Use the Pointer field to select a mouse pointer shape to display when the mouse pointer enters the multi-column list box area.

📖 *For more information:* See “Changing the mouse pointer in a window” on page 6-113.

8 To define the background or text color of your window, click Color. The Select Background and Foreground Colors window displays.

📖 *For more information:* See “Changing window background and text colors” on page 6-115.

9 To change the text font, click Font. The Font dialog displays.

🕮 *For more information:* See “Changing fonts” on page 6-118.

10 To modify slot values owned by the multi-column list box class, click User Slots to display the User Slots dialog.

🕮 *For more information:* See “Changing user slot values” on page 6-101.

11 To configure the multi-column list box, click Configure.

12 Select a method and add a function to the multi-column list box.

🕮 *For more information:* See “Selecting methods and adding functions” on page 6-97.

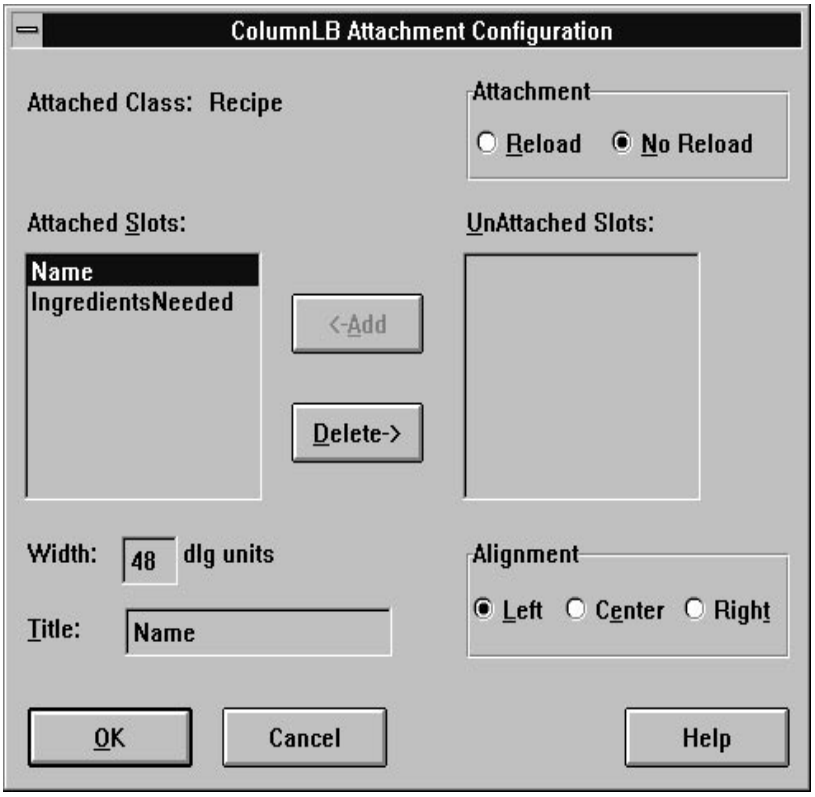
Configuring a multi-column list box

Overview

When you create a multi-column list box, AionDS attaches the multi-column list box to the data class you specify. By default, all slots are columns in a multi-column list box. You can select specific slots to display. You can also control each column's visual appearance, how information updates, and how to select rows at runtime.

Selecting slots

1 From the Slots dialog, click Configure. The ColumnLB Attachment Configuration dialog controls the appearance of the specific columns.



2 To detach slots from the multi-column list box, select the slot in the Attached Slots list and click Delete.

3 To attach slots, select the slot in the Unattached Slots list and click Add.

Ordering columns

Use the ColumnLB Attachment Configuration dialog to set the order of columns in the multi-column list box. The top-to-bottom order of slots in the Attached Slots list determines the left-to-right order of columns in the multi-column list box.

1 Move all slots from the Attached Slots list to the Unattached Slots list by selecting each slot and clicking Delete.

2 Move the slot that should appear leftmost in the multi-column list box to the Attached Slots list by selecting the slot in the Unattached Slots list and clicking Add.

3 From the Unattached Slots list, move additional slots one at a time in the left-to-right order that they will appear in the multi-column list box to the Attached Slots list. AionDS adds each slot to the end of the Attached Slots list.

Formatting columns

Use the ColumnLB Attachment Configuration dialog to format individual columns.

1 Select a column in the Attached Slots list.

2 Specify the column heading in the Title field. The default is the Print Name property. If the Print Name property is empty, then the Name property is the default.

☑ *Tip:* If you do not specify column titles, no title bar appears.

3 If desired, adjust the column width in the Width field.

☑ *Tip:* Adjust the individual column width by sizing the column with a mouse. Use the Width field to adjust multiple columns to the same width.

4 Select a radio button in the Alignment box to set the alignment of the slot values. By default, text fields are left-aligned and numeric fields are right-aligned.

Automatic update

When the slot values change, you can control whether AionDS automatically updates the information in the multi-column list box. For automatic updates, click Reload in the Attachment box. To control updates with the Refresh method, click No Reload.

✋ *Caution:* These options apply to the entire multi-column list box, not to individual columns.

☑ *Tip:* By default, AionDS automatically assigns the No Reload option.

🕮 *For more information:* See *Building GUI Applications: Reference* for more information about the Refresh method.

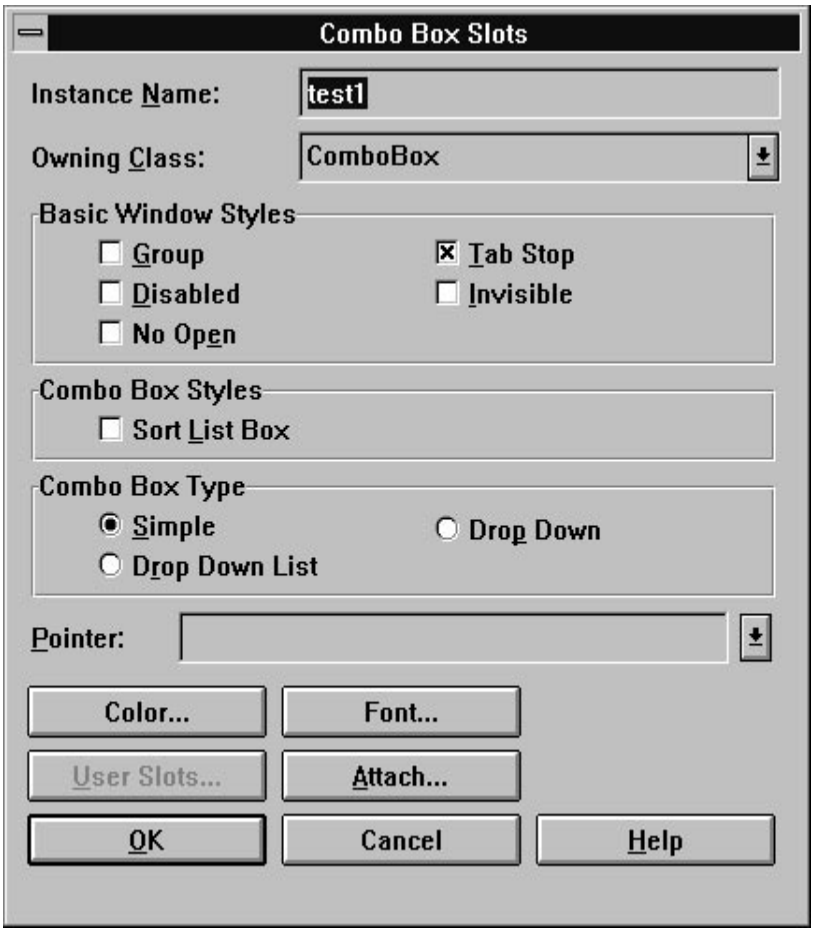
Building combo boxes

Overview

A **combo box** combines the features of a list box and a text window. It displays a list of predefined text items for selection, and also lets users type in items that are not displayed in the list. You usually use a combo box when users will probably choose an existing option, but might create a new one. You can also use a drop-down combo box, because it takes up less window space than a list box.

What to do

1. Create and resize a combo box.
2. Double-click on the combo box. The Combo Box Slots dialog displays.



3 To rename a combo box instance, enter a new name in the Instance Name field. AionDS automatically updates parent and child links with the new combo box window instance name.

4 You can assign a combo box instance to another combo box subclass by selecting a new subclass name in the Owning Class field. AionDS automatically updates parent and child links with the new combo box instance name.

5 In the Basic Window Styles group, you can specify the combo box as part of a group by checking Group. You can also specify the Tab Stop styles for the combo box by checking Tab Stop.

📖 *For more information:* See the *Building GUI Applications: Reference* for details about the different styles.

6 To list the entries alphabetically, select the Sort List Box check box in the Combo Box Styles group.

7 In the Combo Box Type group, you can define one of the following:

Combo box type	Description
Simple	displays the entire list. You can type in an entry that is not listed.
Drop Down List	displays only the current selection. Use the scroll button to display the entire list and choose another selection. You cannot type in an entry.
Drop Down	display only the current selection. You can use the scroll button to display the entire list and choose another selection. You can type in an entry that is not listed.

8 To attach an object to the combo box, click Attach to display the Attach Object dialog and enter the appropriate information.

📖 *For more information:* See "Attaching parameters, slots, or messages to windows" on page 6-99.

9 In the Pointer field, select the mouse pointer shape to display when the mouse pointer enters the combo box.

📖 *For more information:* See "Changing the mouse pointer in a window" on page 6-113.

10 To modify the values of slots owned by the combo box's parent class, click the User Slots dialog and enter the appropriate information.

📖 *For more information:* See "Changing user slot values" on page 6-101.

11 Click OK. The changes are applied and the combo box displays.

12 Select a method and add a function to the combo box.

📖 *For more information:* See "Selecting methods and adding functions" on page 6-97.

Building radio buttons and check boxes

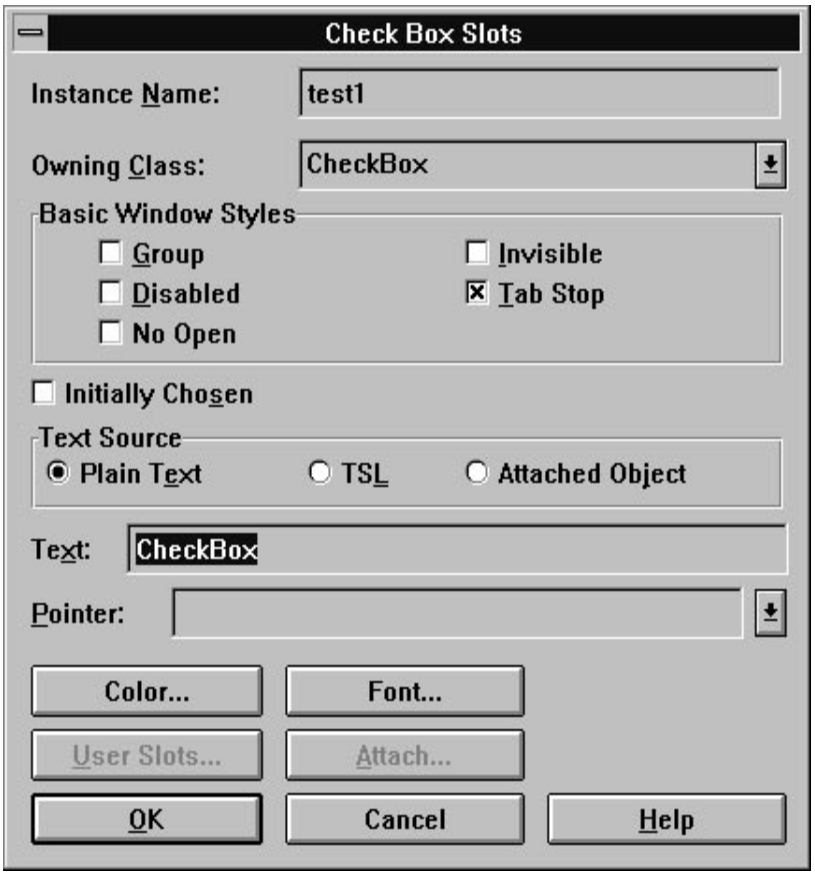
Overview

Radio buttons are graphical buttons that represent a set of related but mutually exclusive options—you can select only one button at a time. You can use group boxes or the Group style to organize radio buttons.

A **check box** is a graphical button that represents a single option that you can turn on or off.

What to do

1. Create and resize a radio button or check box.
2. Double-click on the radio button or check box. The Radio Button Slots or Check Box Slots dialog displays.



3 To rename a radio button or check box instance, enter a new name in the Instance Name field. AionDS automatically updates parent and child links with the new radio button or check box instance name.

4 You can assign a radio button or check box instance to another radio button or check box subclass by selecting a new subclass name in the Owning Class field. AionDS automatically updates parent and child links with the new radio button or check box instance name.

5 In the Basic Window Styles group, you can specify the radio button or check box as part of a group by selecting the Group check box. You can also specify the Tab Stop style for the radio button or check box by selecting the Tab Stop check box.

📖 *For more information:* See the *Building GUI Applications: Reference* for details about the different styles.

6 Check Initially Chosen for the radio button or check box to be selected when the window is opened.

7 In the Text Source group, specify the source of the text that displays in the radio button or check box at runtime.

If you select the Plain radio button, text from the Text field displays in the radio button or check box.

If you select the TSL radio button, you can enter TSL text into the Text field.

If you select the Attached Object radio button, attach an object with the desired text to display in the radio button or check box.

8 To attach an object to the radio button or check box, click Attach to display the Attach Object dialog and enter the appropriate information.

📖 *For more information:* See "Attaching parameters, slots, or messages to windows" on page 6-99.

9 In the Text field, enter the text to display on the radio button or check box. You can also add mnemonics to a radio button or check box.

📖 *For more information:* See "Adding mnemonics to controls" on page 6-106.

10 In the Pointer field, select the mouse pointer shape to display when the mouse pointer enters the radio button or check box.

📖 *For more information:* See "Changing the mouse pointer in a window" on page 6-113.

11 To modify the values of slots owned by the radio button or check box's parent class, click the User Slots dialog and enter the appropriate information.

📖 *For more information:* See "Changing user slot values" on page 6-101.

12 Click OK. The changes are applied and the radio button or check box displays.

13 Select a method and add a function to the radio button or check box.

📖 *For more information:* See “Selecting methods and adding functions” on page 6-97.

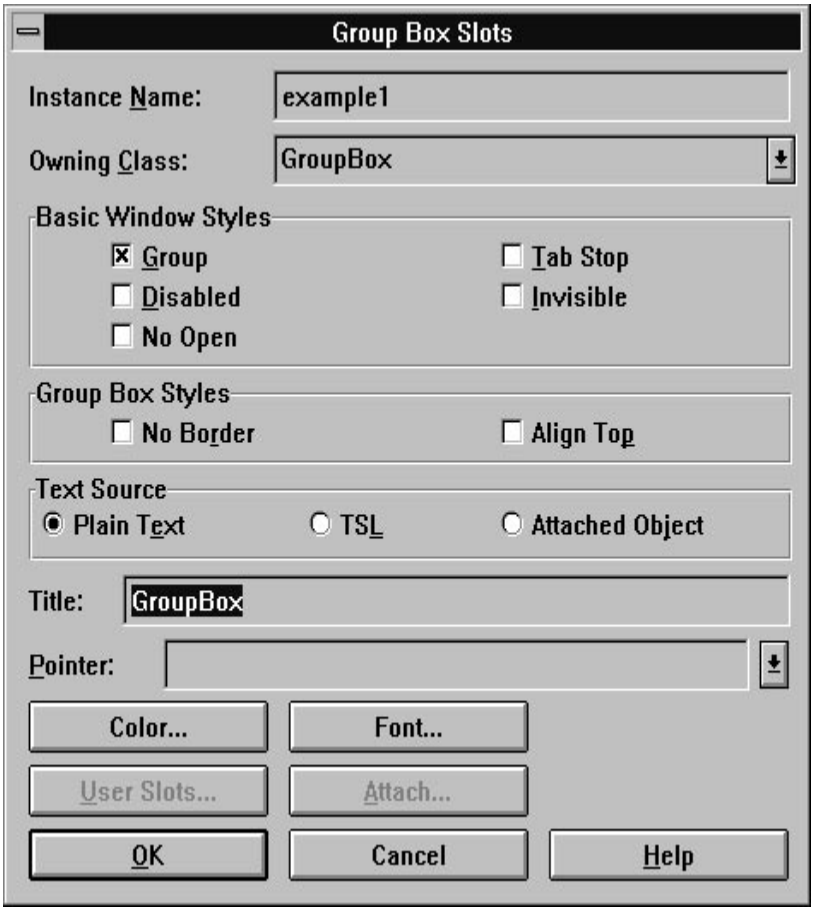
Building group boxes

Overview

A **group box** groups two or more related controls. To include controls in a group box, specify the group box as the parent window for those controls.

What to do

1. Create and resize a group box.
2. Double-click on the group box. The Group Box Slots dialog displays.



3. To rename a group box instance, enter a new name in the Instance Name field. AionDS automatically updates parent and child links with the new group box instance name.

4. You can assign a group box instance to another group box subclass by selecting a new subclass name in the Owning field. AionDS automatically updates parent and child links with the new group box instance name.

5. In the Basic Window Styles group, you can specify the group box as part of a group by clicking on the Group check box. You can also specify the Tab Stop styles for the group box by selecting the Tab Stop check box.

   📖 *For more information:* See the *Building GUI Applications: Reference* for details about the different styles.

6. In the Group Box Styles group, define the display properties of a group box such as whether or not a group box has a border or to align the top of the controls within the group box.

   📖 *For more information:* See the *Building GUI Applications: Reference* for details about the different styles.

7. In the Text Source group, specify the source of the text that displays in the group box at run time.

   If you select the Plain radio button, text from the Title field displays in the group box.

   If you select the TSL radio button, you can enter TSL text into the Title field.

   If you select the Attached Object radio button, attach an object with the desired text to display in the group box.

8. To attach an object to the group box, click Attach to display the Attach Object dialog and enter the appropriate information.

   📖 *For more information:* See "Attaching parameters, slots, or messages to windows" on page 6-99.

9. In the Title field, change the text to the name to be displayed at the top of the group box. You can also add a mnemonic to a group box title.

   📖 *For more information:* See the "Adding mnemonics to controls" on page 6-106.

10 In the Pointer field, select the mouse pointer shape to display when the mouse pointer enters the group box.

📖 *For more information:* See "Changing the mouse pointer in a window" on page 6-113.

11 Click Back Color to change the color of the background within the group box.

📖 *For more information:* See "Changing window background and text colors" on page 6-115.

12 Click Text Color to change the color of the text displayed in the group box.

📖 *For more information:* See the "Changing window background and text colors" on page 6-115.

13 To modify the values of slots owned by the group box's parent class, click the User Slots dialog and enter the appropriate information.

📖 *For more information:* See "Changing user slot values" on page 6-101.

14 Click OK. The changes are applied and the group box displays.

Including controls in a group box

To include controls in a group box:

1 Build a group box.

2 Build the controls to add to the group box and specify the group box as the parent window of each control.

Setting the tabbing order

➢ To set the initial tabbing order for controls in a group box, add the controls to the group box in the desired sequence.

To change the tabbing order, use the Order Children dialog to organize the controls in the group box in the desired order.

📖 *For more information:* See "Organizing controls in tabbing sequences and groups" on page 6-103.

Building pull-down menus

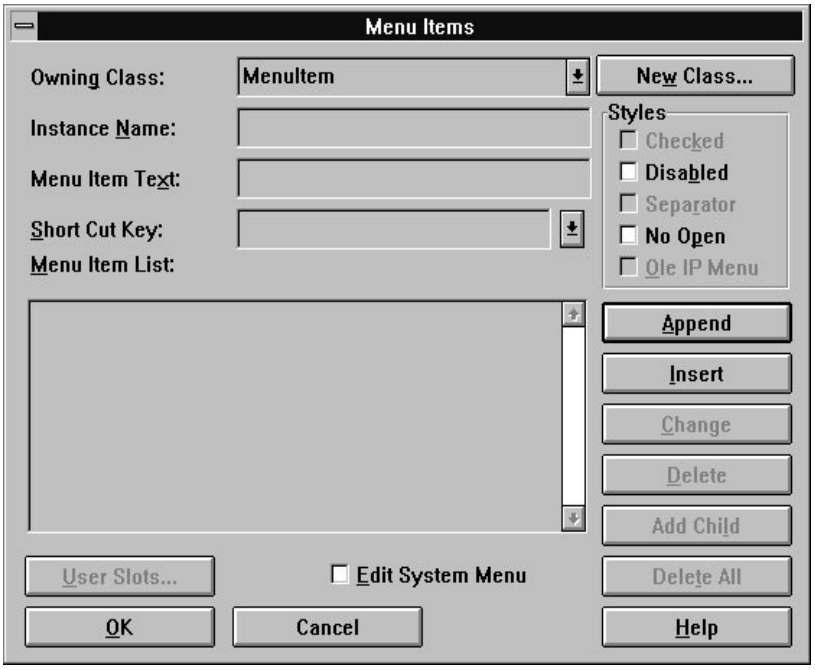
Overview

A **menu item** is a command in a pull-down menu that executes an action. You build pull-down menus to perform complicated tasks. Pull-down menus contain a series of menu items. When you select a menu item, a method and function executes. The method and function usually open another window or execute a sequence of actions such as printing a file or searching for a specific string.

What to do

1 Select Screen.Menu Items. The Menu Items dialog displays.

Reminder: Make sure that a standard window or dialog is selected.



2 To assign a menu item instance to another class, enter a new class name in the Instance Name field. AionDS automatically updates parent and child links with the new menu item instance name.

3 To create a new class for the menu item you are creating, click New Class. The New Class dialog displays.

4 In the Name field, enter a name for the new class.

5 In the Owner field select the parent class of your new class.

6 Click New to return to the Menu Items dialog.

7 In the Instance Name field, enter the menu item's instance name.

8 In the Menu Item Text field, specify the text that displays for this menu item in the pull-down menu.

☑ *Tip:* To make letter in the menu item text part of a mnemonic, enter a tilde (~) directly in front of that letter. See "Adding mnemonics to controls" on page 106.

9 In the Short Cut Key field, select the short cut key combination you can press to execute this menu item.

10 In the Styles group, define the type of menu item, such as a separator line or whether the menu item can be immediately selected when displayed.

11 Check the Edit System Menu check box to add menu items to the system menu.

12 The Menu Item List shows the current order of the menu items in the pull-down menus.

13 You can use the Menu Item List and push buttons on the right side to do the following:

- change menu items
- create pull-down menu selections
- append a menu item after another one
- Insert a menu item before another one
- create menu items as separator lines
- delete a menu item
- delete all menu items

14 To modify the values of slots owned by the menu item's parent class, click the User Slots dialog and enter the appropriate information.

📖 *For more information:* See "Changing user slot values" on page 6-101.

15 Click OK. The changes are applied to the menu items which are added to the menu bar. The parent window displays.

16 To select methods and add functions to the menu items, first select the applicable menu item.

📖 *For more information:* See "Selecting methods and adding functions" on page 6-97.

Creating pull-down menu selections

1 To create a menu item as a selection in a pull-down menu, select the menu item in the Menu Item List to add to a pull-down menu selection.

2 Enter the Instance Name, Menu Item Text, and attributes for the new pull-down menu.

3 Click Add Child.

Result: The menu item is created as a selection in the specified pull-down menu.

Changing menu items

1 To change a menu item, select the menu item in the Menu Item List. Its attributes display.

2 Change the desired attributes.

3 Click Change.

Result: The menu item's attributes are changed.

Appending one menu item after another

1 To append one menu item after another, select the menu item in the Menu Item List after which you want the new menu item to be placed.

2 Enter the Instance Name, Menu Item Text, and attributes for the new menu item.

3 Click Append.

Result: The new menu item is placed after the selected menu item.

Inserting a menu item before another one

1. To insert a menu item before another one, enter the desired attributes of the menu item.
2. In the Menu Item List, select the menu item before which to place the new menu item.
3. Click Insert.

Result: The new menu item is placed after the selected menu item.

Creating separator lines

1. To create a separator line (a line that separates groups of menu items in a pull-down menu), select the menu item below which to place the separator line.
2. Enter the instance name for the separator and check the Separator check box.
3. Click Append.

Result: A separator menu item is placed after the selected menu item.

Deleting a menu item

1. Select the menu item to delete.
2. Click Delete.

Result: AionDS deletes the specified menu item.

Deleting all menu items

➢ To delete all menu items, click Delete All.

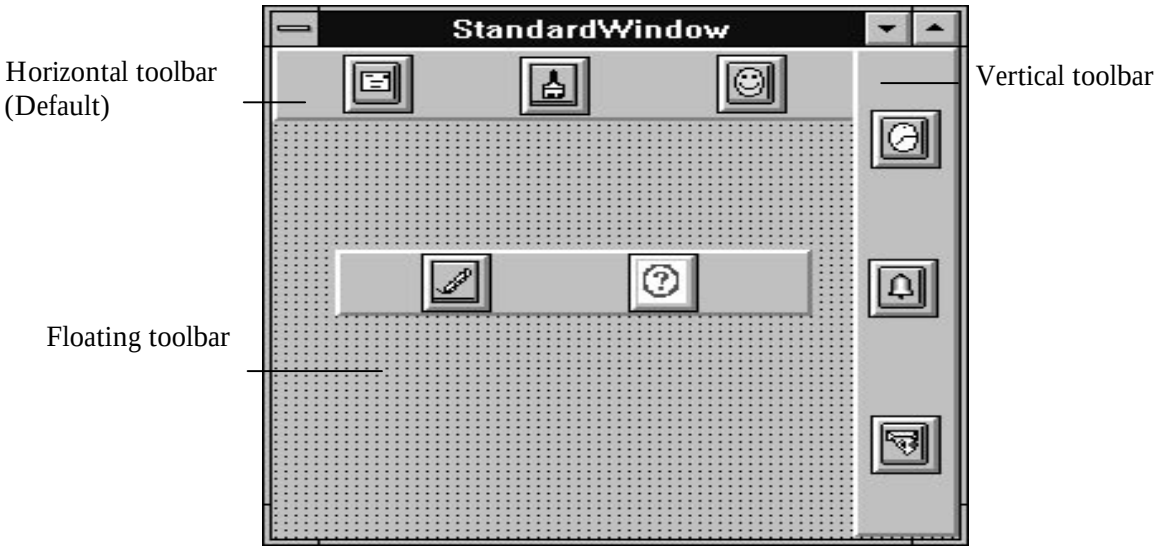
Creating tool bars

Overview

You use a **tool bar** to group tools in a standard window or dialog box.

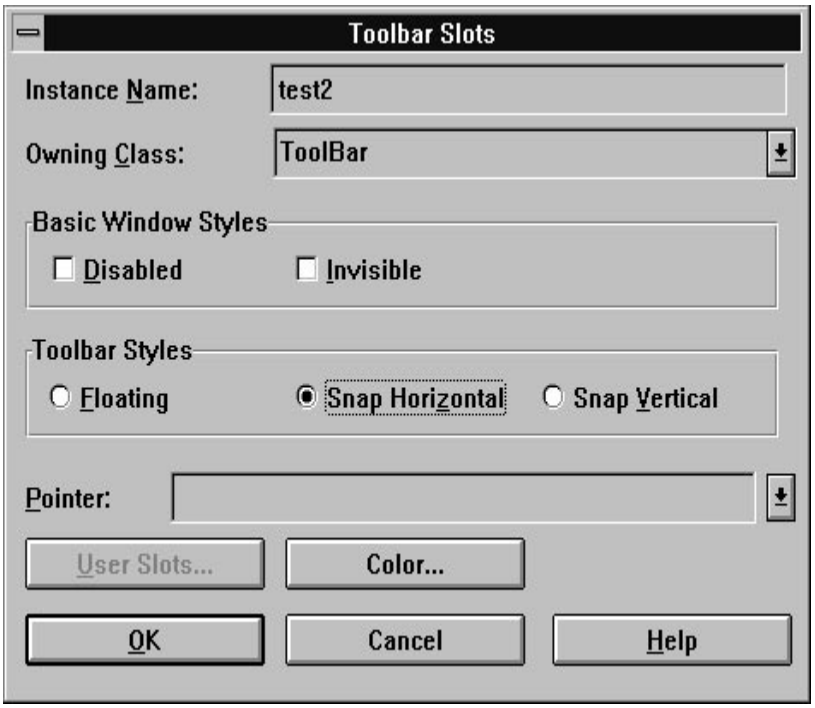
Example

The following example shows horizontal, vertical, and floating tool bars:



What to do

1. Create and resize a tool bar.
2. Double-click on the tool bar. The Tool bar Slots dialog displays.



3 To rename the tool bar instance, enter a new name in the Instance Name field. AionDS automatically updates parent and child links with the new tool bar instance name.

4 You can assign a tool bar instance to another tool bar subclass by selecting a new subclass name in the Owning Class field. AionDS automatically updates parent and child links with the new tool bar instance name.

5 In the Basic Window Styles group, you can specify the tool bar as Disabled or Invisible.

Disabled	You cannot select the tool bar.
Invisible	AionDS temporarily hides the tool bar.

📖 *For more information:* See the *Building GUI Applications: Reference* for details about the different basic window styles.

6 In the Tool bar Styles group, you can specify the position in which the tool bar displays:

Snap Horizontal	The tool bar displays horizontally across the top of the window. The tool bar's width is the same width as the window. This is the default.
Snap Vertical	The tool bar displays vertically along the side of the window. The tool bar's height is the same as the window's height.
Floating	The tool bar can be moved anywhere on the parent window when creating the object statically. The floating tool bar remains stationary at runtime.

📖 *For more information:* See the *Building GUI Applications: Reference* for details about the different styles.

7 Use the Pointer field to select a mouse pointer shape to display when the mouse pointer enters the tool bar.

📖 *For more information:* See "Changing the mouse pointer in a window" on page 6-113.

8 To modify the values of slots owned by the tool bar's parent class, click the User Slots dialog and enter the appropriate information.

📖 *For more information:* See "Changing user slot values" on page 6-101.

9 To define the background color of your tool bar, click Color. The Select Background and Foreground Colors window displays. Enter the appropriate changes.

📖 *For more information:* See "Changing window background and text colors" on page 6-115.

10 Click OK. AionDS applies the changes and the tool bar displays.

Building tools on a tool bar

Overview

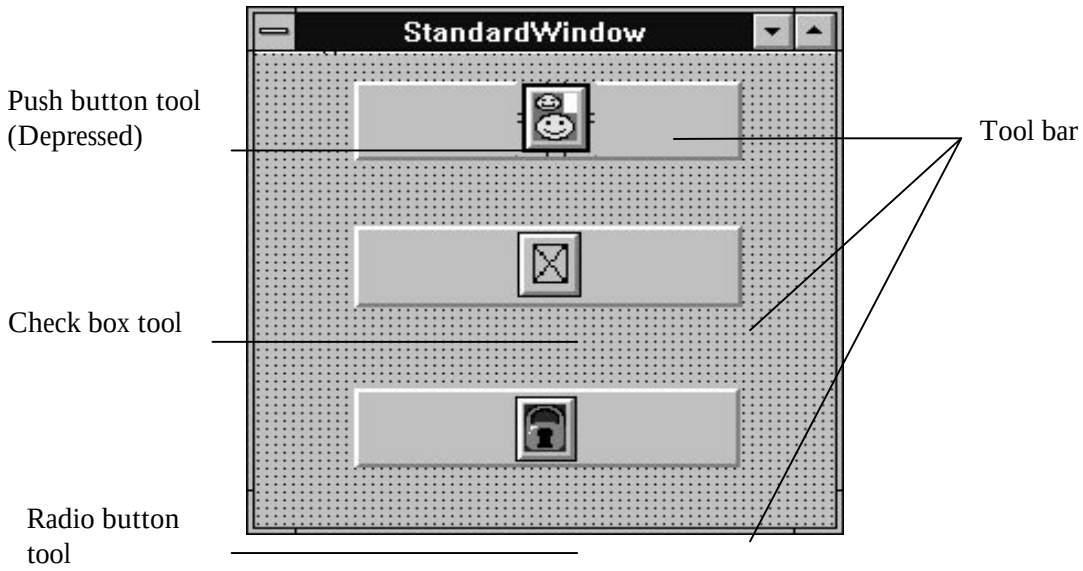
A **tool** is a graphical button you can click to execute an action or set the status of an option. You can only place a tool on a tool bar.

A tool can represent three kinds of behavior:

- push button
- radio button
- check box

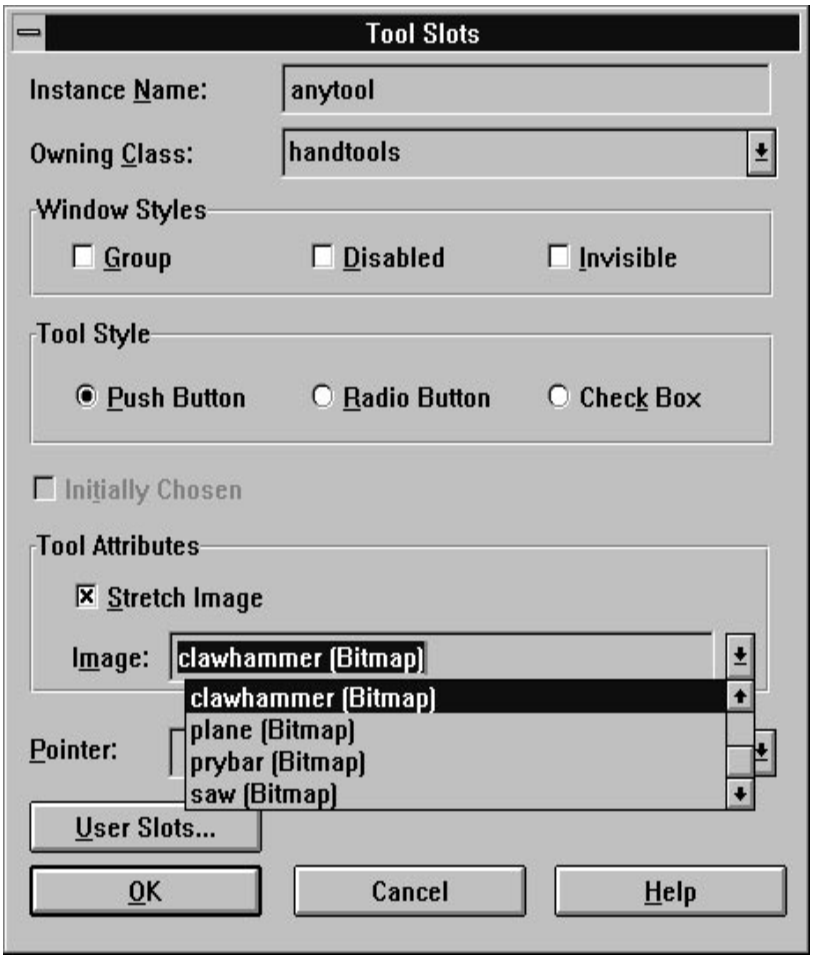
Example

The following is an example of the three kinds of tools built on a tool bar:



What to do

1. Create and resize a tool.
2. Double-click on the tool. The Tool Slots dialog displays.



3 To rename the tool instance, enter a new name in the Instance Name field. AionDS automatically updates parent and child links with the new tool instance.

4 You can assign a tool instance to another tool subclass by entering a new subclass name in the Owning Class field. AionDS automatically updates parent and child links with the new tool instance.

5 In the Window Styles group, you can specify the tool as Group, Disabled, or Invisible.

Group	attaches a tool to a group.
Disabled	the tool selection is disabled. You cannot select the tool.
Invisible	AionDS temporarily hides the tool.

📖 *For more information:* See the *Building GUI Applications: Reference* for details about the different window styles.

6 In the Tool Styles group, you can specify the tool's logic: Push button, Radio Button, or Check Box.

push button	executes a piece of logic.
radio button	represents mutually exclusive condition.
check box	provides a Boolean value.

🔔 *Reminder:* The tool styles determine the tool's behavior, not its appearance.

📖 *For more information:* See the *Building GUI Applications: Reference* for details about the different tools.

7 If you select a radio button or check box from the Tool Styles group, you can select the Initially Chosen check box. AionDS displays the tool as selected when its application opens.

8 In the Tool Attributes group, select the Stretch Image check box to stretch the image to fit the entire push button area.

📖 *For more information:* See the *Building GUI Applications: Reference* for details about the STRETCHIMAGE style.

9 From the Image pull-down menu, select a bitmap instance to display on the tool.

10 Click OK. The image displays on the tool and stretches to fit its dimensions.

📖 *For more information:* See "Linking bitmaps, icons, and mouse pointers to instances" on page 6-108.

11 Use the Pointer field to select a mouse pointer shape to display when the mouse pointer enters the tool area.

📖 *For more information:* See "Changing the mouse pointer in a window" on page 6-113.

12 To modify the values of slots owned by the tool's parent class, click the User Slots dialog and enter the appropriate information.

📖 *For more information:* See "Changing user slot values" on page 6-101.

13 Select a method and add a function to the tool.

📖 *For more information:* See "Selecting methods and adding functions" on page 6-97.

Displaying bitmap windows

Overview

A **bitmap window** defines a location and area in which a bitmap instance displays. When you create a bitmap window or dialog box in a standard window and identify the bitmap instance to display, the bitmap instance links to a bitmap graphic. The only type of input that bitmap windows accept is moving the bitmap with scroll bars, so there is no processing for you to define.

You can also display bitmaps as graphical images on push buttons and tools.

📖 *For more information:* See "Building push buttons" on page 6-74, and "Building tools on a tool bar" on page 6-62.

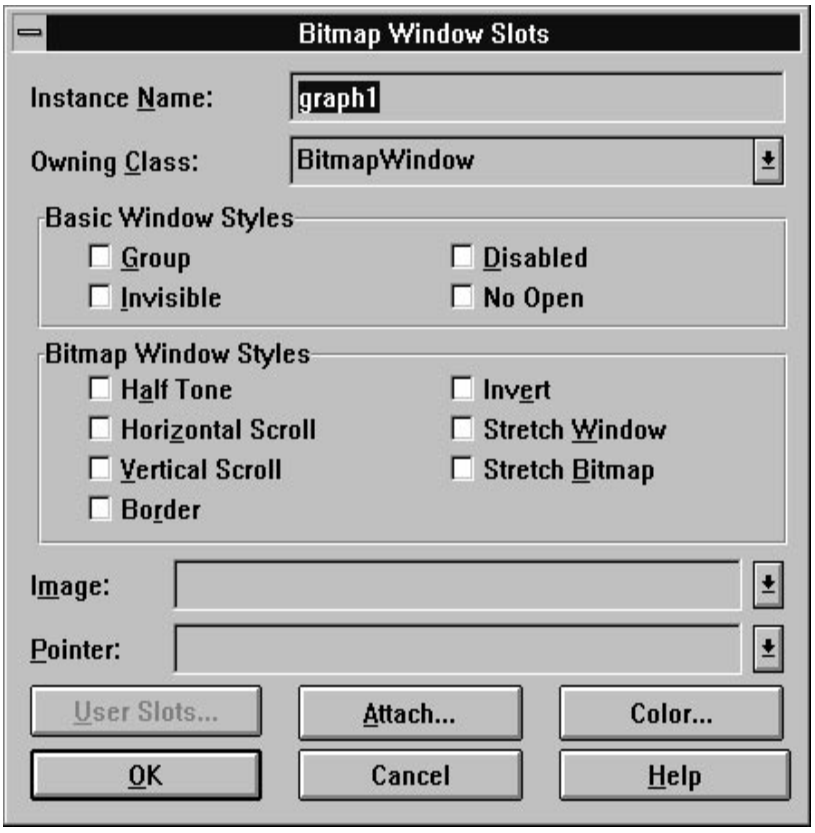
Creating a bitmap graphic

To display graphics in AionDS, the graphics must be in a .BMP format. To create a bitmap graphic, you can use any software program that creates graphics in a bitmap (BMP) format.

What to do

1. Create the bitmap graphic to display.
2. Link the bitmap graphic to an instance.

   📖 *For more information:* See "Linking bitmaps, icons, and mouse pointers to instances" on page 6-108.
3. Create and resize a bitmap window.
4. Double-click on the bitmap window. The Bitmap Window Slots dialog displays.



5 To rename a bitmap window instance, enter a new name in the Instance Name field. AionDS automatically updates parent and child links with the new bitmap window instance.

6 You can assign a bitmap window instance to another bitmap window subclass by selecting a new subclass name in the Owning Class field. Assigning the bitmap window instance to a different class in the Owning Class field automatically updates parent and child links.

7 In the Basic Window Styles group, specify whether the bitmap window is initially enabled or visible, and whether it automatically opens when its parent window opens.

📖 *For more information:* See the *Building GUI Applications: Reference* for details about the different styles.

8 In the Bitmap Window Styles group, define the display properties of the bitmap window, such as stretching the bitmap to fit the window, inverting the colors in the image, adding scroll bars that allow the user to scroll the image, and so on.

   📖 *For more information:* See the *Building GUI Applications: Reference* for details about the different styles.

9 In the Image field, select the bitmap image instance. Link the bitmap image instance to the bitmap image to display in the bitmap window.

10 In the Pointer field, select the mouse pointer shape to display when the mouse pointer enters the bitmap window.

   📖 *For more information:* See "Changing the mouse pointer in a window" on page 6-113.

11 To modify the values of slots owned by the bitmap window's parent class, click the User Slots dialog and enter the appropriate information.

   📖 *For more information:* See "Changing user slot values" on page 6-101.

12 To attach an object to the bitmap window, click Attach. The Attach Object dialog displays. Enter the appropriate information.

   📖 *For more information:* See "Attaching parameters, slots, or messages to windows" on page 6-99.

13 Click OK. AionDS applies the changes and the bitmap window displays.

Creating and editing hot regions

Overview

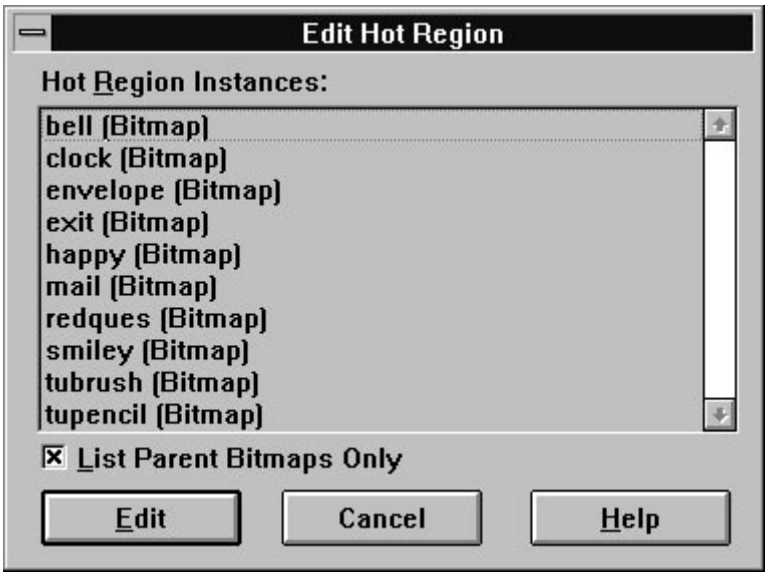
A **Hot region** is a transparent rectangle that you lay over a portion of a bitmap. A Hot Region detects mouse movement and mouse clicks. You can graphically select values by clicking on a region of the bitmap that the Hot Region overlays. It appears that you are clicking on the bitmap image, but you are actually clicking on the Hot Region through which the application processes the input.

If your application displays a city map, for example, you can set up Hot Regions for major points of interest in the city. You select these points by clicking on them. The knowledge base may respond to a click by displaying information about the selected site. The mouse clicks that select Hot Regions process through the When(InstanceName)Chosen demons of the HotRegion class.

To create or edit Hot Regions, first create and link an image file to an instance. You use the Edit Hot Region dialog to display the image and overlay it with Hot Regions. You cannot attach Hot Regions to other objects.

What to do

1 Select Screen.Edit Hot Region. The Edit Hot Region dialog displays.

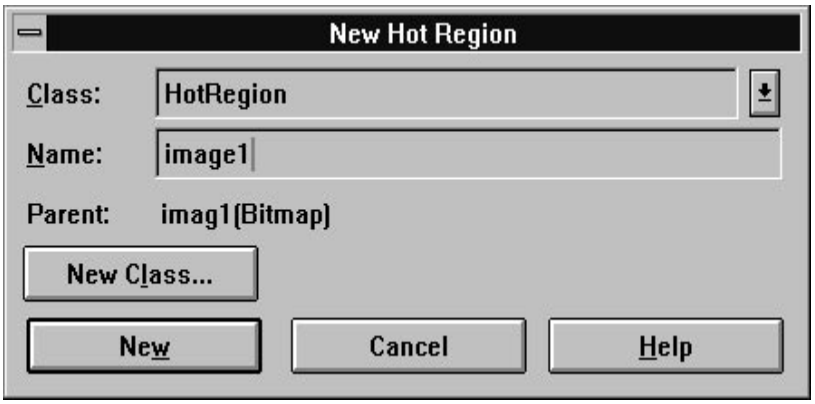


2 In the Hot Region Instances field, select the image instance to create Hot Regions or that contains Hot Regions to modify.

3 If you know the name of the Hot Region instance but do not know the bitmap instance name, uncheck the List Parent Bitmaps Only check box. The Hot Region instance names display.

4 Click Edit. The selected image instance (or the parent image instance of the Hot Region instance that you selected) displays.

5 To create a Hot Region, select Screen.New Hot Region. The New Hot Region dialog displays.



6 In the Class field, select the class to own the Hot Region instance.

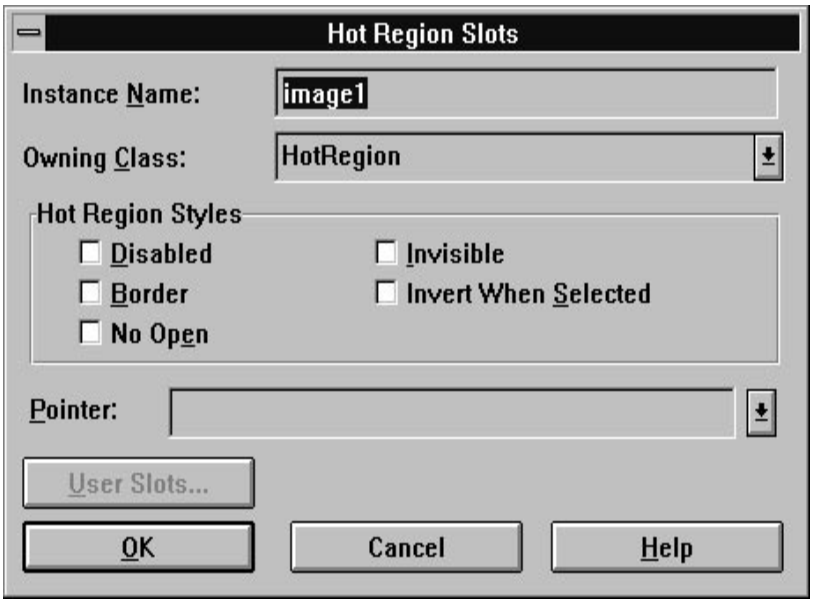
7 In the Name field, enter the name of the Hot Region instance.

8 The Parent field displays the parent window of the Hot Region instance. This is the image instance currently displayed.

9 Click New. You return to the image display and the new Hot Region displays in the image display.

10 Use the mouse to reposition and resize a Hot Region.

11 Double-click on the Hot Region. The Hot Region Slots dialog displays.



12 The Hot Region instance name displays in the Instance Name field. To change the instance name, type over the current name.

13 The parent class of the Hot Region instance displays in the Owning class field. To change the parent class, select a different one.

14 In the Hot Region Styles group, specify whether the Hot Region has visible borders, whether there is a visible response to mouse clicks, and so on.

📖 *For more information:* See the *Building GUI Applications: Reference* for details about the different styles.

15 In the Pointer field, select the mouse pointer shape to display when the mouse pointer enters the Hot Region.

📖 *For more information:* See "Changing the mouse pointer in a window" on page 6-113.

16 To modify the values of slots owned by the Hot Region's parent class, click the User Slots dialog and enter the appropriate information.

📖 *For more information:* See "Changing user slot values" on page 6-101.

17 Click OK.

Result: AionDS applies the changes and the Hot Region editor displays.

18 Select a method and add a function to the Hot Region.

📖 *For more information:* See "Selecting methods and adding functions" on page 6-97.

Deleting a Hot Region

➢ To delete a Hot Region select the Hot Region, then select Edit.Delete GUI window.

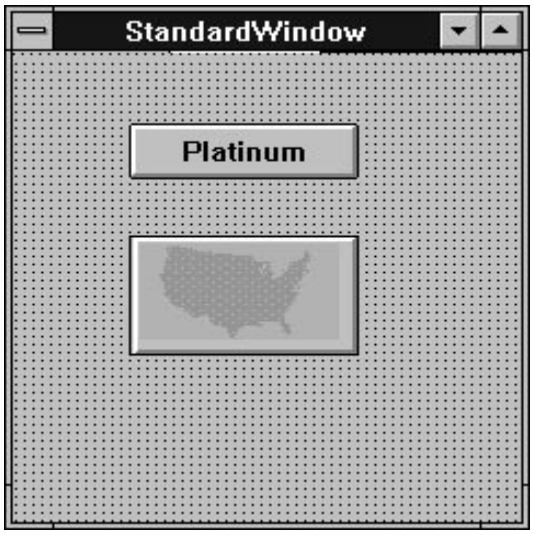
Building push buttons

Overview

A **push button** is a button you can click to execute logic. You can display text or a graphical image on a push button.

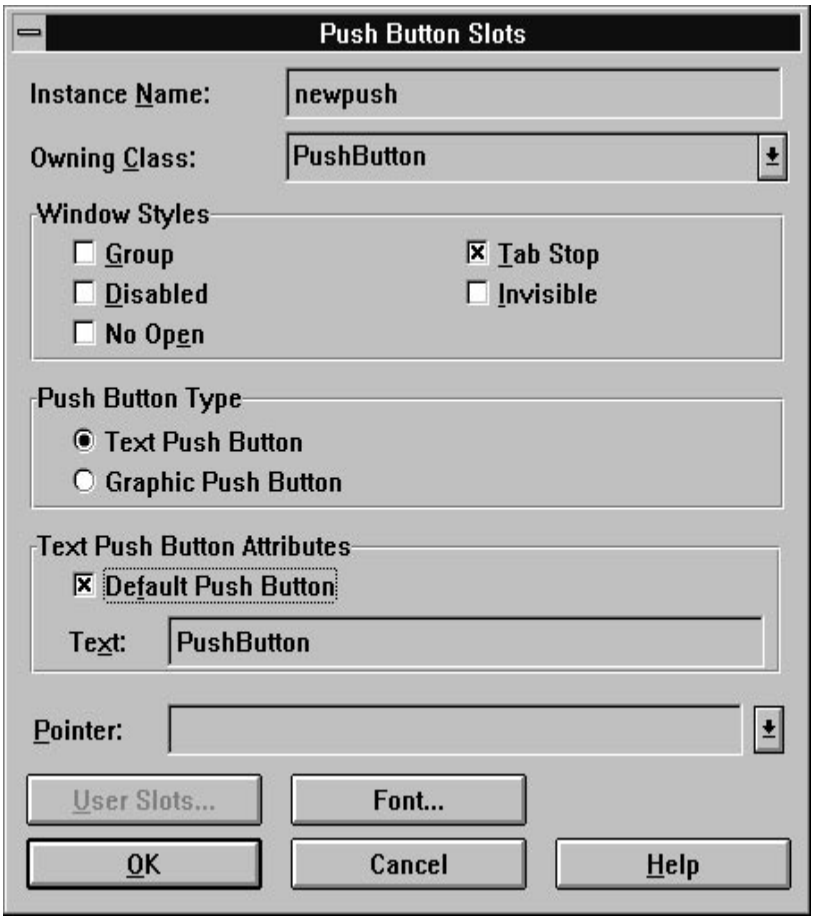
Example

The following example shows textual and graphical push buttons.



What to do

1. Create and resize a push button.
2. Double-click on the push button. The Push Button Slots dialog displays.



3 To rename the push button instance, enter a new name in the Instance Name field. AionDS automatically updates parent and child links with the new push button instance name.

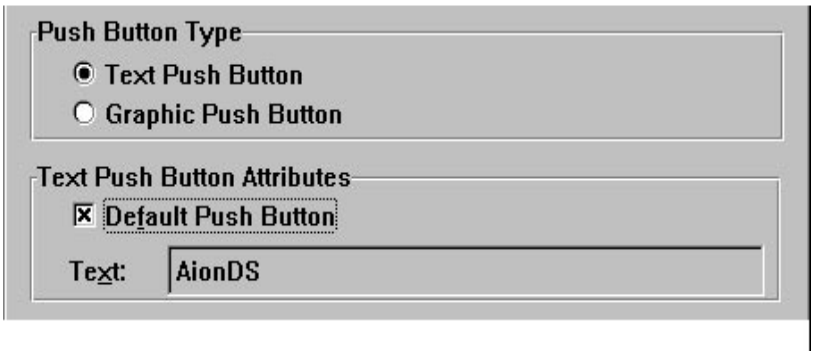
4 You can assign a push button instance to another push button subclass by selecting a new subclass name in the Owning Class field. AionDS automatically updates parent and child links with the new push button instance name.

5 In the Window Styles group, you can specify a push button as part of a group by selecting the Group check box. You can also specify the Tab Stop style by selecting the Tab Stop check box.

📖 *For more information:* See the *Building GUI Applications: Reference* for details about the different styles.

6 In the Push Button Type group, you can select a textual or a graphical push button.

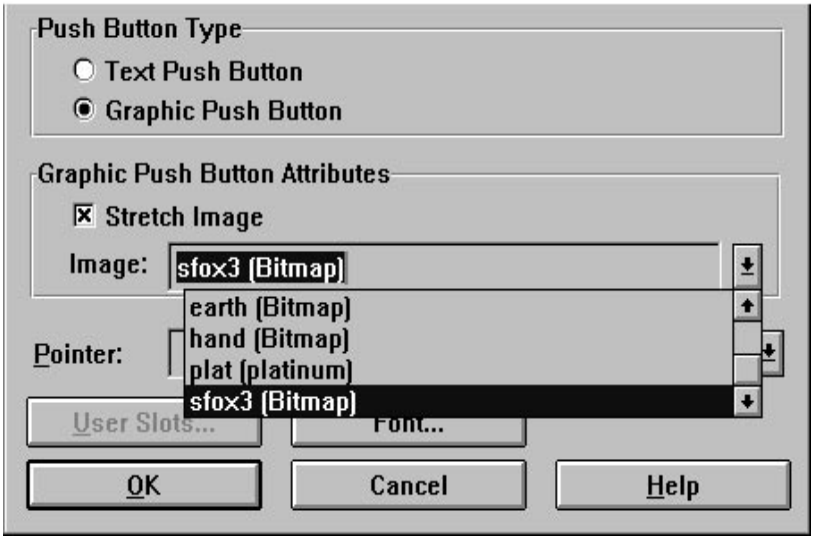
7 If you select a text push button, you can change the text on the push button by using the Text window in the Text Push Button Attributes group. You can enter a mnemonic with the text. The text push button is the default push button.



📖 *For more information:* See "Adding mnemonics to controls" on page 6-106.

✋ *Caution:* You can have only one default push button in a window.

8 If you select a graphic push button in the Push Button Type group, the Graphic Push Button Attributes group box displays.



9 Select the Stretch Image check box to stretch the image to fit the entire push button area.

📖 *For more information:* See the *Building GUI Applications: Reference* for details about the PBS_STRETCHIMAGE style.

10 From the Image pull-down menu, select the bitmap instance to display on the push button.

📖 *For more information:* See "Linking bitmaps, icons, and mouse pointers to instances" on page 6-108.

11 Click OK. The image displays on the push button and stretches to fit the push button's dimensions.

12 From the pull-down menu in the Pointer field, select the mouse pointer shape to display when entering the push button area.

📖 *For more information:* See "Changing the mouse pointer in a window" on page 6-113.

13 To modify the values of slots owned by the push button's parent class, click the User Slots dialog and enter the appropriate information.

📖 *For more information:* See "Changing user slot values" on page 6-101.

14 To change the text font, click Font. The Font dialog displays.

📖 *For more information:* See *Building GUI Applications*: *Reference* for details about fonts.

15 Select a method and add a function to the push button.

📖 *For more information:* See "Selecting methods and adding functions" on page 6-97.

Building icons

Overview

An **icon window** defines a location where an icon instance displays. You create an icon window in a standard window or dialog box and identify the icon instance to display. The icon instance links to an icon graphic file. You cannot change the icon window's size

You can select a method and add a function to an icon window. You can click the icon window to execute an action. It appears that the icon is clicked, but the icon window actually responds to the click. You define the application's response to input in When(InstanceName)Chosen methods.

A number of icon windows, each displaying an icon that represents an action, can be grouped together to create a **tool palette**. The tool palette in the AionDS Window editor was created in this manner.

Creating an icon graphic

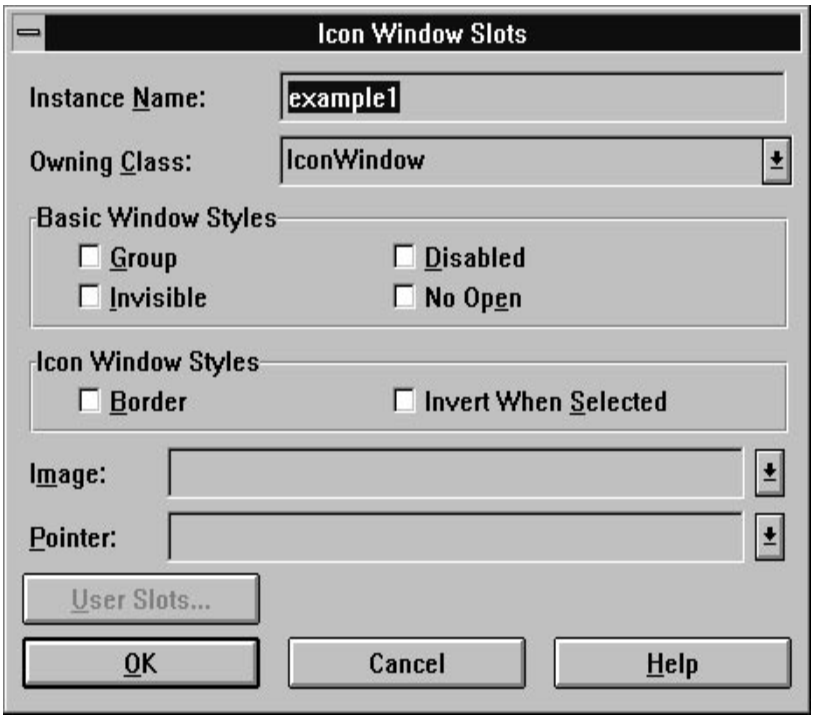
To create an icon graphic, use the tools in the Software Development Kits (SDK) for Windows and OS/2 PM. The icon graphic must be in a Windows or PM ICO format.

What to do

To create an icon and display it in AionDS:

1. Create an icon graphic outside AionDS.
2. Link the icon graphic to an instance.

   📖 *For more information:* See "Linking bitmaps, icons, and mouse pointers to instances" on page 6-108.
3. Create an Icon window.
4. Double-click in the Icon window. The Icon Window Slots dialog displays.



5 To rename an icon window instance, enter a new name in the Instance Name field. AionDS automatically updates parent and child links with the new icon instance name.

6 You can assign an icon instance to another icon subclass by selecting a new subclass name in the Owning Class field. AionDS automatically updates parent and child links with the new icon instance name.

7 In the Basic Window Styles group, you can specify whether the icon window:

- be part of a group
- be hidden temporarily
- cannot be selected until the disabled property is removed
- is not to be opened until the OpenApp command is issued.

📖 *For more information:* See the *Building GUI Applications: Reference* for details about the different styles.

8 In the Icon Window Styles group, specify whether or not the icon window has a border or is inverted.

📖 *For more information:* See the *Building GUI Applications: Reference* for details about the different styles.

9 For the Image field, select the icon image instance. Link the icon image instance to the icon image to display in the icon window.

10 In the Pointer field, select the mouse pointer shape to display when the mouse pointer enters the icon window.

📖 *For more information:* See the "Changing the mouse pointer in a window" section on page 6-113.

11 To modify the values of slots owned by the icon window's parent class, click the User Slots dialog and enter the appropriate information.

📖 *For more information:* See "Changing user slot values" on page 6-101.

12 Click OK. The icon image displays in the icon window.

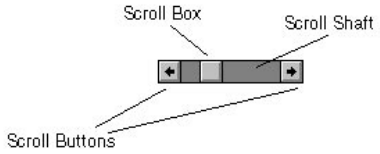
13 Select a method and add a function to the icon window.

📖 *For more information:* See "Selecting methods and adding functions" on page 6-97.

Building scroll bars

Overview

A **scroll bar** is a graphical bar with four main parts, as shown in the following figure:

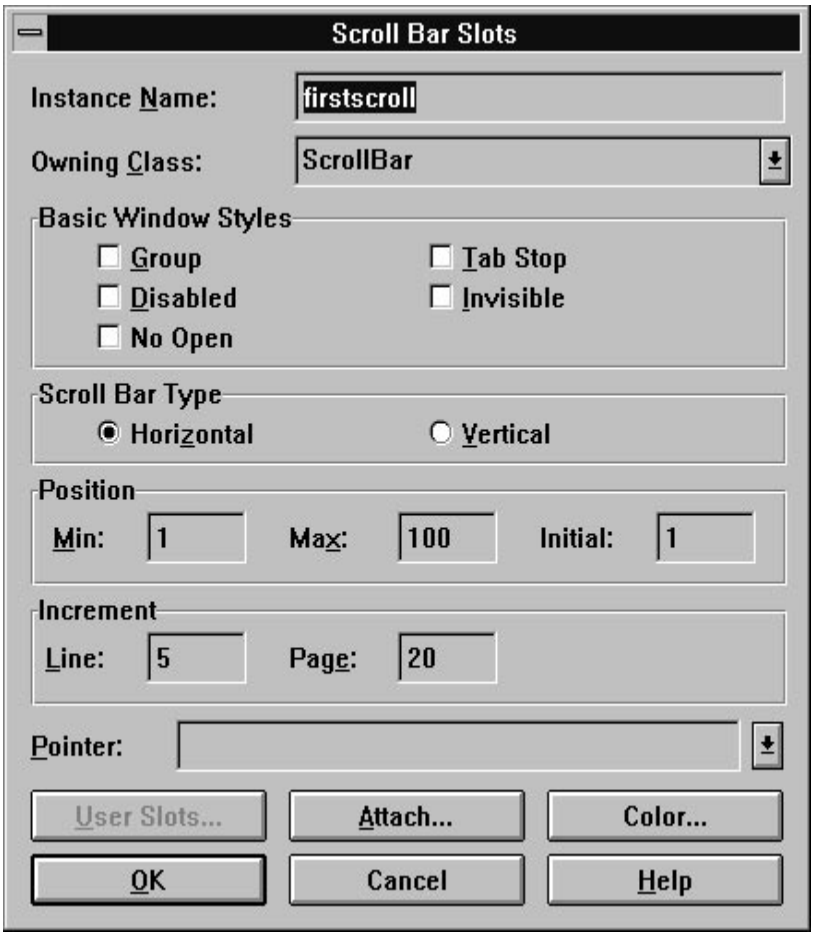


You can move the scroll box within the scroll shaft to display or input a range of values. You can move the scroll box by clicking on the scroll buttons, clicking on the scroll shaft, or dragging the scroll box. You can use a scroll bar as an independent control. You use an independent scroll bar to set values by changing the scroll position. It can be attached to a knowledge base object. Independent scroll bars are useful for displaying or sourcing integer, date, or time values that can fall anywhere within a known range. This section shows you how to build scroll bars as independent controls.

A scroll bar can also display as an integral part of another control, such as a text window, list box, or combo box. A scroll bar that is integrated with another control is considered a feature of the primary control and is discussed in the section that covers the control.

What to do

1. Create and resize a scroll bar. You can change the length of the scroll bar, but you cannot change its width.
2. Double-click on the scroll bar. The Scroll Bar Slots dialog displays.



3 To rename a scroll bar instance, enter a new name in the Instance Name field. AionDS automatically updates parent and child links with the new scroll bar instance name.

4 To assign a scroll bar instance to another scroll bar subclass, enter a new subclass name in the Owning Class field. AionDS automatically updates parent and child links with the new scroll bar instance name.

5 In the Basic Window Styles group, specify whether the scroll bar is initially enabled or visible and whether it is opened automatically when its parent window is opened.

📖 *For more information:* See *Building GUI Applications: Reference* for details about the different styles.

6 In the Scroll Bar Type group, define the type of scroll bar: vertical or horizontal.

7 In the Position group, set the minimum, maximum, and initial positions for the scroll box. Valid values can be integers from 0 to 32,767, inclusive.

8 In the Increment group, specify the following:

Line	How far the scroll box moves when you click on one of the scroll bar arrows
Page	How far the scroll box moves when you click in the scroll bar area

Valid values can be integers from 0 to 32,767, inclusive.

9 In the Pointer field, select the mouse pointer shape to display when the mouse pointer enters the scroll bar.

📖 *For more information:* See “Changing the mouse pointer in a window” on page 6-113.

10 To attach an object to the scroll bar, click Attach to display the Attach Object dialog and enter the appropriate information.

📖 *For more information:* See “Attaching parameters, slots, or messages to windows” on page 6-99.

11 To modify the values of slots owned by the scroll bar’s parent class, click the User Slots dialog and enter the appropriate information.

📖 *For more information:* See "Changing user slot values" on page 6-101.

12 Click OK. The changes are applied and the scroll bar displays.

13 Select a method and add a function to the scroll bar.

📖 *For more information:* See “Selecting methods and adding functions” on page 6-97.

Building OLE controls and objects

Introduction

An OleControl instance represents an OLE Custom Control (OCX). You can extend the graphical user interface in an AionDS/Win application by adding OCXs purchased from a third-party vendor. For example, you can add documents, charts, and sound effects to your application.

AionDS also provides an OleObject class to represent OLE objects. The OleObject instance represents an OLE object that you can embed in an application. For example, a spreadsheet could be an OLE object that you embed into another application.

📖 *For more information:* For more information about OLE and OLE Custom Controls, check the current titles available from Microsoft Press or other publishers.

AionDS and OLE

Overview

Object Linking and Embedding (OLE) is a data transfer protocol used to extend an AionDS application. You can display information or user interface objects from other third-party applications without extensive API calls. Furthermore, the objects appear to be seamlessly integrated. A common example is embedding a spreadsheet within a word processing application.

You can add an OLE object to a standard window or dialog box. You can use an icon window or bitmap window to display a graphic, but you cannot edit the graphic at runtime.

You can create an empty frame for OleObject and fill in that frame at runtime. For example, in a reporting application, you may insert a spreadsheet, a graphic, or a word processing file into an empty frame.

AionDS represents OLE objects as instances of the OleObject class in the WindowObject class hierarchy. You can create subclasses of OleObject, but it is not required.

By default, the runtime edits to an OLE object or control do not persist after the application closes. To save changes to an OLE object, use the SaveObject method.

Important: Although the OleControl and OleObject classes are part of the WindowObject class hierarchy, OLE support is not available in AionDS/2.

OLE concepts

Linking and embedding

Object linking refers to creating a reference to the external object or application. When the object is updated in its own application, the link also updates the version in the container application.

Object embedding refers to placing a copy of the object physically in the container application. Although the copy can be edited within the container application, it is not updated if the original object changes.

Controls and objects

OLE 1.0 included the ability to link and embed **objects**. An object usually is a graphic representation of one application's file in another application. To interact with the object, the user must start a command, such as Edit. One example is a drawing embedded in a word processing file.

Objects are often executable files, although they also can be dynamic link libraries. AionDS represents objects with the OleObject class.

OLE 2.0 includes support for **OLE Custom Controls** (OCXs). An OCX is similar to an object, but often includes another level of control for the application developer. For example, you can often customize characteristics of the OCX, such as how the OCX responds to user actions. The user can interact immediately with an OCX, without using commands. Unlike an OLE object, an OCX is always active.

OCXs are typically dynamic link libraries. AionDS represents OCXs with the OleControl class.

Visual editing

In OLE 2.0 at runtime, you can interact with an OLE object or OCX directly in the container window rather than using the object's or OCX's native environment. This feature is called visual editing. Visual editing is available only when the OLE object or OCX is active at runtime.

☒ *Exception:* OLE objects developed under OLE 1.0 do not support visual editing.

How AionDS implements OLE

Overview

AionDS provides two classes to support OLE: OleControl and OleObject. This section focuses on OleControl, which you use to represent OCXs.

Using AionDS, you can add an OCX to a standard window or dialog box.

Building an OleControl

Overview

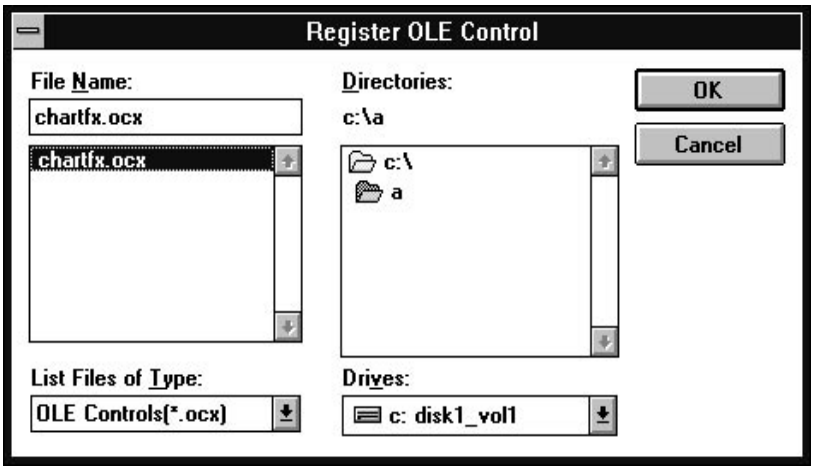
Register the OCX before installing it to a standard window or dialog box. The installation creates the subclass of OleControl for the specific OCX.

Registering OCXs

If you have OCXs that are not in the Registration Database, you can refer to the documentation provided with those OCXs. If you do not have a facility for registering OCXs, you can use AionDS.

To register an OCX using AionDS:

1 Select File.Register OCX. The Register OLE Control dialog displays:



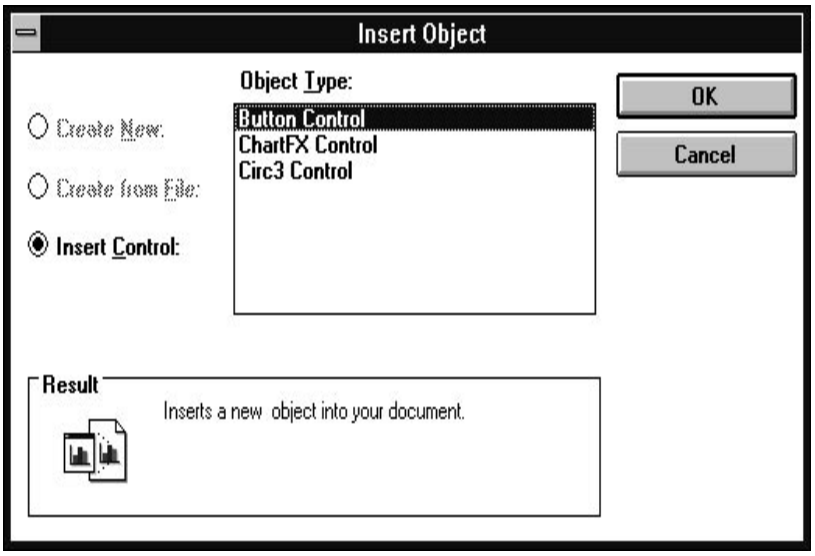
2 Select the drive, directory, and file name for the OCX.

3 Click OK.

Installing an OCX

To install an OCX:

1 Select Screen.Install OCX. The Insert Object dialog displays the OCXs found in the OLE Registration Database.



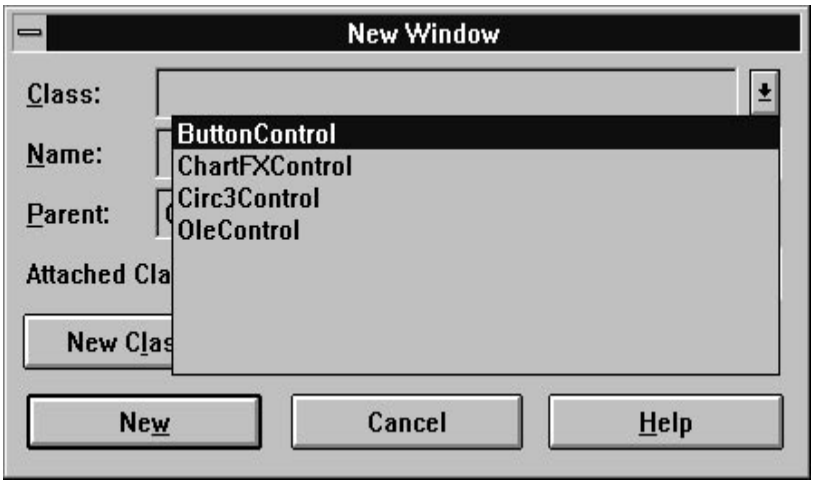
☑ *Tip:* The Create New and Create from File radio buttons are not available because these functions apply to the OleObject class.

2 Select the OCX, then click OK.

Result: AionDS creates a subclass of OleControl for that specific OCX.

Adding the OCX to a window

1 Select the OCX tool from the tool palette, or select Screen.New Window. The New Window dialog displays.



2 Select the subclass for the OCX and specify an instance name.

3 Click New.

Accessing property values

Overview

Unlike other controls, an OCX does not have slots or styles, but its own specific properties. For example, an object that displays only graphics does not have a Font property. You can modify the properties for an OCX in the Window editor or at runtime.

Using the Window editor

1 Double-click on the OCX to open its Properties dialog. The properties you see are specific to the OCX. In this case, the Button Control Properties dialog displays.

2 Enter the caption in the text window

3 Click OK.

Runtime

Keyboard support

AionDS supports all mnemonics and accelerator keys defined by the OCX. If the OCX does not have any mnemonics or accelerator keys, AionDS does not provide a way to define them.

☒ *Exception:* Accelerator keys are not supported for OCXs in dialog boxes.

AionDS does not support tabbing to the OCX because you cannot assign a Tab Stop style to an OCX. Any styles would be represented as properties defined by the OCX.

Similarly, you cannot assign a Default Push Button style to a button-type OCX that supports push button behavior. At runtime, the user cannot press ENTER to select an OCX unless the OCX already has focus. The effect of pressing ENTER also depends on the OCX's definition.

Building an OLE object instance

Overview

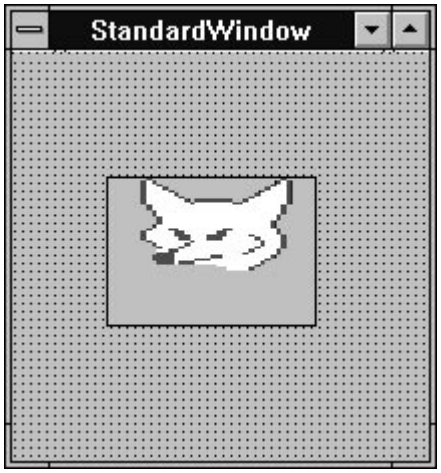
First create a frame or window for the OLE object, then insert a new or existing object. You can also insert an OLE object at runtime.

At runtime you can edit OLE objects to the extent that the OLE application supports editing. You cannot assign runtime input to an attached object.

Important: The OLE object must be in the Registration Database.

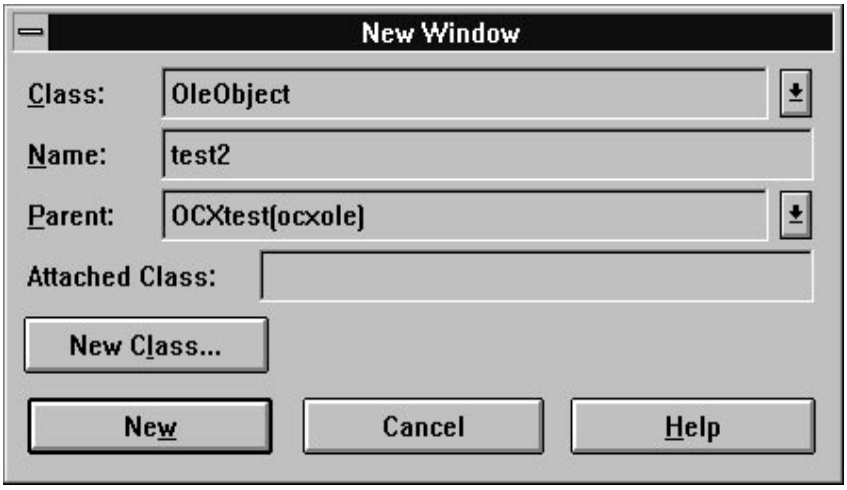
Example

The following is an example of an OLE object:



Creating the frame

1 In the Window editor, select the OLE tool from the tool palette, or select Screen.New Window. The New Window dialog displays.



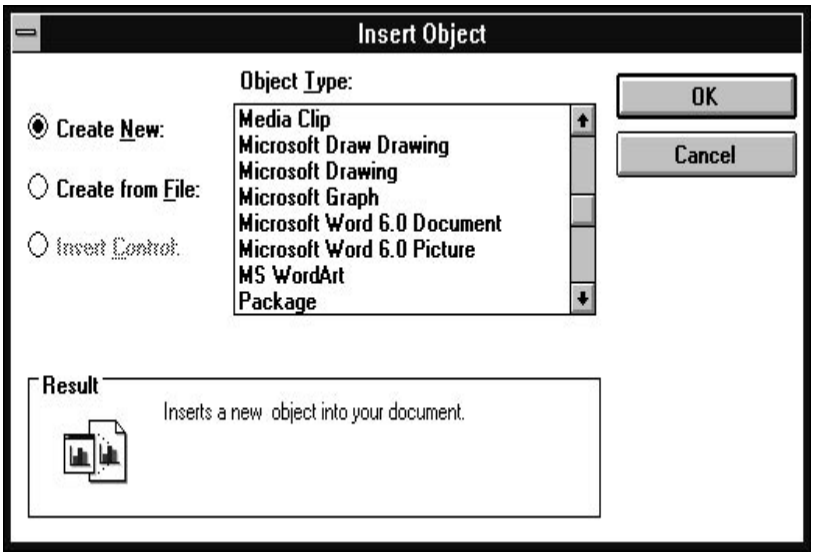
2 Specify the class name and instance name on the New Window dialog.

Result: An empty frame appears on the standard window. The frame, not the OLE object to be inserted, is the OleObject instance.

3 Click OK.

Inserting the object

1 Select Screen.Insert OLE Object. The Insert Object dialog displays.



2 On the Insert Object dialog, select one of the following radio buttons:

Create New	to create a new object. Select the application from the Object Type list box.
Create from File	to insert an existing object. Enter the file name in the File field or use the Browse push button to find the file.

3 Click OK.

Result: The object's application opens. If you select Create New, the application window is empty. If you select Create from File, the file displays for editing.

4 After you create or edit the object, select Update from the File menu; then select Exit.

Result: The object displays in the object frame on the standard window or dialog box.

Empty frames

You might not want to insert an OLE object at edit time. You might not know which object file or which application you need. For example, at runtime you may select a graphic from either a drawing or paint application. You can insert an object by clicking the right mouse button on the frame.

Alternatively, you can insert an OLE object at edit time and select another at runtime. You cannot save runtime changes after the application closes.

When creating empty frames you cannot:

- dynamically insert an object for runtime based on application logic
- save the object inserted at runtime

Runtime

Overview

The OleObject class also handles runtime input differently than the OleControl class. Unlike OleControl, OleObject has no methods or events to customize.

Runtime editing

The most direct way to edit an OLE object during runtime is to double-click on the OLE frame. The object displays in its native editing environment. For example, a spreadsheet object displays in the spreadsheet application in which it was created.

During runtime you can click the right mouse button on the OLE frame. A pop-up menu displays. The first menu item is for inserting OLE objects; the remaining menu items are specific to the current OLE object.

Tool bars

When an OCX or OLE object is active, AionDS displays the OLE application tool bar, if one exists. However, make sure the tool bar does not overlay any other controls in the container window.

Selecting methods and adding functions

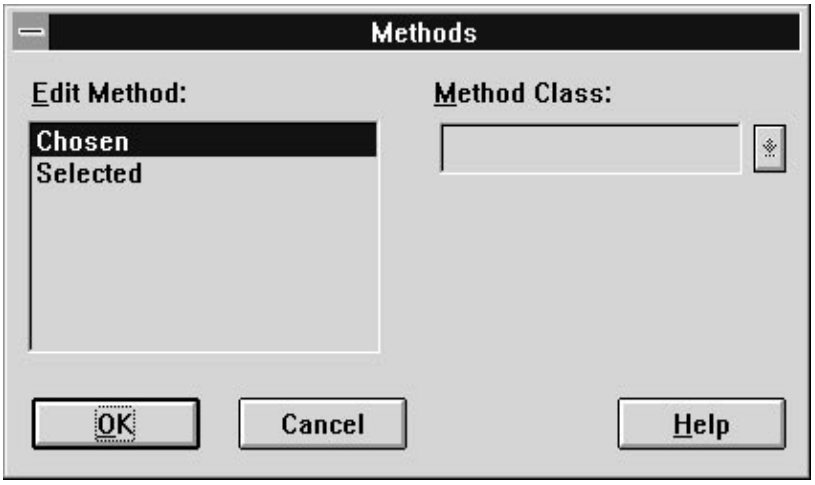
Overview

You can add customized methods and functions to controls. For example, you may select a method and add a function to a menu item to open another window.

What to do

1 To display the Methods dialog for a window, double-click the right mouse button on the window.

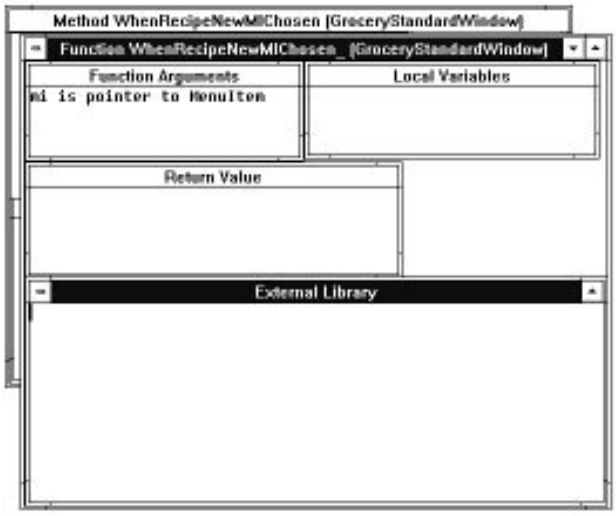
To display the Methods dialog for a menu item, select the menu item with the mouse.



2 In the Edit Method group, select the method to modify.

☑ *Tip:* Only certain methods are available for each object.

3 Click OK. The Methods and Function dialogs display.



4 Make any desired modifications to the function and method.

5 Save and close the function and method by selecting Object.Close. AionDS saves and closes the function and method.

Result: The method and attached function execute when the execution conditions for the window occur. For example, you select the When*InstanceName*Chosen method for a push button and the function opens another dialog. When you click the push button, the method and attached function executes and a dialog opens.

Attaching parameters, slots, or messages to windows

Overview

You can use the Attach push button available in most slots dialogs to attach a parameter, slot, or message to a window. For example, you can attach a parameter to a text window to store user input to that text window.

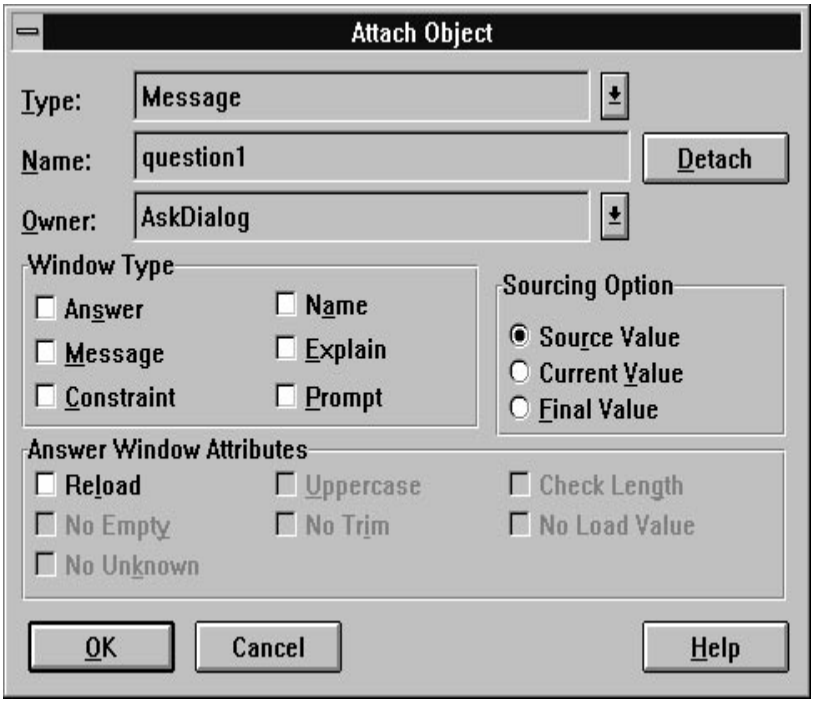
📖 *For more information:* See the *Building GUI Application: Reference* for details about attaching parameters, slots, or messages to a specific window.

What to do

1. In the window slots dialog, if you select the TSL or Attached Object radio button, the Attach push button is highlighted.

   If you select the TSL radio button, you can only select the Reload check box in the Attach Object dialog.

2. Click Attach. The Attach Object dialog displays.



3 In the Type field, specify the object type—slot, parameter, or message—to attach.

4 In the Name field, specify the name of the object to attach.

5 In the Owner field, specify the name of the class that owns the object to attach.

6 In the Window Type group, select the type of window to assign to your window. Each window type maps the window to an object property. For example, selecting Answer provides an area for you to input values.

7 In the Sourcing Option group, select one of the following types of processing to resolve variables.

Options	The object resolves processing:
Source value	by backward chaining for parameters that do not have a value and placing those parameters on a pending list if required
Current value	without backward chaining and uses the current parameter value—even if it is unknown
Final Value	by backward chaining for all unprocessed parameters

8 In the Answer Window Attributes group, select one or more of the options to set the characteristics of the Answer Window type.

If you did not select Answer for the Window Type, only the Reload option is available.

9 Click OK.

Result: The specified object attaches to the window.

Changing user slot values

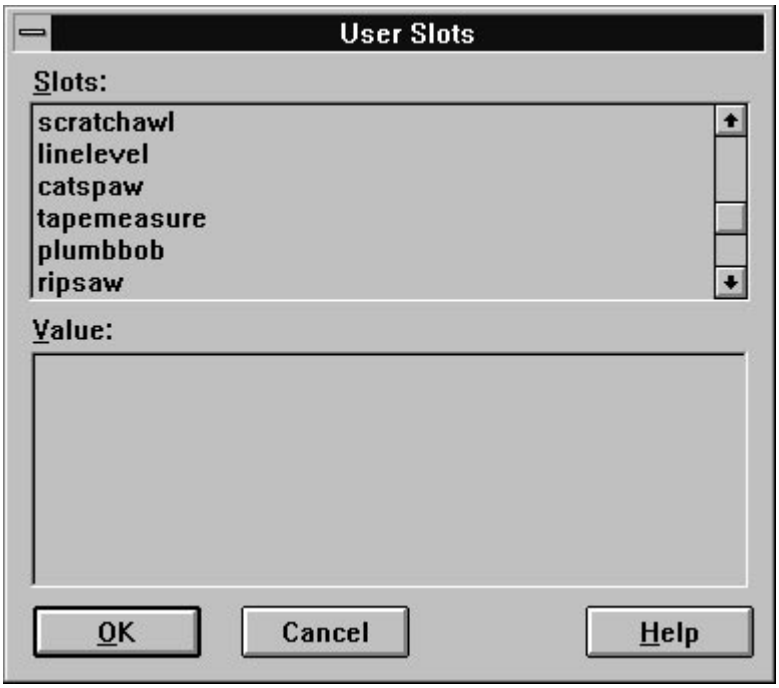
Overview

Use the User Slots push button, which is available in most slots dialogs, to change the values of the slots associated with a window. Depending on which window is open during runtime, you can set different initial values for the same slot.

What to do

1 In the window slots dialog, click User Slots. The User Slots dialog displays.

☑ *Tip:* The User Slots dialog is only available when the window's parent class owns slots. Create user slots using Object.New.



2 In the Slots field, select the slot that contains the value to change.

3 In the Value field, enter the new value to assign to the slot.

4 Click OK.

Organizing controls in tabbing sequences and groups

Overview

You can set and change the tabbing order of child windows and organize child windows into groups by using the Order Children dialog.

At runtime, you can press the Tab key to move the cursor from one child window to the next in the tabbing order. A **tabbing order** is the sequence in which you move the cursor (by pressing the Tab key) to each child window.

- Press the Right or Down arrow keys to move the cursor from the current child window to the next child window in the group.
- For radio buttons, press the Right or Down arrow keys to deselect the current radio button and select the next radio button in the group.

Opening the Order Children dialog

➢ Select Screen.Order Children.

Result: The Order Children dialog displays.

The Parent Windows field automatically displays the name of the parent window from the Window Editor pull-down menu.

The Child Windows field contains the list of child windows belonging to the window whose name displays in the Parent Windows field. Use this field in conjunction with the Tab Stop and Group check boxes to complete the following tasks:

- order the child windows in a tabbing sequence
- organize the child windows into groups

Groups and tabbing orders are independent from each other. You could have overlapping groups and tabbing orders.

Ordering child windows in a tabbing sequence

To order child windows in a tabbing sequence,

- define tab stops for each child window in the sequence
- list the child windows in the desired order.

Defining tab stops

To define a tab stop for a child window:

1. Select the windows to define tab stops.
2. Select the Tab Stop check box.

Result: Tab stops are defined for the windows.

Setting the tabbing order

To set the order of the child windows listed in the Child Windows field:

1. Select one or more child window instances in the list.
2. Place the pointer between the child windows to insert the selected instances. When you place the pointer between child windows, it changes to an insertion pointer.
3. When the insertion pointer displays, click the mouse. AionDS inserts the selected instances in the new position.

Result: The child window list is now in the defined tabbing order. Once you click Order and save the object, the tabbing order is set.

When the user opens a top-level window during an active session, the cursor stops at the first child that has the Tab Stop style. If no child windows have the Tab Stop style, the cursor stops on the first child window.

Grouping child windows

To organize child windows in a group, include each child window in the group and list the child windows in the desired order.

Including child windows

To include child windows in a group, complete the following steps:

1. Select the windows to include in the group.
2. Check Group.

Setting the group order

To set the order of the child windows in a group listed in the Child Windows fields, complete the following steps:

1. Highlight one or more child window name instances in the group.
2. Move your cursor to the place in the group where you want to move the an highlighted instances. As the cursor moves between items, its image changes.
3. Click on the space between items while the cursor is displaying the altered cursor image, and the instances you highlighted are inserted in the new position.

Result: The group is now in the defined order. Once you click Order and save the application, this group order is set.

When you press the Right or Down Arrow keys on the last radio button in the group, the first radio button in the group is selected.

Adding mnemonics to controls

Overview

A **mnemonic** is the ALT key and a letter key. Press the ALT key and a letter key simultaneously to move the cursor to the control. The letter used in the mnemonic is underlined.

You can create mnemonics for the following controls:

- label windows
- push buttons
- radio buttons
- check boxes
- group boxes
- menu items

What to do

➢ To create a mnemonic for a control, insert a tilde (~) in front of the letter to make a part of the mnemonic in the Text field of the control's slots dialog.

Mnemonics for label windows

When you issue the mnemonic for a label window, the cursor is not moved into the label window. The cursor is moved into the next window with the Tab Stop style, as shown in the Order Children dialog. You can change the order of the controls in the Order Children dialog.

📖 *For more information:* See "Organizing controls in tabbing sequences and groups" on page 6-103.

Mnemonics for group boxes

When you issue the mnemonic for a group box, the cursor is moved to the first child window of the group box that has the Tab Stop style.

📖 *For more information:* See "Building group boxes" on page 6-52. See the Group Box chapter in the *Building GUI Applications: Reference*.

Linking bitmaps, icons, and mouse pointers to instances

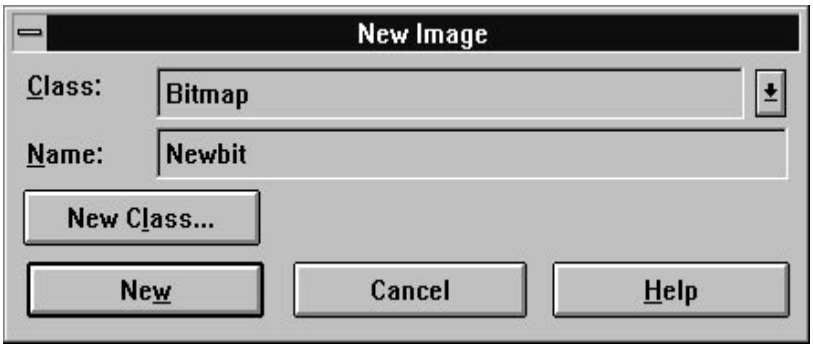
Overview

You can link a bitmap, icon, or mouse pointer graphic outside of AionDS to an instance. An instance linked to a graphic is called an **image instance**. To display the graphic in AionDS, link the image instance to the bitmap or icon window, or specify the image instance in the Pointer field in the window's slots dialog.

You use the New Image dialog to create an image instance to link to the graphic. Use the Image Slots dialog to link the image instance to the graphic file.

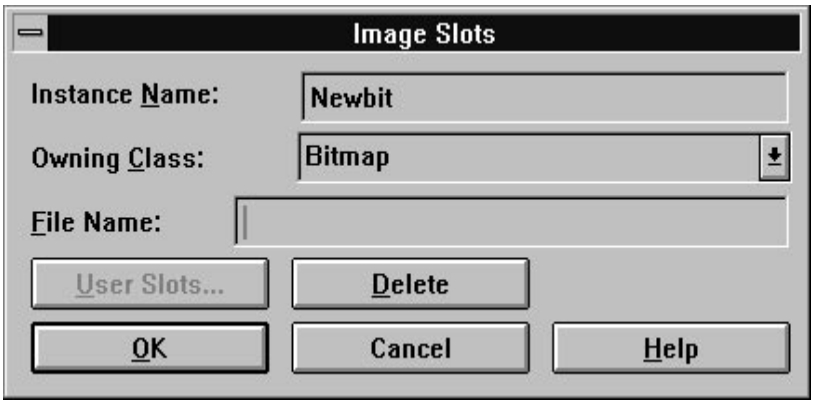
What to do

1. Create the bitmap, icon, or mouse pointer from outside of AionDS.
2. Select Screen.New Image. The New Image dialog displays.



3. In the Class field, select the class to own the instance. The class must match the type of graphic file that will be linked to the instance. The class can be bitmap, icon, or mouse pointer, or a subclass of one of these.
4. In the Name field, enter a name for the instance.

5 Click New. The Image Slots dialog displays.



6 To rename an image instance, enter a new name in the Instance Name field. AionDS automatically updates parent and child links with the new instance name.

7 You can assign an image instance to another image subclass by entering a new subclass name in the Owning Class field. AionDS automatically updates parent and child links with the new image instance name.

8 In the File Name field, enter the path and graphic file name to which you want to link this instance.

9 To modify slot values owned by the instance's parent class, click User Slot to display the User Slots dialog and enter the appropriate information.

📖 *For more information:* See "Changing user slot values" on page 6-101.

10 Click OK. The instance is created and the graphic file is linked to the instance.

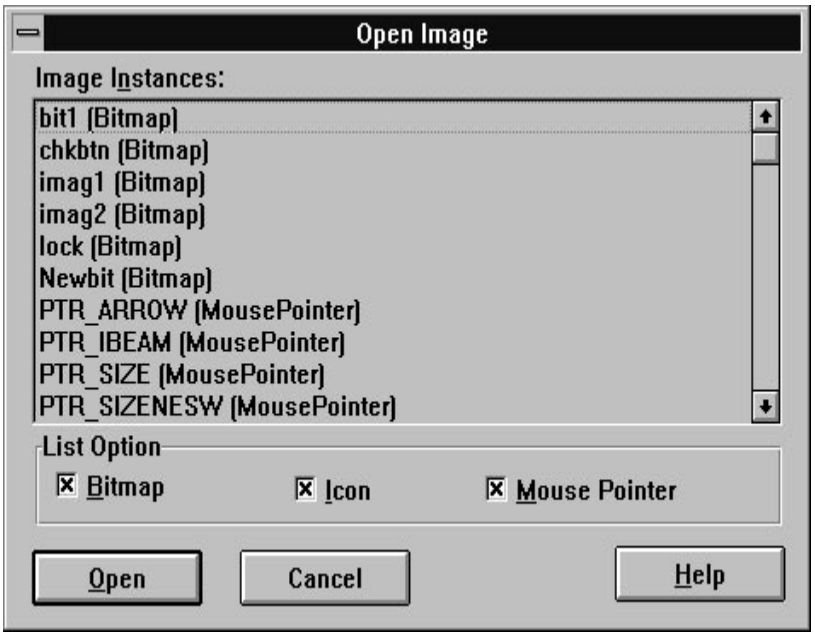
Changing instances linked to bitmaps, icons, or mouse pointers

Overview

You can change the name of an image instance, parent class of an image instance, and file name of the bitmap, icon, or mouse pointer that is linked to an image instance.

What to do

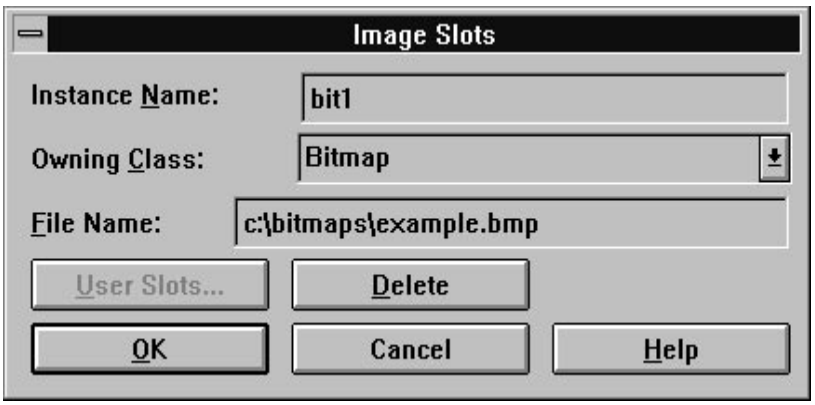
1 Select Screen.Open Image. The Open Image dialog displays.



2 Select the image instance from the Image Instances list.

3 To limit the list of image instances according to the type of graphic—bitmaps, icons, mouse pointers—select the appropriate check boxes in the List Option group.

4 Click Open. The Image Slots dialog for the selected image instance displays.



5 To rename an image instance name, enter a new name in the Instance Name field, and the file name and path of the graphic file in the File Name field. AionDS automatically updates parent and child links with the new image instance name.

6 You can assign an image instance to another image subclass by selecting a new subclass name in the Owning Class field. AionDS automatically updates parent and child links with the new image instance name.

7 You can change the slot values owned by the instance's parent class. Click User Slot to display the User Slots dialog and enter the appropriate information.

📖 *For more information:* See "Changing user slot values" on page 6-101.

8 Click OK. The instance changes and the specified graphic file links to the instance.

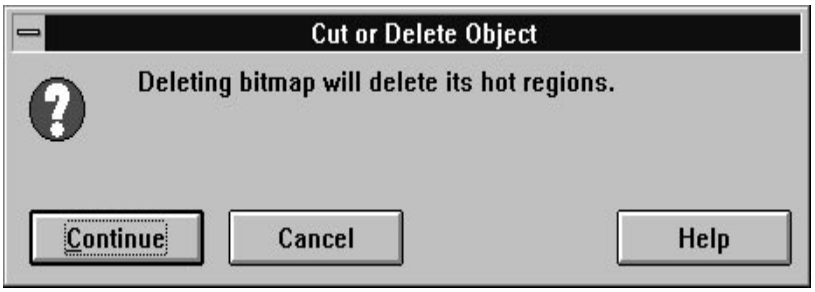
Deleting instances linked to bitmaps, icons, or mouse pointers

Overview

To delete an instance linked to a bitmap, icon, or mouse pointer file, open the Image Slots dialog and click Delete.

What to do

1. Select Screen.Open Image. The Open Image dialog displays.
2. Select the image instance from the Image Instances list.
3. Click Open. The Image Slots dialog for the selected image displays.
4. Click Delete. A confirmation pop-up window displays.



5. To delete the instance, click Continue. You return to the AionDS primary window.
6. To return to the Image Slots dialog without deleting the instance, click Cancel.

Changing the mouse pointer in a window

Overview

When the mouse pointer enters a window, it changes to the specified mouse pointer image. You can change the shape of the mouse pointer to indicate the type of action that the user performs. For example, when the user is required to enter a name into a text window, the mouse pointer changes to an I-beam by default.

You can change a mouse pointer shape to one already included in AionDS, or you can create a new mouse pointer shape. To create a mouse pointer shape, first create a mouse pointer graphic outside AionDS. Link the mouse pointer graphic to an instance, and then specify the instance in a window slots dialog.

Creating a mouse pointer image file

A mouse pointer image must be in one of the following file formats:

Operating system	Supported file format
Windows	CUR
OS/2	PTR

To create a mouse pointer image, use the Software Development Tool kit that comes with Windows or OS/2. You can also use any utility that can create or convert files to these formats.

What to do

1. Create a mouse pointer graphic from outside AionDS.
2. Link the mouse pointer graphic to an instance.
3. In the window slots dialog, select the appropriate instance name of the Mouse Pointer class or subclass in the Pointer field.
4. Click OK.

Linking minimize icons to windows

Overview

To create minimize icons, create an icon image file from outside of AionDS. Link the icon image file to an icon instance. You specify the icon instance for the minimize icon in the window slots dialog's Icon field. The icon image displays when the window is minimized.

📖 *For more information:* See "Building icons" on page 6-79.

What to do

1. Create an icon graphic from outside of AionDS.
2. For the Icon field in the window slots dialog, select the icon instance to display when the window is minimized in the Icon field.
3. Click OK.

Changing window background and text colors

Overview

Use the Select Background and Foreground Colors dialog to define a background or text (foreground) color of a window.

3-D restrictions

If you display windows in 3-D at runtime, the background color defaults to gray.

If you change the background default color in the Select Background and Foreground Colors dialog, turn off the 3-D option in the Display Settings dialog. Uncheck At run time in the Use 3-D controls group to display that color at runtime.

📖 *For more information:* See *Building GUI Applications: Reference* to learn more about the 3-D option.

Available container colors

You can define background and some text (foreground) colors for most containers.

Container	Background colors	Text colors
Ask Dialog	yes	no
Dialog Box	yes	no
Show Dialog	yes	no
Standard Window	yes	no

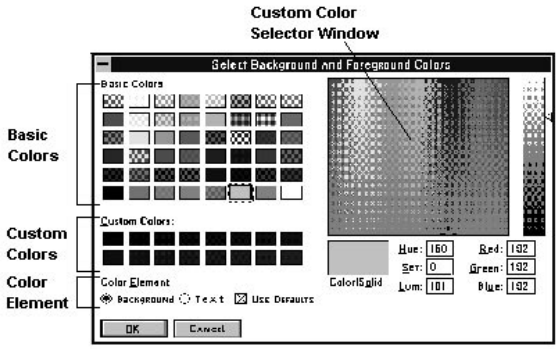
Available control colors

Background and text (foreground) colors are available for most controls.

Control	Background colors	Text colors
Bitmap Window	yes	no
Check Box	yes	yes
ColumnLB	yes	yes
Combo Box	yes	yes
Group Box	yes	yes
Icon Window	no	no
Label Window	yes	yes
List Box	yes	yes
OLE Control	no	no
OLE Object	no	no
Push Button	no	no
Radio Button	yes	yes
Scroll Bar	yes	no
Text Window	yes	yes
Tool	no	no
Tool bar	yes	no

What to do

1 Click Color in the slots dialog. The Color dialog displays.



2 To define a color other than the system default color, click a Basic color rectangle, Custom Color rectangle, or use the mouse to select a color in the Custom Color Selector window. AionDS displays that color in the Color|Solid display window.

3 Click OK to return to the slots dialog.

4 After completing the slots dialog, click OK. Return to the Window editor and the new color displays for the background or text.

The background colors of child windows inherit color changes made to the parent's background color. To change the background color to a child window, use the Select Background and Foreground Colors dialog.

Using the system default colors

➢ To use the operating system default color, select the Use System Default Color check box.

The first time you open the Color dialog for a window, the default color is set.

Changing fonts

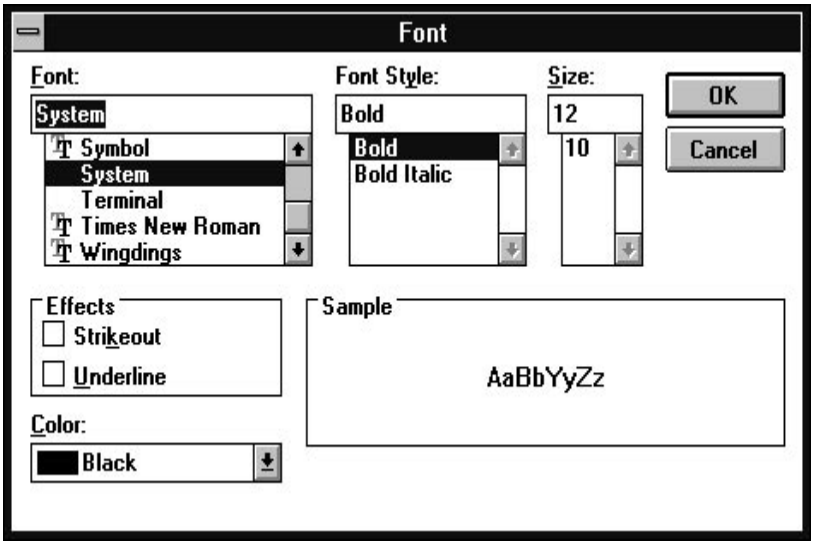
Overview

Use the Fonts dialog to change the font and color of the text displayed in a window. If AionDS cannot find the specified font, AionDS uses the closest available font as determined by the operating system. If the operating system cannot find a font that closely matches the specified font, AionDS uses the system font.

Caution: Do not change the font if your application will run on different platforms.

What to do

1 In the slots dialog, click Font. The Font dialog displays.



2 Select the font, font style, font size, effects, and color of the text.

3 Click OK to return to the slots dialog.

4 After completing the slots dialog, click OK. You return to the Window editor and the new text displays.

📖 *For more information:* See the *Building GUI Applications: Reference* for details about fonts.

Building a DDE system

Overview

Use Dynamic Data Exchange (DDE) to pass data between two applications running on the same PC. The two applications have a **conversation**—one is the client and the other is the server. The **client** application requests information, while the **server** application returns the requested information. For example, the portfolio manager knowledge base may provide recommendations to a report generator application.

An application can have multiple conversations at the same time. For example, an AionDS knowledge base may manage stock portfolios. This knowledge base could request stock prices from an application with direct access to several stock exchanges.

This section describes building static client and server applications:

- **Static** client and server applications are built at edit time

  Use static conversations at edit time when the servers and topics are known. You need to know what kind of data the end user wants and from which application.

  Use static data items when the data is known at edit time. The knowledge base objects necessary to create data items must be static and internal.

- **Dynamic** client and server applications are built at runtime

  If the data resides in a database, dynamically create the data items.

  📖 *For more information:* See *Building GUI Applications: Reference* to learn how to build dynamic client and server applications.

Client

To set up your knowledge base as a client application, use the Client class and Link class. You can also create subclasses of Client and Link and use these.

Server

To set up your knowledge base as a server, use the Topic class or define a subclass of Topic.

DDEObject class hierarchy

The DDEObject class hierarchy contains the following classes:

AionDS as the client

Overview

When AionDS is the client application, the DDEObject and WindowObject classes are closely related. A conversation is defined for a specific window. That window must be an instance of either a Standard Window or a Standard Window subclass. The conversation is only active while that window is open.

Determine how the application starts conversations. For example, you can initiate a conversation when selecting an OK push button or when opening a window. You can also define controls to display data from the server.

Creating client conversation instances

This section describes how to create the Client instance and set up knowledge base logic to start the DDE conversation

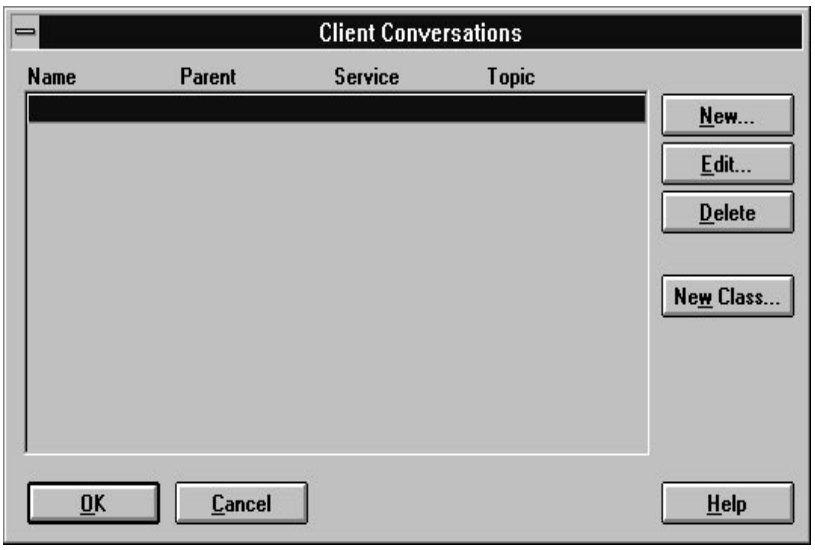
📖 *For more information: Building GUI Applications: Reference* for details about the logic used to start the DDE conversation.

What to do

In the Window editor, open the window for which to establish a DDE conversation.

Displaying the New Conversation dialog

1 Select Screen.DDE.Conversation. The Client Conversations dialog displays.

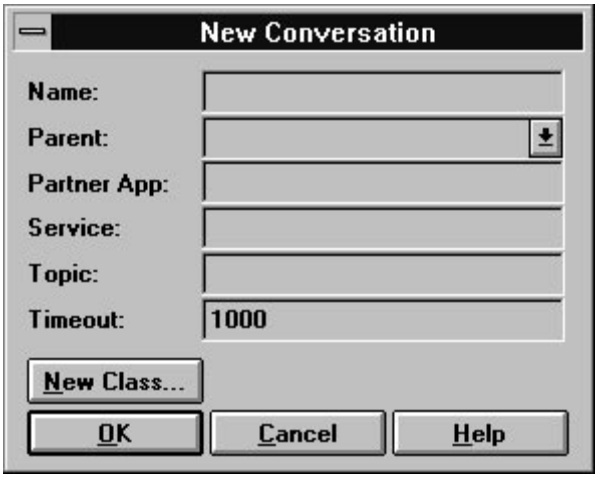


2 To create a new subclass of the Client class, click New Class. The New Class dialog displays.

3 Complete the New Class dialog.

📖 *For more information:* See "Creating windows, dialogs, and controls" on page 6-14.

4 In the Conversations dialog, click New. The New Conversation dialog displays.



Creating the Client instance

In the New Conversation dialog:

1 Specify a conversation name in the Name field.

2 Select the parent class for the Client instance in the Parent field.

3 To create a new parent class for this Client instance, click New Class. The New Class dialog displays.

4 Complete the New Class dialog.

📖 *For more information:* See "Creating windows, dialogs, and controls" on page 6-14.

5 Specify the server's executable file name (without the .EXE extension) in the Partner App field.

6 Specify the service name as defined by the server in the Service field.

7 Specify the topic name in the Topic field.

8 Specify the time-out period in the Time-out field. You can enter a value between 0 and 64,000 milliseconds. The default is 1000 milliseconds.

9 Click OK.

Modifying conversations

Use the Client Conversations dialog to edit or delete existing Client instances.

editing a conversation

1 select the conversation name and click Edit. The Edit Conversation dialog displays.

2 The Edit Conversation dialog contains exactly the same fields as the New Conversations dialog. Change the desired fields and click OK.

Result: The conversation fields change and you return to the Client Conversation dialog.

deleting a conversation

1 select the conversation name and click Delete. A confirmation window displays.

2 Click Delete, to delete the conversation. The conversation is deleted and you return to the Client Conversation dialog.

3 Click Cancel to cancel the deletion. The conversation is not deleted and you return to the Client Conversation dialog.

Creating link instances

This section describes how to create a Link instance. A **link** provides a connection between a knowledge base object in the client and data in the server. There are two types of links:

- **warm link** notifies AionDS when the requested data item's value changes
- **hot link** sends the new value when the requested data item's value changes

Also set up knowledge base logic to activate the DDE link.

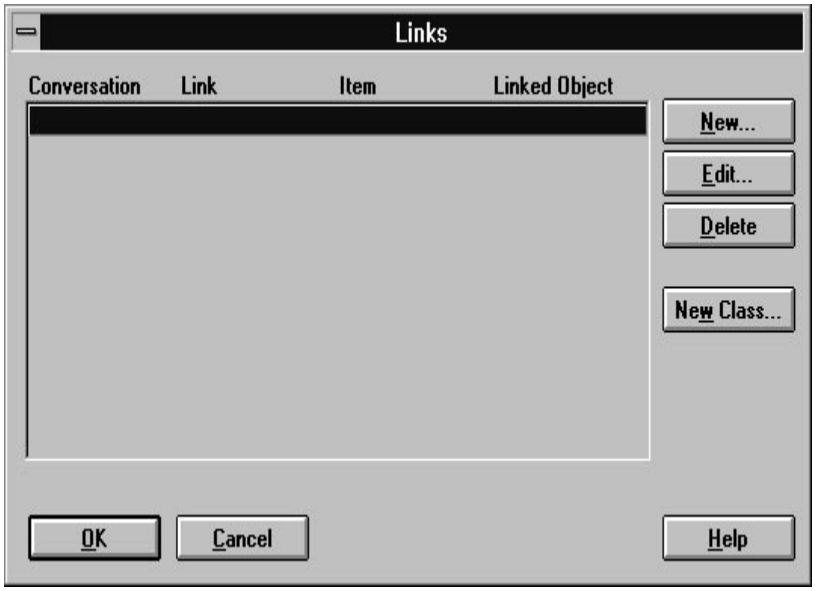
📖 *For more information:* See *Building GUI Applications: Reference* for details about logic used to activate the DDE link.

What to do

In the Window editor, open the standard window to have the links.

Displaying the New Link dialog

1 Select Screen.DDE.Links. The Links dialog displays.



2 To create a subclass of Link instead of using the Link class, click New Class. The New Class dialog displays.

3 Complete the New Class dialog.

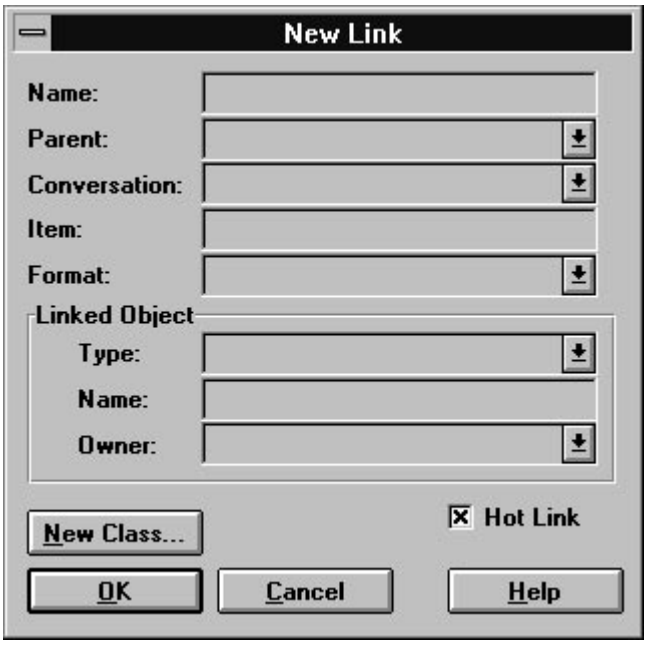
📖 *For more information:* See "Creating windows, dialogs, and controls" on page 14.

4 In the Name field, enter a name for the new class.

5 In the Owner field select the parent class of your new class.

6 Click New. You return to the New Conversation dialog.

7 In the Links dialog, click New. The New Link dialog displays.



Creating the Link instance

In the New Link dialog:

1. Specify the name of the Link instance in the Name field.
2. Select the name of the parent class for the Link instance in the Parent field.

   ☑ *Tip:* The parent class is either the Link class or a subclass of Link.
3. To create a subclass of Link instead of using the Link class, select New Class. The New Class dialog displays.
4. Complete the New Class dialog.

   🕮 *For more information:* See "Creating windows, dialogs, and controls" on page 14.
5. Select the conversation name in the Conversation field.

   🔔 *Reminder:* The conversation must already exist.
6. Specify the server's name for the data item in the Item field.

7 In the Format field, select the format for transferring the server data.

☑ *Tip:* The format is either text (dde_FORMAT_TEXT) or AionDS binary (dde_FORMAT_AIONBINARY).

8 Select the linked object's type in the Type field.

☑ *Tip:* The type is either a slot or parameter.

9 Specify the linked object's name in the Name field.

10 Select the linked object's parent in the Owner field.

11 By default, the Hot Link check box is selected so that the value is updated automatically. Deselect this check box to have a warm link instead.

12 Click OK.

Modifying links

Use the Links dialog to edit or delete existing Link instances.

1 **To edit a link,** select the link name and click Edit Link. The Edit Link dialog displays.

2 The Edit Link dialog contains exactly the same fields as the New Link dialog. Change the desired fields and click OK.

Result: The link fields change and you return to the Link dialog.

1 **To delete a link,** select the link name and click Delete Link. A confirmation window displays.

2 Click Delete to delete the link.

Result: The link is deleted and you are returned to the Links dialog.

3 Click Cancel to cancel the deletion.

Result: The link is not deleted and you are returned to the Links dialog.

Building AionDS as the server

Overview

To make AionDS the server, organize the client application information by creating topic classes with data item instances:

- **Topic** classes contain data item instances
- **Data item** instances identify the knowledge base objects containing data values requested by a client application. A data item does not contain the actual data, but the name the client application uses to request the data and a pointer to the knowledge base object containing the data.

Linking data

You do not need to define a user interface for data that AionDS provides to a client application. You do not have to link the data items in DDEObject to the windows in the WindowObject class. However, a server application must have at least one window that can act as an anchor for DDE messages from client applications.

Creating topics and data items

This section describes how to create topics and data items. You can also customize the default processing for server requests for data items.

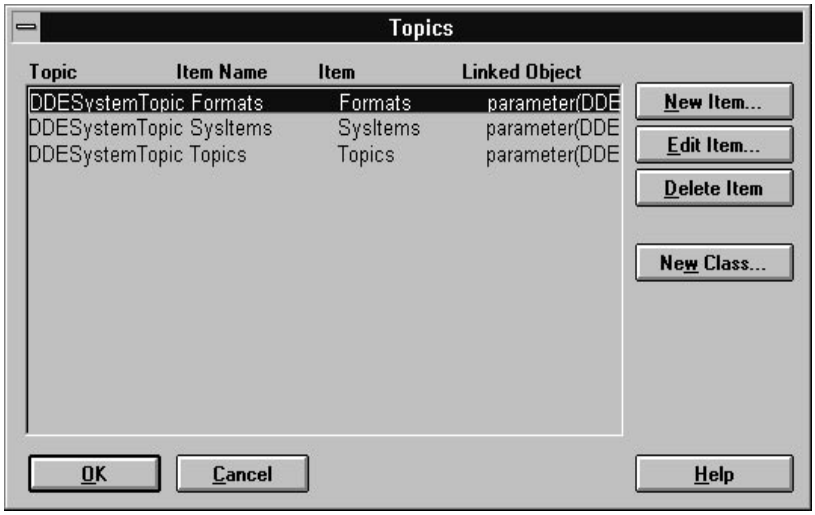
📖 *For more information:* See *Building GUI applications: Reference* to learn how to customize default processing.

AionDS provides information about its DDE implementation through a System topic. You can add data items to the System topic.

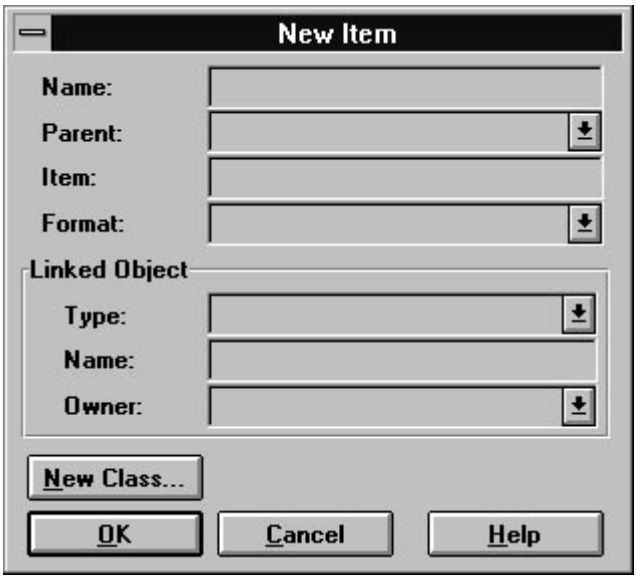
What to do

Use the Topics dialog to access the New Item dialog.

1 Select Screen.DDE.Topics. The Topics dialog displays.



2 Click New Item. The New Item dialog displays.



3. Specify the instance name for the topic class in the Name field.
4. To create a new topic class for this data item, click New Class. The New Class dialog displays.
5. Complete the New Class dialog.

   📖 *For more information:* See "Creating windows, dialogs, and controls" on page 6-14.
6. Select the topic name in the Parent field.

   ☑ *Tip:* To add a data item to SystemTopic, select SystemTopic in the Parent field. AionDS updates the SysItems data item at runtime, adding any new data items in SystemTopic.
7. Specify the data item name in the Item field.
8. Select the data format in the Format field.

   ☑ *Tip:* The format is either text (dde_FORMAT_TEXT) or AionDS binary (dde_FORMAT_AIONBINARY).
9. Select the linked object type in the Type field.

   ☑ *Tip:* The type is either a slot or parameter.
10. Specify the linked object's name in the Name field.
11. Select the linked object's parent in the Owner field.
12. Click OK.

Modifying data items

Use the Topics dialog to edit or delete data items.

1. **To edit a data item,** select the item name and click Edit Item. The Edit Item dialog displays.
2. The Edit Item dialog contains exactly the same fields as the New Item dialog. Change the desired fields and click OK.

Result: The item fields change and you return to the Topics dialog.

1. **To delete a data item,** select the item name and click Delete Item. A confirmation window displays.
2. Click Delete, to delete the item.

   Result: The item is deleted and you return to the Topics dialog.
3. Click Cancel to cancel the deletion.

   Result: The item is not deleted and you return to the Topics dialog.

Chapter 7
Working with External Databases

Introduction

This chapter shows you how to define database access methods to store and retrieve information from databases external to AionDS. You can access QSAM, dBASE, SQL, and VSAM databases.

In this chapter

Database menus

Overview

When you are editing a class, the Database pull-down menu is available.

Summary

The Database pull-down menu contains the following commands:

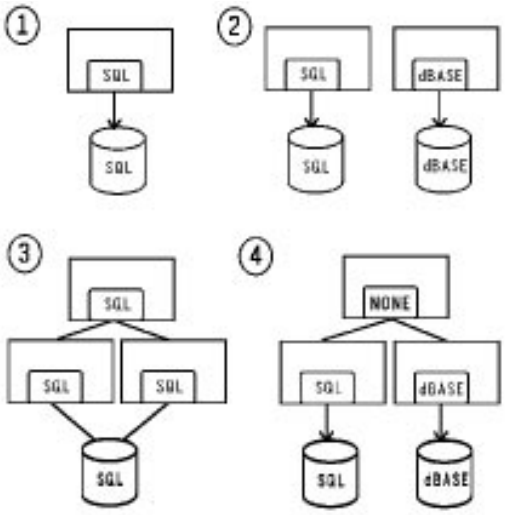
Access method	lets you select one of the following database access methods: None SQL QSAM VSAM dBASE
Import definition	lets you import the database definition from an external source and generate AionDS Type Definition Language (TDL) statements for the database.
Export definition	lets you export the class definition property into an external database. When the access method is SQL or dBASE, AionDS generates the external database.
Generate slots	lets you generate the slots for an AionDS class. AionDS uses the Class Definition property to generate the slots.
Generate class	lets you generate the Class definition property from the slots of the class.

Access method configuration

Overview

An Access Method describes the database structure and coordinates the integration of the data within an AionDS class. AionDS can access a single database or multiple external databases by using single class or multiple class access methods. Since a class can have only one assigned access method, define multiple classes to access different databases.

Access method configurations



The previous figure shows the following access method configurations:

1 one class, one access method, one database

2 multiple classes, multiple access methods, multiple databases

3 multiple classes, one inherited access method, one database

4 multiple classes, multiple specialized access methods, multiple databases

Configuration 1 shows one class inheriting one access method. The class has access to the database.

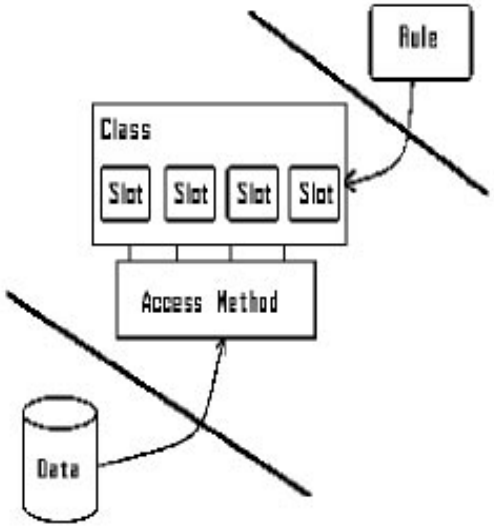
Configuration 2 shows multiple classes. Each class has one access method and accesses one database.

Configuration 3 shows two classes inheriting access methods from the parent class. The two classes have access to the single database. If more than one database exists, the classes can be dedicated to different databases.

Configuration 4 shows a parent class with NONE specified in the Access Method property. Each subclass is dedicated to a different database and requires a different access method. Subclasses cannot use a different access method than the parent class. If the subclasses require different access methods, do not specify an access method for the parent class.

Access methods at runtime

The following figure shows the Access Method and its relationship to AionDS at runtime.



Required definitions

For automatic data integration at runtime, set up the following definitions at edit time:

- an access method for the class
- properly defined class properties
- matching external database and database-related class properties

Access method class

Any class meeting these requirements is called an **access method class**.

The figure on page 7-10 shows a rule AionDS uses to process an access method class. When executing a rule, AionDS accesses the specified external database and reads specified records matching the selection criteria. The record fields correspond to slots in the access method class.

Persistent data

Persistent data is the instances of an access method. The slots holding the record fields are available for use by AionDS in the same way as normal slots and parameters. Instances of an access method class reside in an external database. AionDS does not delete these instances after the application terminates.

Persistent data is available after an application terminates, and remains available to AionDS as if the persistent data were an AionDS object. For example, fields held in slots are used in a rule or displayed in a message. After the application terminates, AionDS erases or returns the previously accessed records to the database, depending on what you specify in the Update property.

Defining an access method

What to do

To use the Automatic Database Interface (ADI) to store and retrieve information from an external database, define a database access method.

To define an access method:

1. Specify the access method of a class.
2. Specify the database file to access.
3. Map the external record to AionDS through the Class Definition property and through slots in the class. The Class Definition property maps the external data structure and the slots map the individual fields of the record.

   Use one of the following methods to map the external record to AionDS through the Class Definition property and the slots in a Class:

   - Create the external database. Generate the Class Definition and corresponding slots from the external database.
   - Specify the access method and database attributes. Generate the Class Definition and slots. Create the external database from the Class Definition and slots.
4. Specify Selection Criteria to limit the number of records read into AionDS memory during data access.
5. Specify any pertinent load, update, or integrity information.

Changing the access method

What to do

1. Open the appropriate class.
2. Select the new access method from Database access in the Database menu.

Caution: AionDS might delete information specific to the previous access method. If so, perform the following steps:

1. Map the new external record to AionDS through the Class Definition property and the slots in the Class.
2. Specify Selection Criteria to limit the number of records read into AionDS memory during data access.
3. Specify any pertinent load, update, or integrity information.

Generating a class definition and slots

What to do

To generate a Class definition and the corresponding slots from an external database:

1. Open the appropriate class. The database menu displays.
2. Select a database access method from Database.Access Method.
3. Identify the external database by specifying the required attributes in the following properties:

QSAM	External Source Specification property
dBASE	External Source Specification and Data File properties
SQL	External Source Specification, Tablename, Interface, and Access String properties
VSAM	External Source Specification property

📖 *For more information:* See Appendix A of the *I/O Reference* for the appropriate syntax in the External Source Specification property. See "Properties available in the class" on page 7-10.

4. Select Database.Import definition. AionDS builds the Class Definition property.
5. Select Database.Generate slots. AionDS builds the slots for the Class.

Generating an external database

What to do

To generate an external database based on the Class definition:

1. Open the appropriate class. The Database menu displays.
2. Select a database access method from the Database.Access Method menu.
3. Select an external database:

QSAM	External Source Specification property
VSAM	External Source Specification property
dBASE	External Source Specification and Data File property
SQL	External Source Specification, Tablename, Interface, and Access String properties

   📖 *For more information:* See Appendix A of the *I/O Reference* for the appropriate syntax in the External Source Specification property. See "Properties available in the class" on page 7-10.

4. Select Database.Generate class. AionDS builds the Class Definition property.

   ☯ *Alternative:* Open the Class Definition property and specify the attributes in the Class definition.

5. Select Database.Export Definition. AionDS creates the external database.

Properties available in the class

Overview

When you select a database access method for a class, AionDS adds several database-related properties to the class. This section describes the properties and the entries available in each.

- Access String (SQL only)
- Alternate File Name (VSAM only)
- Alternate Key Field (VSAM only)
- Class Definition
- Commit Mode
- Data File Name (dBASE, QSAM, and VSAM only)
- Data Integrity Check
- Data Location
- Database (SQL only)
- External Source Specification
- Index File Names (dBASE only)
- Interface (SQL only)
- Key Fields
- Load Mode
- Mapped Slots
- Selection Criteria
- Table Name (SQL only)
- Update Mode

Access String (SQL only)

Purpose

The Access String property is user-specified, containing information necessary to establish a connection with SQL, such as a user ID or password.

📖 *For more information:* See the *General Reference*.

Alternate File Name (VSAM only)

Purpose

You can only specify this property only if the VSAM file is a keyed-sequence data set (KSDS) or an entry-sequenced data set (ESDS).

Alternative File Name contains an optional cluster or path name used to load records from a KSDS. You only need to specify the Alternative File Name when loading records from a base cluster or path and updating those records using a different base cluster or path.

Example

The Data File Name property specifies a base cluster, and the Alternate File Name property specifies a path built from that base cluster. Using the alternate index, AionDS loads records from the Alternate File Name, and updates those records in the Data File Name with the primary index.

Alternate Key Field (VSAM Only)

Purpose

Use this property only if the VSAM file is a KSDS or an ESDS, and you specify an Alternate File Name.

The Alternate Key Field specifies the optional key field that you use to limit the number of records retrieved from the Alternate File Name (see Selection Criteria). The key field must reference a mapped slot in the access method.

Class Definition property

Purpose

The Class Definition property can be either user-defined or AionDS-defined. The Class Definition property maps the external database data structure to AionDS. The following example shows an external dBASE record mapped in the Class Definition property.

```
file(dBASE) of record
   item is string(20)
   price is integer(10)
   department is string(20)
   priority is string(10)
   purchased is Boolean
end
```

What to do

The record structure in the Class Definition property must match the data structure in the external dBASE, QSAM, SQL, or VSAM database exactly.

You can either enter the record structure manually in the Class Definition property or AionDS automatically maps the record structure, specifying a Database File Name (or Table Name for SQL) and executing the import option.

One method is to specify a Database File Name (or Table Name for SQL) and execute the IMPORT DEFINITION command.

📖 *For more information:* See the section “External Source Specification” on page 7-22.

The other method requires that you specify slots to match each record field in the external database data structure and then execute the GENERATE command. The GENERATE command maps each defined slot to a field in the class definition.

Commit Mode

Purpose

Use the Commit Mode property to control when to execute commits and rollbacks for SQL databases. To commit changes automatically on an update to an SQL database, set Commit Mode to Automatic. To commit or roll back changes manually using the DBMS statement with either the commit or rollback option, set Commit Mode to On Request.

📖 *For more information:* See the *Language Reference* to learn more about the DBMS statement.

Changes committed or rolled back

When a commit or rollback is executed, the current SQL unit of work is committed or rolled back—all classes which have current SQL connections are committed or rolled back to the last commit point. A **unit of work** consists of the changes made by the knowledge base to databases since the last commit.

Valid values

You can set this property to the following values:

Automatic	A commit is executed when the update is completed.
On Request	A commit is not executed when the update is completed. You can use the DBMS statement with either the commit or rollback option to execute a commit or rollback.

Data File Name (dBASE, QSAM, and VSAM only)

Purpose

The Data File Name property is user-specified, indicating the name of the data file and its location.

Path names

You can specify path names in the Path Names property. For example, if the DATA directory on drive D:\ contains the data file, you can specify the following path name for the dBASE data file named PURCHASES.DBF:

```
d:\data\purchases.dbf
```

VSAM

If the Access Method is VSAM, this property contains the name of the base cluster or path (KSDS or ESDS) to access.

RRDS

With RRDS, use the Data File Name for loading and updating records.

KSDS or ESDS

With a KSDS or ESDS, the Data File Name is used for updating records. If an Alternate File Name is not specified, the Data File Name is used for loading records. If a KSDS is being updated, the Data File Name must identify a base cluster or path with a UNIQUE key.

Data Integrity Check

Purpose

The Data Integrity Check property is user-specified, indicating whether AionDS must check each record in the external database. AionDS can verify that the data remains unchanged before writing the records to the external database.

Valid values

You can set the Data Integrity Property to:

- YES

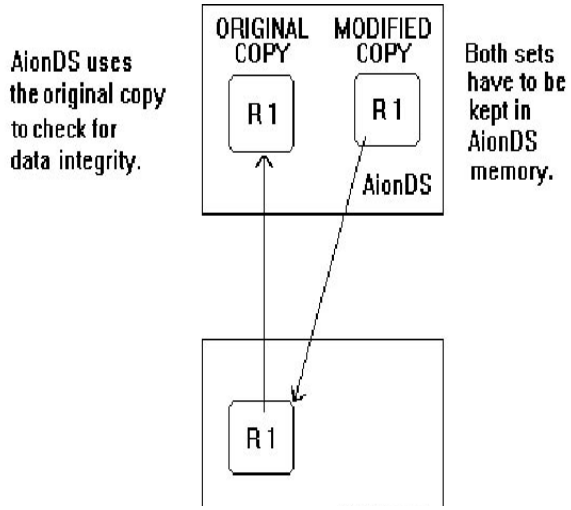
  AionDS checks each record for modifications before writing the record to the external database. For example, when AionDS loads a record from the database, it copies the record to memory. The original record still exists in the database and can be modified by other users or processes.

- NO

  AionDS checks the original database record and displays an error message for any changes found since the last copy. AionDS does not maintain data integrity.

Memory cost

Specifying Yes increases the amount of memory needed by the data. To compare each record with its corresponding external record, AionDS keeps a copy of the original record in memory, in addition to any modifications made during a knowledge base execution. The following figure illustrates this concept.



If the record in the database matches the copy of the original in AionDS, AionDS writes the modified record to the database.

If the record in the database differs from the copy of the original, AionDS updates as many records as it can and writes a message stating that not all records could be updated. If you are using SQL, AionDS performs all possible updates, but does not issue a commit.

Data Location

Purpose

The Data Location property is AionDS-specified or user-specified, indicating one of the following:

- the database is located in the same environment as AionDS
- access to the database is through a data communications link

Valid value

You can set this property to:

LOCAL	the data is in the same environment as AionDS, making the data accessible without a data communications link. This is the default.
REMOTE	the data is in a CICS, IMS or IDMS-DC region, limiting access using a data communications link. To access data remotely, you need the AionDS Cooperative Processing Option.

On the mainframe

You can run a knowledge base in AionDS/CICS or AionDS/IMS to access data remotely.

📖 *For more information:* See Appendix A, "Porting Knowledge Bases," in the *General Reference* for details about porting knowledge bases from the PC to the mainframe.

VSAM and SQL

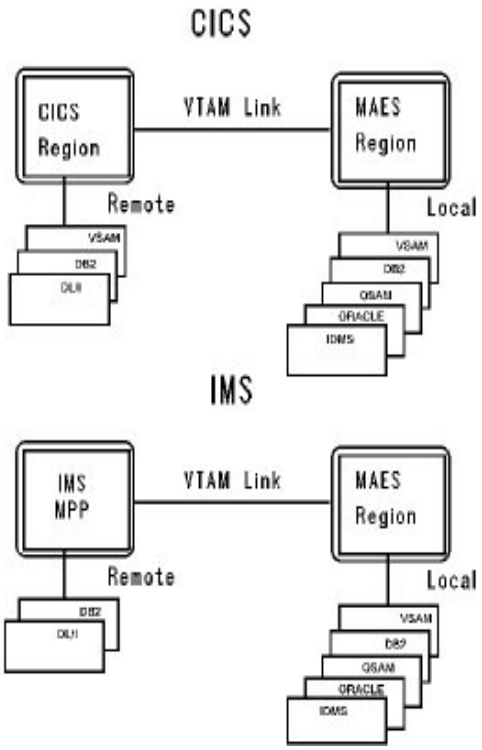
Knowledge bases can access VSAM, DB2, and DL/I databases from AionDS/CICS. Knowledge bases also can access DB2 and DL/I databases locally or remotely from AionDS/IMS. When running a knowledge base through CICS or IMS transaction drivers (AAES and AIONADS), a connection to the MAES region is made and, with certain access methods, you can specify whether data access is local or remote. For example, for a

CICS transaction with VSAM access specified, you can access the VSAM file locally, either within the MAES address space, or remotely within CICS.

📖 *For more information:* See the *I/O Reference* to learn about local and remote access.

AionDS, CICS, and IMS

The following figure shows the relationship between AionDS and the other components in CICS and IMS environments.



Local requests use a memory-to-memory protocol and do not require VTAM services. Local requests do not synchronize data access with CICS or IMS message traffic.

Remote requests travel across a VTAM link before CICS or IMS executes them. These requests are synchronized with message traffic and can be recovered in the event of a system failure.

The following requests are always local:

- AionDS/MVS requests to DB2, Oracle, and IDMS
- AionDS/VM requests to SQL/DS and Oracle

Database (SQL only)

Purpose

The Database property is user-specified, and indicates the default name of the SQL database. AionDS uses this name to locate and establish a connection with SQL.

External Source Specification

Purpose

When AionDS generates a class definition, use the External Source Specification property to specify a source. The External Source Specification property contains a statement that specifies the kind of source and its location. For example, you can use the following specification to generate a class from a COBOL copy file that contains a record definition:

```
cobol import * from 'ce371xx.cobol.copy(prog_1)'
```

📖 *For more information:* See Appendix A of the *I/O Reference* for external sourcing syntax.

Generating the statement

To generate the statement automatically, specify a Data File Name for dBASE or Table Name for SQL, and select the Import definition command. When executing the Database.Import definition command, AionDS automatically maps the external data structure to the Class Definition property.

📖 *For more information:* See “Defining an access method” on page 7-6.

Index File Names (dBASE only)

Purpose

The Index File Names property is user-specified, indicating the name of the index file, or files, in dBASE.

You can specify any number of index file names separated by commas. For example, you can specify the following dBASE index file names:

```
index file names --> purchase.ndx,
                     return.ndx,
                     inventory.ndx
```

Index File Specification

Purpose

The Index file specification property appears in AionDS after you select Database.Access Method and specify dBASE or SQL.

Use the Index File Specification property to change the external file name to assign fields (of the class definition) to index files.

Valid values

The syntax is as follows:

```
indexname = field1 [+ field2 [+ fieldn]] [,]
```

indexname name of index

field name of column (this column name must exist as a field in the class definition)

Interface (SQL only)

Purpose

The Interface property is user-specified, and specifies the default SQL access method; for example, DB2.

Valid values

For the following systems you can specify the listed interface names in the interface property:

AionDS/AIX

AionDS/AIX uses the following Interface property values to access the corresponding databases:

Interface property value	Database accessed
IBM	IBM DB2/6000
Informix	Informix
ODBC	use the Open Database Connectivity (ODBC) interface to connect to any ODBC-compliant database
Oracle	Oracle
Sybase	Microsoft/Sybase SQL Server 4.X/10.X

AionDS/HP-UX

AionDS/HP-UX uses the following Interface property values to access the corresponding databases:

Interface property value	Database accessed
IBM	IBM DB2/6000
Informix	Informix
ODBC	use the Open Database Connectivity (ODBC) interface to connect to any ODBC-compliant database
Oracle	Oracle
Sybase	Microsoft/Sybase SQL Server 4.x/10.x

AionDS/MVS

AionDS/MVS uses the following Interface property values to access the corresponding databases:

Interface property value	Database accessed
DB2	IBM Database 2 (DB2)
dbc/1012	Teradata DBC/1012
Oracle	Oracle

AionDS/OS2

AionDS/OS2 uses the following Interface property values to access the corresponding databases:

Interface property value	Database accessed
DB2	IBM DB 2
IBM	IBM DB2/2
Informix	Informix
ODBC	use the Open Database Connectivity (ODBC) interface to connect to any ODBC-compliant database
Oracle	Oracle
Sybase	Microsoft/Sybase SQL Server 4.X/10.X

AionDS/VM

AionDS/VM uses the following Interface property values to access the corresponding databases:

Interface property value	Databases accessed
Oracle	Oracle
SQL/DS	IBM/SQL/Data System (SQL/DS)
dbc/1012	Teradata DBC/1012

AionDS/Win

AionDS/Win uses the following Interface property values to access the corresponding databases:

Interface property value	Databases accessed
DB2	IBM DB 2
IBM	IBM DB2/2
Informix	Informix
Microsoft	Microsoft SQL Server 6.0
ODBC	use the Open Database Connectivity (ODBC) interface to connect to any ODBC-compliant database
Oracle	Oracle
Sybase	Microsoft/Sybase SQL Server 4.x/10.x

Key Fields

Purpose

The Key Fields property specifies the fields used as index fields. To update SQL tables, you need key fields for rows.

If you specify an index in the external sourcing specification, AionDS automatically maps the key fields:

```
dBASE IMPORT * from data.dbf indexing (purchase.ndx)
```

AionDS maps the index fields in the dBASE index file PURCHASE.NDX and lists the index fields in the Key Fields property.

📖 *For more information:* See Appendix A of the *I/O Reference* for external sourcing syntax.

SQL

AionDS maps the key fields automatically when importing from DB2/2, Informix, ODBC, and Oracle. AionDS does not map the key fields automatically when importing from Sybase.

Syntax

The following is the syntax for SQL key fields:

$key_1, key_2, \ldots , key_n$

`key` name of the SQL table columns that an index specifies. If the index specifies more than one column, insert a plus sign between the column names.

Example

An SQL table has the following columns:

```
int1
int2
txt1
txt2
```

These columns are indexed as follows:

Index	Columns
`indx1`	`int1`
`indx2`	`int2, txt2`

To map these indexes and columns into the Key Fields property correctly:

```
int1, int2 + txt2
```

Access method is VSAM

If the Access Method is VSAM and the file is a KSDS, you can specify one key field. To update the file, use the key field. The key field must reference a mapped slot in the access method.

When loading records, use the key field to limit the number of records retrieved from the file (see the Selection Criteria property). When updating records, use the key field to locate the record to update.

Load Mode

Purpose

The Load Mode property is user-specified, and indicates whether AionDS automatically loads data or uses the DBMS LOAD command.

Valid values

You can have the following values:

AUTOLOAD	AionDS automatically loads the data for a class into memory. This entry is the default.
ONREQUEST	When encountering a DBMS LOAD command while executing a knowledge base, AionDS loads only the selected data into memory.

Example

When AionDS encounters the following KDL statement, AionDS loads the required data:

```
if purchased is true
then
  dbms(load,cl_access)
end
```

If the parameter or slot called `purchased` evaluates to the value of TRUE, AionDS loads the required data as specified in the access method attached to the class called `cl_access`. ONREQUEST data is never automatically loaded; data is loaded only when AionDS encounters the DBMS LOAD statement.

Mapped Slots

Purpose

The Mapped Slots property is an AionDS-maintained property that lists the slots in the class mapped to fields in the database.

Valid values

You can map slots by:

- using the Generate class command
- using the Generate slots command
- modifying the Mapped To property of a slot

📖 *For more information:* See "Generating a class definition and slots" on page 7-8 and "Generating an external database" on page 7-9. See the *General Reference* for information about the Mapped To property.

Selection Criteria

Purpose

The Selection Criteria property is user-specified, limiting the records brought into AionDS memory to those records meeting the specified conditions. You can specify any KDL statement that evaluates to a Boolean expression. The following line shows a sample selection criteria statement:

```
purchased = yes
```

Specifying multiple criteria

Using the AND statement, you can combine selection criteria statements:

```
purchased=yes and
pay_method is 'cash'
```

Using the OR statement, you can set multiple conditions, one of which must be met:

```
purchased=no or
pay_method is 'credit'
```

You can combine the OR and AND statements:

```
purchased=no or
pay_method is 'credit' and
credit is 'good'
```

Only those records where the field `purchased` is `no` or the value of the field `pay_method` is `credit` and the value of the field `credit` is `good` are selected and copied into AionDS. AionDS does not select or copy into memory any record that does not meet these criteria.

Access method is VSAM

If the Access Method is VSAM and you retrieve records from a KSDS, you can write the Selection Criteria to limit the number of records retrieved from the file. Specify a field in the Key Fields or Alternate Key Field property.

Consider the following properties:

```
Class Definition:

  file (vsam,noattr) of record
      firstname is string(10)
      lastname is string(30)
      department is string(20)
      salary is real(8:2)
  end

Selection Criteria:

  (department = 'SALES') and (salary > 50000.00)

Key Field:  department
```

With this Selection Criteria, AionDS first selects all records from the file where the department field is `sales`. After retrieving these records, AionDS applies the remaining criteria (for example, consider only those records where the salary field exceeds 50000.00).

Alternatively, consider the following Selection Criteria property:

```
(department > 'SALES') and (salary > 50000.00)
```

AionDS selects and retrieves all records whose department field is greater than `sales` (a KSDS stores records in key-sequenced order). AionDS applies the remaining criteria to these retrieved records.

The use of the Selection Criteria property on a KSDS file is as follows:

- If you do not specify a key field in the Key Fields or Alternate Key Field properties, AionDS applies the Selection Criteria to all records in the file.

- If you specify a key field in the Key Fields or Alternate Key Field properties, AionDS uses the key field to select the records to retrieve. AionDS retrieves only those records whose key field complies with the following:

  keyfield { = | >= | > } *keyvalue*

=	equals keyvalue
>=	is greater than or equal to keyvalue
>	is greater than keyvalue

SQL Select (SQL only)

Purpose The SQL Select property is the string used by a class to acquire its data from a server.

Table Name (SQL only)

Purpose

The Table Name property is a user-specified property indicating the name of the SQL table AionDS accesses. AionDS uses this name to locate and establish a connection with the specified SQL table.

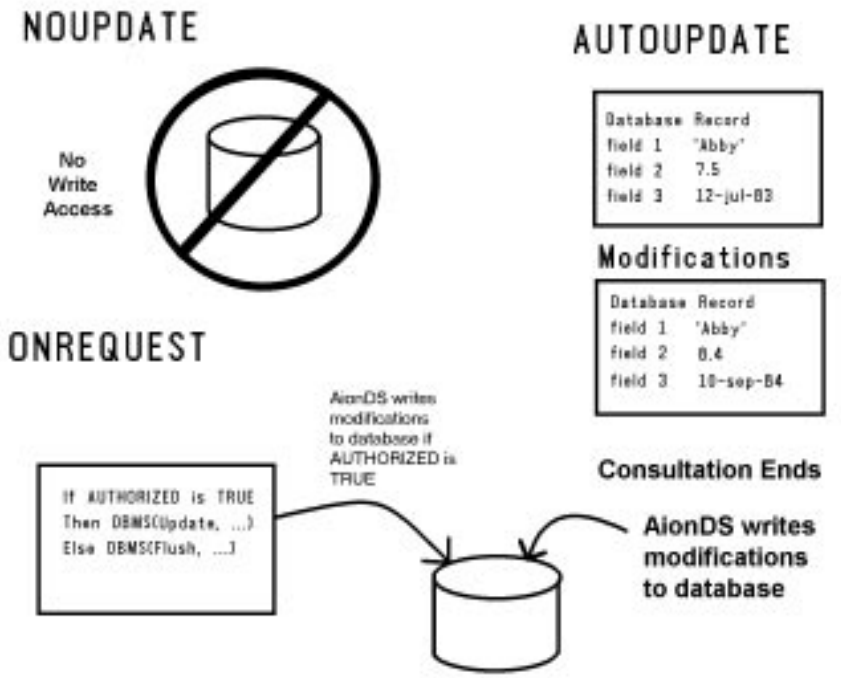
Update Mode

Purpose

In the Update Mode property you can specify whether to use NOUPDATE, AUTOUPDATE, or ONREQUEST.

NOUPDATE	You cannot modify any data accessed by AionDS. This entry is the default.
AUTOUPDATE	When knowledge base execution terminates, AionDS automatically writes your modifications to the database.
ONREQUEST	When encountering a dbms load `command` while executing a knowledge base, AionDS loads only the selected data into memory.

The following figure shows the actions of the three possible entries.



NOUPDATE You cannot make modifications to the database with NOUPDATE selected. Since NOUPDATE does not maintain as much database information in memory, it is the most efficient option. NOUPDATE is the default entry.

ONREQUEST You have write access when AionDS executes the dbms update statement for the specified access method class. As shown above, you can use ONREQUEST to limit write access to a database. You can use onrequest UPDATE in combination with other DBMS statements to limit write access to certain stages of knowledge base execution.

📖 For more information: See Chapter 4 of the *Language Reference*.

AUTOUPDATE	You can use AionDS to make modifications to the database records. The previous figure shows that field 2 and field 3 are modified. At the end of the knowledge base execution, AionDS writes these modifications to the database.

Setting SQL commits and rollbacks

Overview

You can control exactly when commits and rollbacks are performed for SQL connections. In addition, you do not have to know the syntax of the SQL statement for that specific database. You can manually set commits and rollbacks using the DBMS KDL statement and the Commit Mode property.

📖 *For more information:* See the *Language Reference* to learn about the DBMS statement. See the *I/O Reference* to learn about executing commits and rollbacks for specific SQL databases.

Setting commits and rollbacks

Commits and rollbacks can be established using one of the following methods:

- To manually set commits and rollbacks, set the Commit Mode property to OnRequest, and specify DBMS statements at the appropriate places in the data retrieval process.
- To automatically set commits, set the Commit Mode property to AutoCommit.

Changes committed or rolled back

When a commit or a rollback is issued, the current SQL unit of work is committed or rolled back; that is, all classes that have current SQL connections are committed or rolled back to the last commit point. A **unit of work** consists of the changes made by the knowledge base to databases since the last commit.

What to do

1. In the class, set the Commit Mode property to OnRequest.

   Setting Commit Mode to OnRequest ensures that no action is taken on update. If Commit Mode is set to AutoCommit, a commit is issued after an update is completed.

2. In the appropriate class, set the conditions under which you want a commit or rollback to occur, and include the DBMS statement with either the commit or the rollback option.

Setting up SQL error handling

Overview

To set up error handling for SQL-generated errors, you can add the WhenUpdateError and the WhenError methods, create the corresponding functions, and attach the functions to the WhenUpdateError and WhenError methods. In the functions, you can write KDL to handle SQL-generated errors. For example, you can write KDL to issue rollbacks on specific errors. In response to SQL-generated errors, AionDS sends the WhenUpdateError demon while updating an SQL database and the WhenError demon while connecting to or reading data from an SQL database.

What to do

1. Add the WhenUpdateError or WhenError method to the desired class.
2. Create the functions for the WhenUpdateError or WhenError method. Include any KDL for handling errors.
3. Attach the function to the WhenUpdateError or WhenError method.
4. If you are using the WhenUpdateError method, set the Commit Mode property to OnRequest in the Access Method monitor for the class. Setting Commit mode to OnRequest ensures that neither a rollback nor a commit is executed when an update error is encountered. Setting Commit mode to OnRequest lets you use the customized error processing to execute commits and rollbacks. You do not use the Commit mode property with the WhenError method.
5. To display error information, you should use primarily message or text windows—although you could also use display, report, or other GUI windows.
6. If you are using the WhenUpdateError method, you can use the DBMS statement with the commit or the rollback options to commit and roll back changes.

📖 *For more information:* See the *Language Reference* to learn about the DBMS statement. See the *I/O Reference* to learn about executing commits and rollbacks for specific SQL databases.

Chapter 8
Tracking Changes in a Knowledge Base

Introduction

Change Management is used to track, document, and control the changes made to a knowledge base. Use it to do the following:

- facilitate incorporating changes made simultaneously by several people.
- record the objects changed, the person who made the change, and the reasons for the change.
- control changes made to a knowledge base.

In this chapter

Change Management procedures

Overview

This section describes the procedures you should use to track, document, and control the changes made to a knowledge base.

In Change Management, use the following knowledge bases:

- **master knowledge base**. This is typically a knowledge base to control changes.
- **working copy** of the master knowledge base. Check out a working copy of the knowledge base and modify the working copy—not the master knowledge base.

To ensure that your changes do not conflict with other's changes, you should always start with a fresh copy of the master. When you check out a working copy and the master contains unapplied change sets, apply those change sets to your working copy before making changes to your working copy.

📖 *For more information:* See "Checking out a working copy with unapplied change sets" on page 8-19.

As you make changes to your working copy, the changes are recorded in an external file, which is called a change set. The **change set** contains an entry for each object that you change. Changes to an object are stored and applied by property. For example, if you modify only the Facts of a parameter in one change set, only the Facts are updated when that change set is applied. If you modify the Prompt property of the same parameter in a later change set, the Facts property is not changed. Record information about why you are making changes by opening the change set and updating the Comment property of the change set.

After completing all changes and testing your working copy, resolve any conflicts between your changes and changes made by others. When you apply change sets to a working copy or master, a change in an object property made by one change set can overwrite a previous change to the same object property by the other change set. This situation is called a conflict, because the two changes work against each other—one change cancels the other. To check the change sets for conflicts, use the Compare command.

📖 *For more information:* See "Resolving change set conflicts" on page 8-23.

Checking in a change set copies the change set to the master. When you check in a change set, AionDS deletes it from, and starts a new change set in, your working copy. This ensures that any changes you make are copied to the master only once.

To avoid working on out-of-sync working copies, frequently apply the change sets to the master. Check out new, identical working copies more often. Assign a change administrator to apply the change sets to the master.

📖 *For more information:*

- See "Updating out-of-sync working copies with the new master" on page 8-20.
- See "Integrating change sets before applying them to the master" on page 8-21.

Because changes cannot be undone after being applied and saved, back up the master by exporting it before applying changes.

☑ *Tip:* Keep change sets small in size and modular. Having each person make changes to a different part of the knowledge base reduces the chances that they will modify the same object property. For example, make logic changes in one change set and make user interface changes in a separate change set. The compression and apply processes are simplified when dealing with smaller units of data. Keeping change sets small and modular reduces the number of conflicts when you apply change sets. Consequently, change sets can be checked in and applied to the master more frequently and with less effort.

What to do

To track the changes you make to a knowledge base, use the following procedure:

1. Place the master in a location that is accessible to everyone who will be making changes to the master.
2. Check out a working copy of the master.
3. Apply any unapplied change sets to the working copy.
4. Make changes to the working copy. The changes are recorded in a change set.
5. After completing all changes and testing your working copy, check the master and other people's working copies for unapplied change sets.
6. If the master contains unapplied change sets, copy them to your working copy.
7. If any other person is ready to check in their working copy's change set, copy their change set to your working copy.
8. Compare all change sets.
9. Resolve any conflicts.
10. Check in all change sets to the master.
11. Periodically, export the master and apply all change sets.

Change Management menu commands

Overview

This section contains information on the commands that are available from File.Change Management.

Creating a working copy of a knowledge base (Check out)

Overview

To begin the process of modifying a knowledge base when using Change Management, you usually start by creating a copy of the master knowledge base. This is also called **checking out** a working copy of the master knowledge base. When you check out a copy, Change Management is automatically enabled in the working copy. All changes you make are automatically recorded and can later be applied to the master.

What to do

To check out a copy of a knowledge base, perform the following steps:

1. Open the master knowledge base.
2. Select File.Change Management Check out. The Check Out Knowledge Base window displays.

   📖 *For more information:* The Check Out Knowledge Base window is similar to the Create Knowledge Base window. See Chapter 3, "Working at the Knowledge Base Level," to learn more about using the fields in this window.
3. Specify the name, directory, and version of the working copy.
4. Click Check Out.

   If any unapplied change sets exist in the master copy, AionDS copies them to the working copy. Then apply them. See "Change Management procedures" on page 8-2.

View the knowledge base's change sets (List)

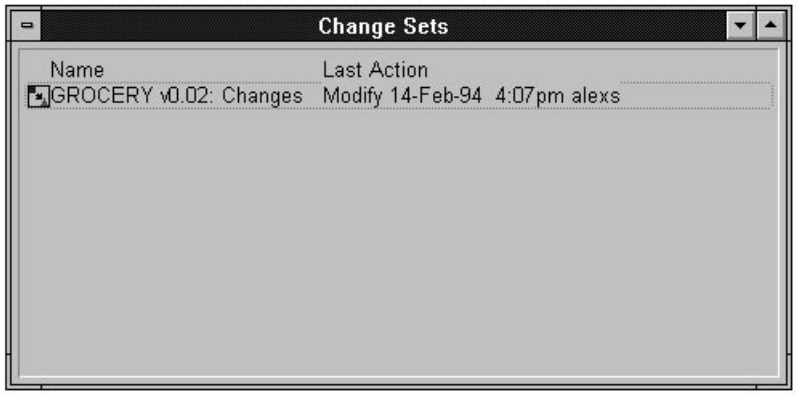
Overview

Change Management records information about the types of changes you make, but it cannot record information about why you are making the changes. Update the comment property of the change set to indicate the reason for changes.

What to do

To list the unapplied change sets in a knowledge base, perform the following steps:

1. Open the knowledge base that contains the change sets.
2. Select File.Change Management List. The Change Sets window displays.



The Change Sets window displays the name of the change set and the last action performed on this change set. This Last Action field includes the type of action, date and time of the action, and the name of the person performing the action.

The type of action can be the following:

Modify	change set is the base change set for the current knowledge base
Chkin	change set was checked into the master
Apply	change set was applied to the master

Editing the knowledge base's change sets (Open)

Overview

Change Management records information about the types of changes you make, but it cannot record information about why you are making the changes. Update the comment property of the change set to indicate the reason for changes.

What to do

To open a change set to view or update the information, perform the following steps:

1. Open the knowledge base that contains the change set.
2. Select File.Change Management List. The Change Sets window displays.
3. Select the change sets.
4. Select File.Change Management Open. The Change Set windows for those change sets display.

   ☯ *Alternative:* Double-click on the desired change set in the Change Set List.

   To list properties that are not visible, select Property.Open.

   ☑ *Tip:* Customize the display for the Change Set Editor just like for any AionDS editor.

The following list describes the change set properties:

Property	Description
Title	enter a name for the change set equal to or less than 50 characters
Source KB	shows the path and name of the master knowledge base
Modified	shows the time and date when the change set was last modified and by whom
Checked in	shows the time and date when the change set was last checked into the current knowledge base and by whom
Applied	shows the time and date when the change set was last applied to the current knowledge base and by whom
Comment	enter any text into the Comment property. It serves as a notepad to describe the nature and purpose of the changes contained in the change set.
Reference	enter any text into the Reference property. Organizations often document changes to applications and refer to them by a change number. The Reference property serves as a place for you to enter any change number to refer to the changes in this change set.
Changed Objects	AionDS automatically keeps track of the objects that you change and lists them in the Changed Objects Property. This property is a selectable list. In other words, open the editor for an object by double-clicking on the object name or by selecting the object and pressing ENTER. AionDS refreshes this list each time you save the knowledge base. This property lists only those objects that you changed since the last time you saved the knowledge base. To view the objects that you changed since you enabled Change Management, print the change set.

Moving change sets to the master knowledge base (Check in)

Overview

When you have completed the changes in your working copy and you want to include them in the master knowledge base, the first step is to check the changes into the master.

When you check in changes, AionDS moves the change set to the master knowledge base and deletes it from the working copy. This transfers all changes recorded in your change set to the master knowledge base.

What to do

To check in changes to the master knowledge base, perform the following steps.

1. Open the knowledge base containing the change set you want to check in.

   To verify the name and directory of the master knowledge base before you check in the change set, open the change set.

   ☑ *Tip:* It is also a good idea to update the comments of the change set to indicate any final changes you have made.

2. Select File.Change Management Check In. The change set is moved to the master and deleted from the copy.

Applying change set modifications to the knowledge base (Apply)

Overview

To incorporate the changes made in a working copy into the master copy, Apply the change set. Usually, before applying change sets, check them in from the working copy and compare them to other change sets to make sure there are no conflicts.

You can also apply the changes made in other working copies. This is helpful if you are working with others to modify the same master knowledge base. Copy the change sets from another working copy or from the master knowledge base and apply the changes to your working copy.

📖 *For more information:* See "Change Management procedures" on page 8-2.

✋ *Caution:* Once the changes have been applied and saved, they cannot be undone. For this reason, it is a good idea to export the knowledge base before applying changes.

🔔 *Reminder:* When you export the knowledge base, AionDS compresses the change set. After the change set is compressed, all changes for an object are collapsed into one entry. This action reduces the amount of space required to store the change set.

What to do

To apply changes in one or more change sets:

1. Open the knowledge base to which to apply the changes.
2. Select File.Change Management List. The Change Sets window displays.

   To apply changes in change sets that are not part of this knowledge base, copy the change sets from the knowledge base in which they exist to your knowledge base.

   ☑ *Tip:* When you apply multiple change sets (done by selecting multiple change sets on the Change Sets window), AionDS applies them one at a time, in the order in which they are listed in the Change Sets window.
3. Select the change sets to apply.
4. Select File.Change Management Apply.

   If there are any errors, AionDS displays a window listing all the errors.
5. Save the changes to the knowledge base by selecting File.Save.

Comparing change sets (Compare)

Overview

Use the Compare command to identify conflicts before change sets are applied. The Compare command compares the objects in one change set with the objects in others. If any conflicts are reported, you can investigate the reason for the conflict and see which change to apply.

📖 *For more information:* See "Resolving change set conflicts" on page 8-23.

What to do

To compare changes in one or more change sets, perform the following steps.

1. Open the knowledge base containing the change sets.
2. Select File Change Management List. The Change Sets window displays.

   ☑ *Tip:* Compare changes in change sets that are not part of this knowledge base, copy the change sets from the knowledge base in which they exist to your knowledge base. You need not apply them to compare them.
3. Select the change sets to check for conflicting changes.
4. Select File.Change Management Compare.

   AionDS searches the changes in the selected change sets and generates a file containing information about any conflicts. The information indicates the change sets that contain the conflict, and the object and property that are in conflict.

   When the compare process is complete, a confirmation pop-up window displays.
5. To view the file that contains the information on any conflicts, click View.

Printing change sets (Print)

Overview

To print the objects in change sets or save them in a file (in text format), select File.Change Management Print. The information that is printed or the file that is created contains all the properties as well as an entry for each modified object in the change sets.

📖 *For more information:* See Chapter 3, "Working at the Knowledge Base Level," for a complete description of the Print, Format, and Sort windows.

What to do

You can print more than one change set at a time. To print one or more change sets, perform the following steps.

1. Open the knowledge base containing the change sets.
2. Select File.Change Management List.

   🔔 *Reminder:* If you want to print changes in change sets that are not part of this knowledge base, copy the change sets from the knowledge base in which they exist to your knowledge base. You need not apply them to print them.
3. Select the change sets you want to print.
4. Select File.Change Management Print. The Print window displays.
5. To print the object information, enter the name of your printer, which in many installations is LPT1 or PRN.

   📖 *For more information:* Consult your operating system documentation to determine the exact name of your printer.
6. To create a new file for the change sets, specify the name and directory in the Print window.
7. To overwrite an existing file, select the file and make sure that the Append to file check box is not selected.
8. To append the change sets to an existing file, select the file and click Append to file check box.

9 To change the order in which the changed object information is printed, click Sort. The Sort window displays.

When you complete changing the sort attributes, click OK to return to the Print window.

10 To perform any of the following tasks, click Format. The Print Format window displays.

- Enter header and footer text that appears on each page
- Change the page width and the page length
- Indicate the printer type
- Compress the output into as few lines as possible
- Add an index to the end of the printout
- Add a table of contents to the end of the printout
- Add an extended table of contents containing entries for all objects to the end of the printout
- Add a summary of all the objects printed to the end of the printout

When you complete changing the print format attributes, click OK to return to the Print window.

11 Click Print.

Starting Change Management (Enable)

What to do

To enable Change Management, perform one of the following steps.

- Select File.Change Management Check Out. When you build a working copy, Change Management is enabled in the working copy automatically.
- Select File.Change Management Enable. This will create the change set and begin the tracking process. After you enable Change Management, AionDS records all changes to the knowledge base.

Stopping Change Management (Disable)

What to do

- To disable Change Management, select File.Change Management Disable. Any changes made to the knowledge base when Change Management is disabled are not recorded.

Caution: Change Management can be disabled at any time. However, if you are using Change Management to integrate changes into a master knowledge base, you should never disable Change Management. If you do, all changes made while it is disabled are lost when you check your changes back into the master knowledge base.

Checking out a working copy with unapplied change sets

Overview

You might want to check out a new working copy before all the checked-in change sets have been applied to the master. When you check out a working copy and there are unapplied change sets in the master, AionDS automatically copies the unapplied change sets to your working copy.

To minimize potential problems with conflicts, apply the change sets to your working copy in the order in which they will be applied to the master. For example, suppose a change set contains a modification to Rule_1 in the knowledge base. If you apply the change set to your working copy before making changes, it will contain the new rule text. Then, if you modify the same rule, you will be making a change from the new rule text and not the old. When the change sets are applied to the master, your change will not overwrite the change contained in the other change set.

☑ *Tip:* Select multiple change sets from the Change Set list for a single Apply command; they are applied in order listed, one by one, each in its entirety. This is usually the most efficient method of applying change sets, as long as they have been checked for conflicts.

Updating out-of-sync working copies with the new master

Overview

Optimally, all outstanding change sets are applied to the old master at the same time and each person checks out a new working copy to make more changes. All working copies are synchronized with the new master. Sometimes a working copy's change set is not checked into the old master when the checked-in change sets are applied to the old master. Consequently, this working copy is out-of-sync with the new master. To prevent integration problems when you work on an out-of-sync working copy, update the out-of-sync working copy with the new master.

What to do

To integrate the out-of-sync working copy, complete the following steps:

1. Check out a new working copy from the latest version of the master.
2. Copy and apply the out-of-synch working copy's change set to the new working copy.
3. Delete the out-of-sync working copy.
4. Make your changes to the new working copy.
5. When you complete your changes, check both the out-of-sync and current change sets into the master.

Integrating change sets before applying them to the master

Overview

Verify changes made to individual change sets before applying them to the master. Also verify the combination of the various change sets affected before applying them to the master. To verify that no conflicts exist between change sets, copy and paste all change sets to one working copy and apply them.

For example, developers A and B complete and test their changes. Developer A wants to verify the combination of changes before incorporating any change sets into the master. The change set from Developer B's knowledge base is copied and pasted into Developer A's knowledge base and applied. Developer A can verify the end result before making updates to the master. If conflicts between the two change sets exist, Developer A can correct the situation in a third change set and check it into the master.

Copying, cutting, pasting, and deleting change sets

You can copy, cut, paste, and delete Change Sets just like other objects. This is helpful when you need to include in your working copy the changes made by another person.

When you copy or cut a change set, it is copied to the clipboard, so you can then paste it. Usually, you paste it into another knowledge base.

Caution: If you delete a change set, it is not copied to the clipboard.

What to do

To cut, copy, or delete a change set, perform the following steps:

1. Open the knowledge base that contains the change sets you wish to cut, copy, or delete.
2. Select File.Change Management List.
3. Select the change sets you wish to cut, copy or delete.
4. Select Edit.Cut, Edit.Copy or Edit.Delete Change Set. AionDS presents a confirmation dialog. Confirm the change sets you want to copy, cut, or delete, then click Copy, Cut, or Delete.

To paste a change set, perform the following steps:

1. Open the knowledge base in which to paste the changes.
2. Select Edit.Paste Change Set. The Paste dialog displays. Confirm the change sets to paste and click Paste.

Resolving change set conflicts

Overview

You can completely avoid conflicts only if you do not make changes to object properties which you know are changed in other outstanding change sets. To keep yourself from making changes to object properties that others have changed, check with others working on the same knowledge base. Also, frequently applying the checked-in change sets to the master. Checking out new working copies helps ensure that everyone working on the same knowledge base has the most current working copy.

Use the Compare command to compare change sets to for any conflicts. If conflicts exist, resolve the conflict depending on the type of conflict involved. Manual editing may be required to merge the conflicting changes effectively.

The Compare command generates a conflict report that indicates the object properties in conflict. To view the actual changed properties, print the change sets, then browse the generated file. To view the original properties, open an editor for the objects.

This section describes the procedures for determining the order of apply and repairs for the following types of conflicts:

- two change sets with a single conflict
- two change sets with multiple conflicts
- more than two change sets with multiple conflicts

What to do

To resolve the conflict, perform the following steps:

1. Determine which change set to apply first. At the same time, determine the repairs you need to make.

   📖 *For more information:* See "Determining the order of apply and repairs" on page 8-24.

2. Check out a working copy of the master.
3. Apply the change sets that contain the conflict to the working copy. Apply them in the order you determined in the first step.

4 Correct the objects that require repair and save the knowledge base.

5 Check in the change set to the master knowledge base. Delete the working copy.

6 Apply the conflicting change sets in the same order as before.

7 Apply the change set with the repairs.

Determining the order of apply and repairs

When you apply multiple change sets, the contents of the first change set are overlaid with the contents of the second change set. This can cause problems if any conflicts exist in the change sets.

Because of change set conflicts, the order in which change sets are applied can be important. For example, if Change Set #1 adds a significant amount of code to a state Agenda, while Change Set #2 contains a one-line change to the same, and both changes are desired, applying #1 after applying #2 results in a smaller repair effort.

For this reason, when multiple change sets must be applied, it is often desirable to control the order in which they are applied. When the desired order is not the listed order, apply the change sets one by one, using separate apply commands.

Two change sets with a single conflict

If there is only a single conflict, one of the following conditions exists:

- The change from one change set overlays the change in the other. This is the case if a change was made in error, and the second change set corrects the error in the first. This can also be the case if one developer applies changes from another change set and then modifies an object that had been previously modified. In other words, the change in the second change set includes the change from the first change set. This case is common.

  Apply the second change set after applying the first. No repairs are necessary.

- The changes from both change sets are required. This is the case when two people inadvertently modified the same property and both changes should be made to the master. This can also be the case if you do not apply change sets that exist in the master when you check out your working copy.

  The order of apply does not matter. The repair that must be made is contained in the first change set. It is a copy of the change that you made to the object that has the conflict.

Two change sets with multiple conflicts

If more than one conflict exists, one of the following conditions exists:

- All changes in the second change set should overlay the changes from the first change set. This is similar to the first case listed above.

  Apply the second change set after applying the first. No repairs are necessary.

- It is likely that some changes from each change set are required.

  Examine the change sets to determine which contains the fewer required changes. This is the change set you should apply first. If all changes from both change sets are required in the master, order is not important.

  In this case, the repair is contained in the change set you apply first. It is a copy of the change made to the objects that have conflicts.

More than two change sets with multiple conflicts

When there are more than two change sets to apply, the concepts used in determining which change set to apply are the same. However, the size of the problem grows very quickly because you will most likely need to make repairs for some of the changes in each change set.

For this reason, try to minimize the number of unapplied change sets.

Chapter 9
Debugging Knowledge Bases

Introduction

The debugger is a utility that you use to find and fix errors in a knowledge base. You can use the debugger to interrupt knowledge base processing, view runtime values and conditions, and execute KDL step by step. You can update the knowledge base while the knowledge base is running, but you cannot save any changes until after the knowledge base completes execution.

In this chapter

Continued

Continued

Debugging procedure

Overview

To correct the errors in a knowledge base effectively, complete the following procedure:

1. Run the knowledge base with the debugger.

   📖 *For more information:* See "Running a knowledge base with the debugger" on page 9-4.

2. Find and fix errors in the knowledge base.

   📖 *For more information:* See "Finding and fixing errors in the knowledge base" on page 9-7.

3. Periodically, abort the execution, save the changed objects and knowledge base, and run the knowledge base to verify that the errors are fixed.

   📖 *For more information:*

   - see "Aborting execution" on page 9-25
   - see "Saving objects" on page 9-26
   - see "Saving the knowledge base" on page 9-27
   - see "Rerunning the knowledge base to verify corrections" on page 9-28

To perform most debug operations, you execute the commands in the Run pull-down menu.

Running a knowledge base with the debugger

Procedure

To run the knowledge base with the debugger, complete the following steps:

1. Select File.Run. The Run window displays.
2. Select the Enable debugger radio button.
3. Click Run.

Result: The following actions occur:

- the knowledge base is executed
- the Run pull-down menu displays in the menu bar
- the Call stack object list displays
- the Breakpoints window displays
- the Trace window displays
- the execution stops at the beginning of the entry state agenda

Run menu summary

Summary The Run pull-down menu contains the following commands:

Step Into	executes current statement and stops on the first statement of called object
Step Over	executes current statement, executes called object, and stops on next statement in the current object
Pop	completes execution of current property and stops on object that called the current object
Go	executes knowledge base until a subsequent breakpoint is reached or it completes execution
Set Breakpoint	To set a breakpoint, specify an object property at which the debugger should stop when executing an application. First select the object on which to stop, then select Run.Set breakpoint.
Clear Breakpoint	Clear Breakpoint clears a previously set breakpoint. Multiple breakpoints can be cleared by selecting multiple objects and selecting Run.Clear-Break-Point. To quickly clear all breakpoints select Run.List Breakpoint, select Edit.Select All, and select Run.Clear Breakpoint.
List Breakpoint	List all the breakpoints that are currently set.
Call Stack	Display the current call stack. The default call stack view is a details view which will display the object, property and row/col information.

Trace	Calls the AionDS browser to display the current trace file.
Display	Displays the values of parameters, instances, and functions' local variables.
Application Window	Brings the application window to the foreground. This command is useful when the AionDS system has covered up the running application.
Abort	Terminates the knowledge base execution and returns control to the AionDS editing environment.
Restart	Aborts the current application and restarts its execution. This option is always disabled under Windows.
Interrupt	Interrupts the running application and forces a breakpoint. This option is always disabled under Windows.

Finding and fixing errors in the knowledge base

Overview

To find and fix errors in a knowledge base, examine the flow of control and the values generated by different objects. Using the debugger, you can suspend knowledge base execution on objects where you suspect errors might occur and step through those objects one statement at a time. You can also examine the execution trace, the call stack, and the parameter, instance, and the function values.

To suspend knowledge base execution on the objects that you think might contain errors, set breakpoints at the beginning of the objects. A **breakpoint** is a place in the knowledge base where execution is suspended. The Object editor displays. The property windows in which the statements exist are displayed. You can continue the execution in the object one statement at a time.

📖 For more information:

see “Suspending execution” on page 9-8

see “Continuing execution” on page 9-10

see “Displaying an object” on page 9-12

see “Displaying parameter, instance, and function values” on page 9-14

see “Displaying the execution trace” on page 9-15

see “Displaying the call stack” on page 9-18

see “Changing object properties” on page 9-19

see “Debugging objects in which errors most commonly occur” on page 9-20

Suspending execution

Overview

When execution is suspended you can perform debugging tasks. Execution is suspended at the default interruption points or breakpoints. You set breakpoints by using the Run.Set breakpoint. To examine a specific object, you should set a breakpoint at the beginning of the object. When execution is suspended at the breakpoint or default interruption point, you can execute the process one statement at a time.

Default interruption points

By default, execution is suspended when one of the following conditions occurs:

- the first statement of the entry state agenda is reached
- an action required from the user

Setting breakpoints

Specify an object property at which the debugger should stop when executing an application. Breakpoints are selected by first selecting the object on which to stop and then selecting Run.Set breakpoint.

You can set a breakpoint on an object, a property of an object, or a collection of objects.

On an object

To set a breakpoint on an object, complete the following steps:

1 Select the object for which to set a breakpoint.

2 Select Run.Set Breakpoint.

Result: The breakpoint is set on the object.

On a property

To set a breakpoint in a property, complete the following steps:

1 Open the object property for which to set a breakpoint.

2 Position the cursor on the statement to set a breakpoint.

3 Select Run.Set Breakpoint.

Result: The breakpoint is set on the property's statement.

On a collection of objects

To set breakpoints on a collection of objects, complete the following steps:

1 Select the objects for which to set breakpoints.

2 Select Run.Set Breakpoint.

Result: Breakpoints are set for those objects.

Deleting break points

To delete previously set breakpoints, complete the following steps:

1 Select the objects that have breakpoints in the object list window or the breakpoints in the Breakpoints window.

2 Select Run.Clear Breakpoints.

To quickly clear all breakpoints, complete the following steps:

1 Select Run.List Breakpoints. The Breakpoints window displays.

2 Select Edit.Select All.

3 Select Run.Clear Breakpoints.

Result: All breakpoints are deleted.

Listing break points

➢ To list all currently set breakpoints, select Run.List Breakpoints.

Result: The Breakpoints window displays.

Continuing execution

Overview

You can continue the execution of the knowledge base in one of the following ways:

- execute into the current statement and stop on the called object
- execute over the current statement and the called object, and stop on the next statement
- exit subprocessing
- go to the next breakpoint

Executing the current statement and stopping on the called object

➢ To execute into the current statement and stop on the first statement of the called object, select Run.Step Into.

☑ *Tip:* If the current statement does not call an object, the execution is stopped on the next statement.

Example

If you are debugging function A and the current statement calls function B, function B is called and execution is stopped on function B's first line. (This operates like Next in the character-based system.)

➢ To execute over the current statement and the called object, and stop on the next statement, select Run.Step Over.

☑ *Tip:* If the current statement does not call an object, the execution is stopped on the next statement.

Example

If you are debugging function A and the current line calls function B the system executes function B and stops on the next line of the current function (A). (This operates like Skip in the character-based system.)

Executing the current object and returning to the calling object

➢ To execute the current object and return to the calling object, select Run.Pop.

Result: Execution is stopped immediately after the call statement in the calling object.

☑ *Tip:* If the current object was not called by another object, execution continues until a breakpoint or default interruption point is reached, or an action is required of the user.

Example

If function A calls function B and you are currently in function B, selecting Pop executes function B and stops just after the call statement in function A.

Going to the next break point or default interruption point

➢ To go to the next breakpoint, default interruption point, or the point at which an action is required of the user, select Run.Go.

Displaying an object

Overview

Certain objects display properties that are only available when the debugger is running. The following table shows the properties that are available only when the debugger is running and the corresponding object:

Object	Properties
Rules	Bindings, Bindings and matches, Changed instances
Slots	Effect of Assignment
Parameter	Effect of Assignment
States	Delayed Rules, Forward Fire agenda, Pending list
Functions	Values property (locals, input and output parameters)

What to do

➢ To display an object, open the appropriate object editor.

Bindings property

The Bindings property is only available when pattern-matching rules are executed. The window shows sets of instances that match the criteria of a pattern-matching rule. Each Selected Bindings entry shows the following information:

Orderby	shows the integer expression from the orderby clause
Recency	shows the relative point at which an instance in a binding was changed. Higher numbers indicate that an instance in the binding has changed more recently than instances in a binding with a lower number
Inhibited	shows bindings blocked by an EXISTS clause

The following example shows a Selected Bindings entry:

```
-------------------------Selected Bindings--------------
----------
node_0001
flight_0001
------------------------------------------------------
----------
orderby:  2154                  Recency: 138  <<Inhibited
node_0003
flight_0020

orderby:  2257                  Recency: 138
node_0003
flight_0022
```

Changed instances property

The Changed instances property is available only if dynamic instances are used. The property lists changed instances. Often, the changed instances are newly created dynamic instances.

Matches property

The Matches property is only available if pattern-matching rules are used. The property lists all the instances that match the criteria specified in the pattern-matching rule at the time AionDS tries the rule. These instances match the criteria for individual instances. These instances are used by AionDS to form bindings, or groups of instances that together meet all the criteria in the rule premise. For example, the Matches property can show the dynamic instances of the node and flight classes that match the premise of the expand rule.

Effect of Assignment property

The Effect of Assignment property lists the rules associated with the sourced parameter or slot. These rules fire when the parameter or slot has been assigned a value.

Delayed Rules property

The Delayed Rules property lists the rules that are delayed because all the premises could not be met within the current scope.

Forward Fire Agenda property

The Forward Fire Agenda property lists the rules that fire when a forwardchain on-request statement is encountered in the Agenda.

Displaying parameter, instance, and function values

Overview

For parameters, the Value property displays in the Parameter editor. For instances and function, the Values property displays in the Instance and Function editors. For the Function editor Values property, the input arguments, output arguments, and local values properties display.

☑ *Tip:* To display these properties, change the editor template by setting up the Object editor property windows in the desired format, then select Settings.Set Window Template; and Settings.Save Settings.

🕮 *For more information:* See Chapter 10, "Customizing AionDS," to learn about setting up object editor templates.

What to do

To display parameter, instance, and function values, complete the following steps:

1. Select the parameter, instance, or function from the object list window.
2. Select Run.Display.

Displaying the execution trace

Overview

The Trace window shows the steps that the inference engine takes as it processes the knowledge base.

A trace consists of statements that outline the sections of the inference engine. You can search for a specific text string and copy parts of the trace to the clipboard to copy into another application or another window in AionDS.

📖 *For more information:* See *General Reference*, Chapter 19, "Trace File," for trace statement examples of inference engine processing.

Trace statements

Traces statements are divided into the following categories:

- action blocks
- actions

Action blocks

An action block consists of two separate trace statements that show the start and completion of an inference engine action. The following example shows an action block:

```
--evaluating twin
--twin failed
```

The inference engine starts processing the object twin at the first trace statement. The second statement shows the result of the processing, in this case, twin failed. An action block can be separated by other action blocks. Each indentation of a trace statement indicates that the action is a subgoal of the previous statement. Action blocks can thus be nested.

Action

An action is complete in itself. An example of an action would be the assignment of a value to a slot or parameter as shown in the following example:

```
-->>parameter is 5
```

Assignments are indicated by the double-headed arrow. The value assigned is located at the end of the statement, usually after the keyword is or are. In the example statement, 5 is the value assigned to the parameter. The parameter or slot name to which the value is being assigned is to the immediate right of the arrow.

What to do

- To examine the execution trace, select Run.Trace. The Trace window displays.

Copying part of the trace

To copy part of the trace, complete the following steps:

1. Select the text to copy.
2. Select Edit.Copy in the Trace window.

Result: The text is copied to the clipboard.

Searching for a specific text string

To search for a specific text string, complete the following steps:

1. Select Edit.Search from the Trace window.
2. Enter the text string for which to search in the Search string field.
3. Select the Case-sensitive check box to search for the string exactly as you entered it including upper and lower case.
4. Select the Wrap check box to start the search at the beginning of the trace when the end of the trace is reached.
5. Click OK.

Result: AionDS finds the first occurrence of the specified text string after the cursor.

Searching for next occurrence

➢ To search for the next occurrence of the same text string, select Edit. Search Again from the Trace window.

🔔 *Reminder:* If you selected the Wrap check box in the Search dialog, when you select Search Again at the end of the trace, AionDS starts the search at the beginning of the trace.

Displaying the call stack

What to do

- To examine the call stack, select Run.Call stack. The Call stack window displays.

The default call stack view is a detailed view which displays the object, property and row/col information. The view can be changed to a graphical view for a graphical call stack. The graphical call stack only shows the names of the objects.

Changing object properties

Overview

You can make changes to object properties while running a knowledge base. However, to save those changes first abort or complete the execution.

📖 *For more information:* See "Saving objects" on page 9-26 and the "Saving the knowledge base" on page 9-27.

What to do

To change an object property, complete the following procedure:

1. To open the object, double-click the object in the object list window. The object editor displays.
2. Open the property by selecting Property.Open from the Property window. The Open Property window displays.
3. Double-click the property to change. The property window displays.
4. Make the changes to the property.

Debugging objects in which errors most commonly occur

Overview

This section describes debugging the objects in which errors most commonly occur. The following objects are described:

- parameters and slots
- immediate rules
- on-request rules
- pattern-matching rules

Debugging parameters and slots

What to do

To find out how a parameter or slot received its value or why it did not receive a value:

1. Set a breakpoint on the Sourcing property of the parameter.
2. Execute the Go command. AionDS stops at the parameter.
3. Execute the Step Into command repeatedly. AionDS examines each of the following possible sources to get a value for the parameter or slot:
   - WhenNeeded demon
   - Facts property of the parameter
   - User sources
   - System sources
   - Default

Debugging immediate rules

What to do

To find out why an immediate rule did or did not execute:

1. Set breakpoints on the Effects list for every parameter or slot in the premise of the rule. The Effects list shows the rules affected by the assignment of the parameter.
2. Execute the Go command and wait for the Effects list to appear.
3. Execute the Step Into command repeatedly to step through the list. When AionDS encounters an Immediate rule, it fires the rule. When the rule is fired, the debugger places you in the Rule Definition property of the rule.
4. Execute the Display command on the parameters in the premise of the rule to check the parameter values. The Display command shows a description of the current state of the parameter. When the premise of the rule is met, the rule fires. If the rule is out of scope, it cannot fire.
5. If you encounter an out-of-scope rule on the Effects list, set a breakpoint on the Delayed Rules property of the state that owns the rule. When AionDS enters the state, the debugger displays the Delayed Rules list. AionDS steps through the list one item at a time and attempts to fire the rules.

Debugging on-request rules

What to do

To debug the behavior of on-request rules:

1. Set a breakpoint on the Forward Fire Agenda. The Forward Fire Agenda shows a list of rules and their priority. The higher priority rules are tried first.
2. If the rule does not appear on the Forward Fire Agenda, set a breakpoint on the Effects list of the parameters present in the premise of the rule. When the parameter is assigned and AionDS encounters the rule in the Effects list, AionDS adds it to the Forward Fire Agenda.

Debugging pattern-matching rules

What to do

To debug a knowledge base with pattern-matching rules, use the following procedure:

1. Set a breakpoint on the Changed Instances property of the rule.
2. Set a breakpoint on the Bindings and Matches property of the rule.
3. Execute the Go command.

   The execution proceeds to the first breakpoint, the Changed Instances property.
4. Check to see that the appropriate instances are listed.

   An ifmatch rule considers the set of changed instances and determines if the rule can be fired with any of them. When AionDS encounters the rule, it shows the list of changed instances as well as the rule's source text.
5. Execute the Step Into command.

   The Bindings and Matches windows display. The Bindings and Matches windows show you a complete list of matches made, a list of successful binding sets, and the selected bindings.
6. Check whether the appropriate bindings were made.

Aborting execution

What to do

- To abort the execution, select Run.Abort.

Result: The execution is aborted and you are returned to the development environment.

Saving objects

What to do

To save objects that you change while running the knowledge base, perform the following procedure:

1. After you make the changes to the objects, abort or complete the execution.
2. Select the appropriate object editor.
3. Select Object.Save. A confirmation pop-up dialog displays.
4. Click Save.

Result: The object is saved.

Repeat steps 2 to 4 for each object you want to save.

Saving the knowledge base

What to do

To save changes made to a knowledge base while it is running:

1. Abort or complete the knowledge base execution.
2. Select File.Save.

Result: The knowledge base is saved.

Rerunning the knowledge base to verify corrections

What to do

- To verify any corrections made, run the knowledge base again by selecting File.Run.

Chapter 10
Customizing AionDS

Introduction

In this chapter, you'll learn how to customize AionDS windows, the development, runtime, and system environments and database interfaces, and how to manage profiles.

The commands are available in the Settings pull-down menu. The Settings pull-down menu is divided into three groups of commands separated by horizontal lines.

- Group 1 consists of Fonts, Display, Set window template, and Save settings. These settings control how AionDS windows display.
- Group 2 consists of Environment, Database, Run, Run files, Run programs, and System settings. These settings control the AionDS profile.
- Group 3 consists of commands to save and load the AionDS core profile.

In this chapter

Customizing AionDS windows

Summary

Use the commands from the first group of the Settings pull-down menu to control how AionDS windows display. The values changed using these commands are saved in the AION.INI file. These commands are:

Fonts	controls the fonts that AionDS uses to show you text in various windows; the Font Settings window displays
Display	controls the tab width, auto-save interval, 3-D appearance of windows, and the button bar and message bar locations; the Display Settings window displays
Set window template	sets the layout of Object editor windows, object list windows, and the Window editor
Save settings	saves changes to window settings in the AION.INI file

Changing screen fonts

Overview

It is convenient to have different windows display different fonts. When fonts differ, you can quickly tell where the information in a window comes from.

For example, suppose you have an object editor open, with the Used By and the Message Text properties opened, as shown in the following example. If the fonts are different, you can visually determine that AionDS supplies the text for Used By, while a developer supplies the text for Message Text.

Types of fonts

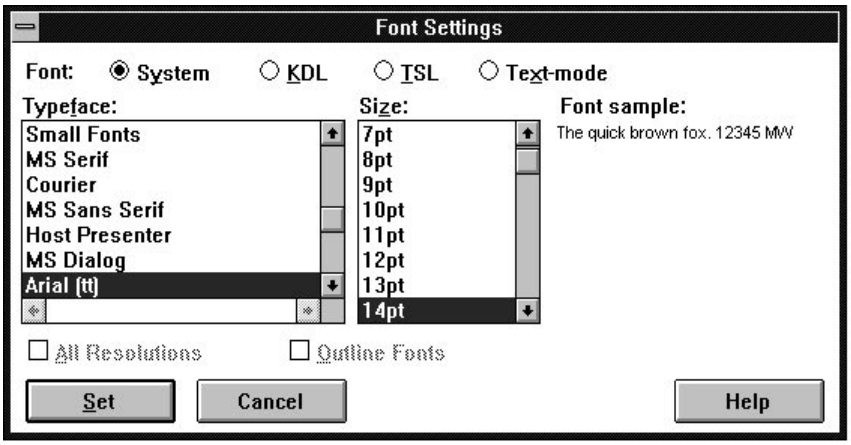
You can use the Font Settings window to change the following types of fonts:

System	represents the font in any window where AionDS presents the information about a knowledge base, such as system-supplied list boxes and the message bar
KDL	represents the font in any window where you enter KDL, such as the Rule Definition property of a rule, or the Facts window of a slot
TSL	represents the font in any window where you enter TSL, such as the Message Text property of a Message or the Prompt property of a Parameter or Slot
Text-mode	represents the font in the **consultation** window; a consultation is a knowledge base execution

What to do

To change the fonts for displaying text in windows, perform the following steps:

1. Select Settings.Fonts. The Font Settings window displays.
2. Select a Font radio button to indicate which type of font to change.



3 Select the style and size of the font in the Typeface and Size list boxes. The Font Sample box shows you a sample of the currently selected font.

   Reminder: To maintain compatibility, GUI AionDS has the same property widths as character-based AionDS. Text in objects such as rules and messages can only be 50 characters wide for KDL text and 76 characters wide for TSL text. Because of this limitation, we recommend that you use 10-point System monospace type for the KDL and TSL properties.

4 Check All Resolutions to list all fonts available to your system along with size in points (and character size in parentheses).

5 Check Outline to list all fonts available to your system, including outline fonts. The outline fonts have an asterisk next to them in the Typeface list box.

6 Click Set.

TextWindows using TSL

TextWindows using TSL display with the System monospaced font by default. To override this default.

1 Create your text windows as instances of a subclass of the TextWindow class.

2 Specialize the WhenOpened method to set the font to proportional.

3 Include the following in the WhenOpened method:

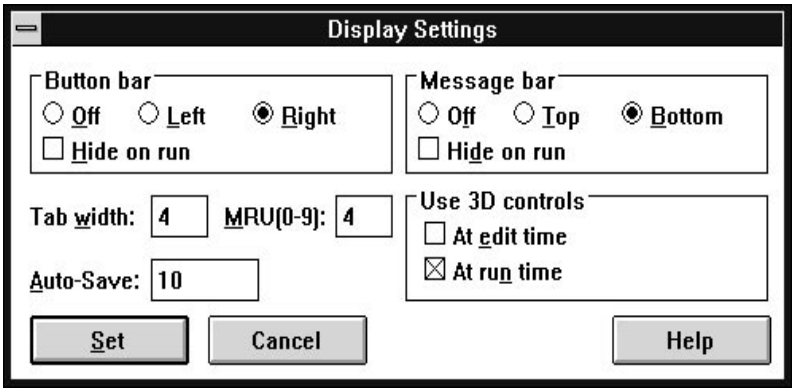
```
GetFontInfo (
Typeface,FontStyle,
Width,Height)

SetFontInfo (
FN_SYSPROPORTIONAL,
Fontstyle,Width,
Height)
```

Customizing the message and button bars

What to do

You can control whether the message bar and button bar are on or off and where they display on the screen. To set these options:

1 Select Settings.Display. The Display Settings window displays.



2 In the button bar group, click one of the following radio buttons:

Off	does not display the button bar
Left	displays the button bar on the left side of the screen
Right	displays the button bar on the right side of the screen

3 In the Message bar group, click one of the following radio buttons:

Off	does not display the message bar
Left	displays the message bar on the left side of the screen
Right	displays the message bar on the right side of the screen

4 To hide the button bar and the message bar when you run a knowledge base, check Hide on run.

5 Click Set.

Removing the message and button bars with the mouse

To remove the message bar or button bar from the screen with the mouse, double-click on any part of the message bar or a place on the button bar where no icons display.

Displaying windows with a 3-dimensional appearance

What to do

You can display windows with a 3-dimensional appearance. To display windows with a 3-dimensional appearance:

1. Select Settings.Display. The Display Settings window displays.
2. In the Use 3-D controls group, check At runtime to display AionDS windows with a 3-dimensional appearance when running a knowledge base. Check At edit time to display AionDS windows with a 3-dimensional appearance when you edit a knowledge base.

   Note: All AionDS/2 applications use 3-dimensional controls as Version 2.0 of OS/2. AionDS/2 does not have the check box to enable 3-D controls at runtime.

   In the Window editor, when you edit windows that display at runtime, they display in a 3-dimensional appearance only if you check At runtime.
3. Click Set.

Customizing the auto-save time

What to do

You can customize the auto-save function. At regular intervals, **auto-save** prompts you to save work in progress to disk. You can control the number of minutes elapsed before you are prompted to save your work. If you do not enter a number, auto-save is not activated. To set the auto-save time:

1. Select Settings.Display. The Display Settings window displays.
2. In the Auto-Save field, enter the number of minutes to elapse before AionDS prompts you to save work in progress to disk.
3. Click Set.

Customizing the most recently used knowledge base list

What to do

You can list up to nine recently used knowledge bases in the File pull-down menu by performing the following steps:

1. Select Settings.Display. The Display Settings window displays.
2. Enter the number of knowledge bases to display in the Most Recently Used (MRU) field.
3. Click Set.

Customizing the tab width

What to do

You can control **tab width—**how many spaces are inserted in a text-editing window when you press the Tab key. To set the tab width:

1. Select Settings.Display. The Display Settings window displays.
2. Enter the number of spaces to insert with the Tab key.
3. Click Set.

Customizing window size and position

Overview

You can customize the size and position of an Object editor and property windows, object List windows, and the Window editor. To customize and save a window's size and position, save it to a template. You can set a window template for each type of window. For example, suppose you customize the Class editor window and save it as a template. The next time you open the Class editor, the Class editor window has the same property windows open. The property windows are the same size and in the same place within the Class editor, and the state of the scroll bars is the same.

What to do

1. To set a window template, make the window active and customize it.
2. Select Settings.Set window template. The next time you open another window of the same type, it looks the same as when you saved it.

   📖 *For more information:* Setting a window template during an AionDS session does not save the changes after you close AionDS. To save changes between sessions of AionDS, select Settings.Save. See "Saving customized settings" on page 10-13.

✋ *Caution:* To maintain compatibility, GUI AionDS has the same property widths as character-based AionDS. Text in objects such as rules and messages can only be 50 characters wide for KDL text and 76 characters wide for TSL text. Because of this limitation, you should customize the Object editor templates at 50 characters wide for KDL properties and 76 characters wide for TSL properties.

Saving customized settings

What to do

➢ To save your settings before exiting AionDS, select Settings.Save.

Result: The settings are saved in the AION.INI file.

When you exit AionDS, if you changed settings, you are prompted to save your settings.

➢ To save the current settings, click Save in the pop-up confirmation window.

Result: The current settings are saved until the next time you change them.

☑ *Tip:* You can distribute a standard set of default window settings by copying AION.INI from machine to machine.

Customizing profiles

Overview

A profile contains specifications for the development, runtime, and system environments, and database interfaces. Use the group 2 commands in the Settings pull-down menu to change the current profile. Each command displays a window in which you change profile information. Select Settings.Save core profile to write the changes to a specified profile file or create a new one. Select Settings.Load core profile to replace the current profile with another profile.

Default profile names

The following tables show the default profile file names for AionDS and AionES.

AionDS

Platform	System profile	User profile	KB profile
OS2	profsys.ads	profile.ads	*kb_name*.*vll*/ *kb_name*.pro
WIN	profsys.ads	profile.ads	*kb_name*.*vll*/ *kb_name*.pro

kb_name name of the knowledge base

vll version number

AionES

Platform	System profile	User profile	KB profile
OS2	profsys.aes	profile.aes	*kb_name*.pro
WIN	profsys.aes	profile.aes	*kb_name*.pro

kb_name name of the knowledge base

Loading profiles

Overview

AionDS profiles are loaded automatically in a specific order when you start AionDS and execute a knowledge base. You can also load a new profile at any time.

Automatically loaded profiles

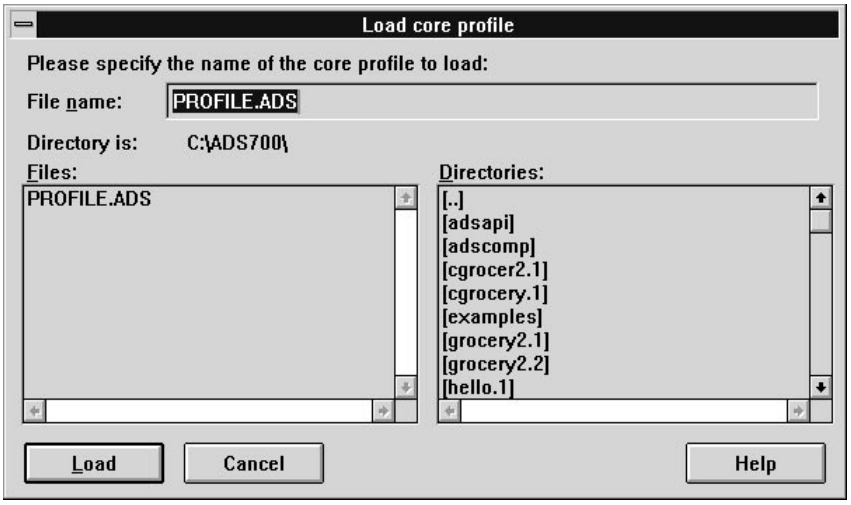
The following three profiles are loaded automatically. The first two profiles, PROFSYS.ADS and PROFILE.ADS, must be available in a directory in the environment variable DPATH when you start AionDS.

1. AionDS loads the system profile PROFSYS.ADS.
2. AionDS loads the user profile PROFILE.ADS.
3. AionDS loads the knowledge base profile *kbname*.PRO when knowledge base execution begins, if the following conditions exist:
   - *kbname*.PRO is available in the directory that stores a knowledge base, where *kbname* is the name of the open knowledge base
   - Load KB profile is checked in the Run window

When any profile option is specified in more than one of these three types of profiles, the last one to load takes precedence. When these files are not available, AionDS uses profile defaults that are shipped as part of AionDS.

Manually loading a profile

1 At any time, you can load any valid profile file by selecting Settings.Load core profile. The Load Core Profile window displays.



2 Select the path and enter the file name of the profile to load.

3 Click Load.

Result: The selected profile is loaded and replaces the current profile.

When changes take effect

Changes to profile options in use when AionDS is running do not take effect immediately. For example, you cannot change the current log file by modifying the LOG-FILE option, because the file is already open. For the change to take effect, save the change to a profile file and restart AionDS.

When you run a knowledge base, AionDS overrides the current values with values present in the knowledge base profile for the duration of the knowledge base execution. When the knowledge base execution terminates, the profile reverts to the default, system, and user profile options.

Development system and execution system

AionDS uses different resources in the development system than in the execution system. For example, the trace file is used by the execution system but not by the development system. You can change the trace file in the development system and the change takes effect when you run the knowledge base. However, changing the trace file when you are running a knowledge base does not affect the current execution because the trace file is already open.

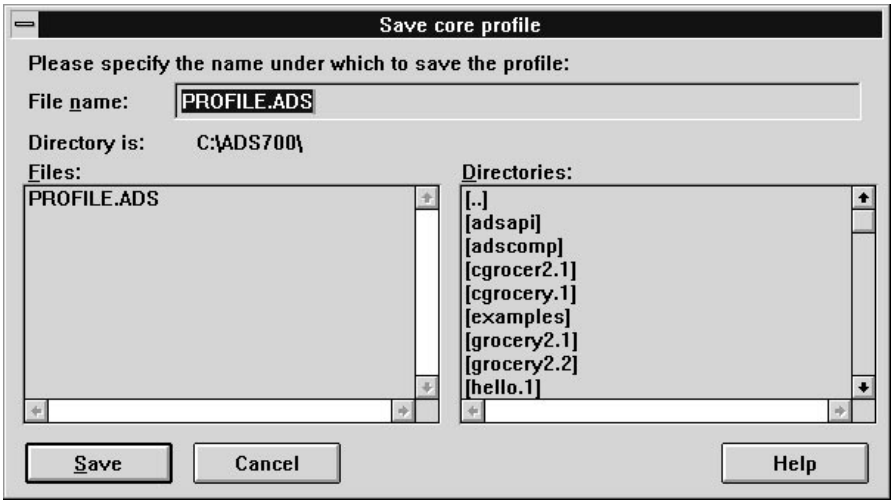
Saving profiles

Overview

You can save the current AionDS profile settings at any time. If you save a profile as the standard names PROFSYS.ADS or PROFILE.ADS in a directory available in the AION or DPATH environment variable, the options saved in these files are reflected the next time you open AionDS. If you save a profile as the file *kbname*.PRO (where *kbname* is the name of the knowledge base) in the directory where the knowledge base is stored, the next time you run the knowledge base *kbname*, those profile characteristics are loaded. If you save the profile with another name, load that profile using the Load core file command in order for the options to take effect.

What to do

1 To save the current profile settings, select Settings.Save core profile. The Save Core Profile window displays.



2 Select the path and enter the filename of the profile to save.

3 Click Save.

Result: The current settings are saved to the specified profile file.

Creating and editing profiles using a text editor

Overview

You can also create and edit a profile file using a text editor, since it is saved in ASCII text format. You might need to use a text editor to change a profile, because some profile options cannot be changed using the Settings pull-down menu.

📖 *For more information:* See "Profile options not available from the Settings pull-down menu" on page 10-40.

What to do

1. Create a new text file using a text editor.
2. Enter each attribute to set on a new line. You need not specify an attribute if you accept the default value.

   ☑ *Tip:* AionDS starts more quickly when you specify only the profile options that override default options.
3. Save the text file.

🔔 *Reminder:* To load the user profile (PROFILE.ADS) or system profile (PROFSYS.ADS) automatically when you start up AionDS, save the user or system profile in a directory in your AION or DPATH path variable.

To load a knowledge base profile (*kbname*.PRO) automatically when you run the *kbname* knowledge base, save the profile in the directory where the knowledge base is stored.

Example

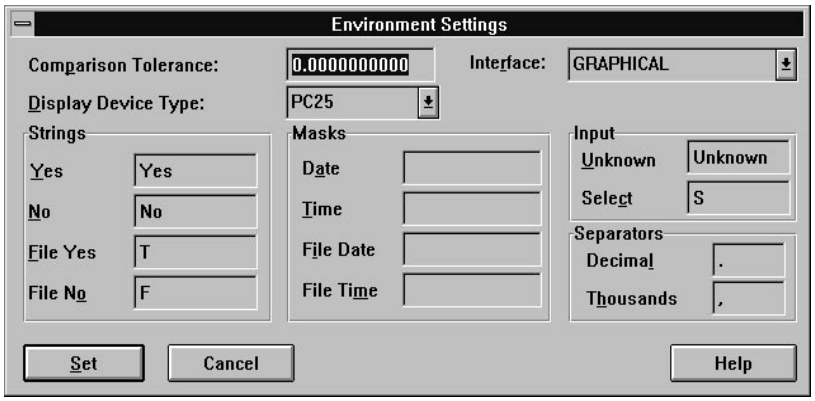
The following is an example of a user profile:

```
Auto-Export           = 1
Auto-Store-File       = ANS
Log-File              = LOG
Trace-File            = TRACE
Trace-Level           = 4
```

Changing environment settings

What to do

1 To change the environment settings, select Settings.Environment. The Environment Settings window displays.



2 Change the desired settings.

3 Click Set.

Result: The current environment settings are replaced.

Comparison Tolerance

Comparison Tolerance specifies the tolerance used in the comparison of real numbers. The default is 0.0000000000.

AionDS stores real numbers with a floating-point representation, which can cause numbers to be stored in a slightly different way than they are entered in the knowledge base. For example, .10 may be stored as .09999998. The Comparison Tolerance option specifies the threshold within which AionDS considers real numbers to be equal.

On the mainframe, AionDS also uses the comparison tolerance when displaying real numbers in the Consultation monitor. For example, with a comparison tolerance of .0001, AionDS displays the number .09999998 as .10.

Setting the Comparison Tolerance option ensures that comparisons are compatible in different environments. For example, you can set this option to .0001 for compatibility between the PC and the mainframe.

This field sets the COMPARISON-TOLERANCE profile option.

Interface

Interface indicates the type of user interface that knowledge bases should use at runtime and edit time.

This option can have one of the following values:

GRAPHICAL	knowledge base should be built and executed to use a graphical user interface (GUI) using the GUI class library 📖 *For more information:* See the *Building GUI Applications: Reference* to learn more about graphical knowledge bases.
CHARACTER	knowledge base should be built and executed to use a character mode interface 📖 *For more information:* See *User's Guide (Character-based)* for details about character-based knowledge bases.

This field sets the USER-INTERFACE profile option.

Display Device Type

Display Device Type specifies the kind of terminal that AionDS emulates. The choices are:

Device type	Terminal	Rows	Columns
*	all	any	any
PC*	PC	any	any
PC25	PC	25	80
PC43	PC	43	80
PC50	PC	50	80
M*	3270	any	any
M2	3270	24	80
M3	3270	32	80
M4	3270	43	80
M5	3270	27	132

For a character-based knowledge base, you can use the * setting to edit all windows of a display with the Screen and Window Properties editors. This setting is not recommended for running knowledge bases.

Your hardware must support the device you specify.

This field sets the DEVICE-TYPE profile option.

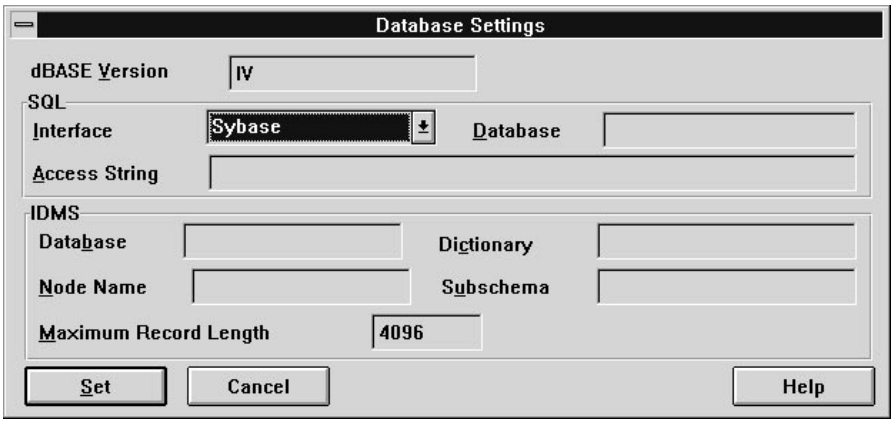
Strings	Yes	specifies the string to use when AionDS accepts or displays a true Boolean value during user sourcing This field sets the YES-STRING profile option
	No	specifies the string to use when AionDS accepts or displays a false Boolean value during user sourcing This field sets the NO-STRING profile option
	File Yes	specifies the string to use for disk I/O when AionDS reads or writes a TRUE Boolean value in textual format This field sets the FILE-YES-STRING profile option
	File No	specifies the string to use for disk I/O when AionDS reads or writes a false Boolean value in textual format This field sets the FILE-NO-STRING profile option
Mask	Date	specifies the default mask applied to dates for user input. You can specify any valid date mask. The default is blank. If the date option is blank, AionDS uses the mask d-m-y. AionDS only uses the date option when the base type of an object does not have an associated mask. This field sets the DATE-MASK profile option.
	Time	specifies the default mask applied to times for user input. You can specify any valid time mask. The default is blank. If the time option is blank, AionDS uses the mask H:M:S.F. AionDS only uses the time option when the base type of an object does not have an associated mask. This field sets the TIME-MASK profile option.
	File Time	specifies the mask to use during I/O when AionDS reads or writes a time in textual format This field sets the FILE-TIME-MASK profile option.
	File Date	specifies the mask to use during I/O when AionDS reads or writes a date in textual format This field sets the FILE-DATE-MASK profile option.

Input	Unknown	contains the string that AionDS recognizes as an "unknown" response. The default unknown string is "Unknown in AionDS" This field sets the UNKNOWN-INPUT profile option.
	Select	specifies the character string to use to select values in selection-input windows. The string cannot contain more than five characters This field sets the SELECT-INPUT profile option.
Separator	Decimal	specifies a single character to display as the decimal point in a real number This field sets the DECIMAL-SEPARATOR profile option.
	Thousands	specifies a single character to display to separate thousands (000) in a number This field sets the THOUSANDS-SEPARATOR profile option

Changing database settings

What to do

1 To change the database settings, select Settings.Database. The Database Settings window displays.



2 Change the desired settings.

3 Click Set.

Result: The current database settings are replaced.

dBase Version

dBase Version specifies the version of the dBASE software used with AionDS. The default is IV.

This option can have one of the following values:

III specifies use of dBASE III

IV specifies use of dBASE IV

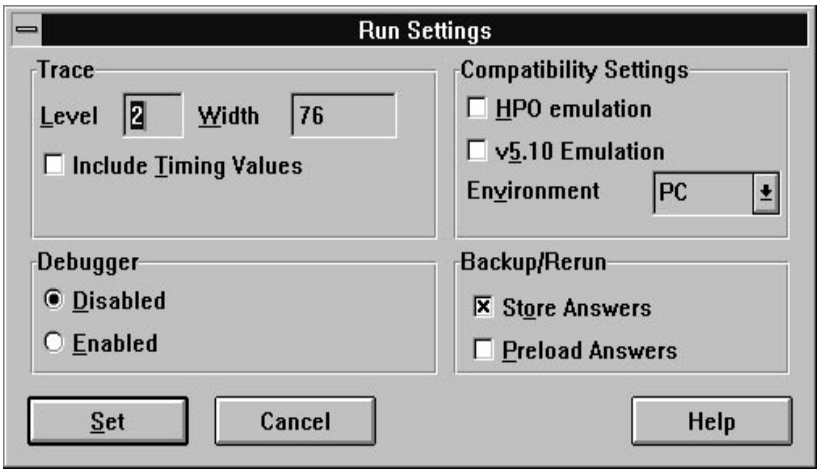
This field sets the DBASE-VERSION profile option

SQL	Interface	specifies the default SQL access method In AionDS, this option can have one of the following values: IBM — IBM's DB2/2 is the SQL access method ORACLE — Oracle for OS/2 is the SQL access method SYBASE — Microsoft/Sybase SQL Server is the SQL access method ODBC — You use the Open Database Connectivity (ODBC) interface to connect to any ODBC-compliant database DB2 — IBM Database 2 (DB2) is the SQL access method (requires the Cooperative Processing Option) This field sets the SQL-INTERFACE profile option
	Access String	defines the default access string used to establish an SQL database connection This field sets the SQL-ACCESS-STRING profile option
	Database	specifies the default database name used to establish an SQL connection This field sets the SQL-DATABASE profile option

Changing run settings

What to do

1 To change the run settings, select Settings.Run. The Run Settings window displays.



2 Change the desired settings.

3 Click Set.

Result: The current run settings are replaced.

Trace controls

If you delete a trace file while a knowledge base is running, AionDS creates another trace file.

Trace options

The Level field specifies the default trace level. This option can have one of the following values:

0 trace file not available. Execution Trace in the Run window is set to off. When running character-based applications in AionDS/2, the explanation facility is not available

1 trace file not available. Execution Trace in the Run window is set to Explanation only. When running character-based applications in AionDS/2, the Explanation facility is available

2, 3 trace file available. Execution Trace in the Run window is set to On. When running character-based applications in AionDS/2, the Explanation facility is available

4 The trace file displays on screen. When running character-based applications in AionDS/2, the Explanation facility is available. Execution Trace in the Run window is set to On

This field sets the TRACE-LEVEL profile option

Width controls the width of the trace output. The default is 76. If the output is greater than the Width specification, it is wrapped onto the next line and indented. A Width setting of zero specifies no wrapping. You should set the Width to zero when using the Include Timing Values option

This field sets the TRACE-PAGE-WIDTH profile option

Include Timing Values

controls whether timing information is written to the trace file. The default option is off (unchecked). If unchecked, Timing information is not written to the system log file

If checked, each entry in the trace file is preceded by a system time stamp. The time displays in milliseconds

This field sets the TRACE-TIMING profile option

Trace and the AionDS profile

If the trace file name in the Run Files window has pound signs (#) in the filename, AionDS creates a unique trace file for each knowledge base execution and for each session of AionDS running concurrently.

If the trace file name in the Run Files window is blank, AionDS does not produce a file even if the trace is turned on.

Debugger

Debugger controls whether the user can enter debug mode during knowledge base execution. The default is Disabled.

This option can have one of the following values:

Disabled cannot enter debug mode during knowledge base execution

Enabled can enter debug mode during knowledge base execution

This field sets the DEBUG-MODE profile option.

Compatibility Settings

HPO emulation controls whether checking for HPO compatibility is performed during knowledge base development. The default is OFF (unchecked). If unchecked, HPO compatibility restrictions are not applied to knowledge base development and execution. If checked, HPO compatibility restrictions are applied to knowledge base development and execution.

This field sets the HIGH-PERFORMANCE profile option.

v5.10 Emulation — specifies whether sourcing operations by the inference engine are compatible with Version 5.10 and 5.11 of AionDS. See Appendix E of the *General Reference* for more information. If you change this option, re-import your knowledge base to ensure correct operation.

This field sets the 510-COMPATIBILITY profile option.

Environment — specifies the compatibility of AionDS on the PC with AionDS on the mainframe. This option can have one of the following values:

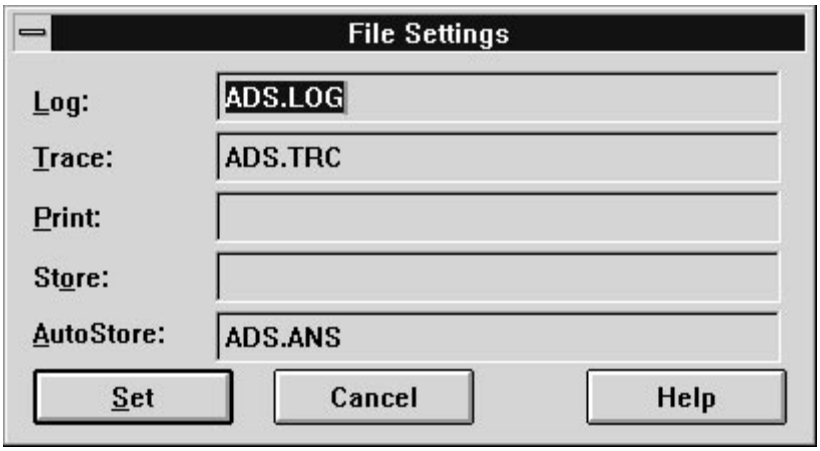
- OS — AionDS on the PC emulates the operation of AionDS on the mainframe, to the extent possible
- PC — AionDS on the PC takes advantage of features unique to the PC

This field sets the COMPATIBILITY profile option.

Changing file settings

What to do

1 To change the log, trace, print, store, and auto-store files written by AionDS, select Settings.Run Files. The Files Settings window displays.



2 Change the desired settings.

3 Click Set.

Result: The current file settings are replaced.

Log

specifies the name of the system log file. For AionDS, the default is ADS#####.LOG. A unique log is written for each instance of AionDS.

The Log filename is used only when AionDS is started. If you change the Log filename, you have to restart AionDS for the change to be applied.

The value of the Log field is overridden if the ADSLOG environment variable exists on the PC.

This field sets the LOG-FILE profile option.

Trace specifies the file that stores the trace output. For AionDS, the default value for this option is ADS#####.TRC. A unique file is written for each instance of AionDS. If you do not specify a file in this field, the Execution Trace controls in the Run window are not available.

The Trace field is overridden if you assign a file for the ADSTRACE environment variable on the PC.

This field sets the TRACE-FILE profile option.

Print specifies the file name that receives the contents of the PRINTWINDOW command when it is issued from the Consultation monitor. Component AionES also uses this file to print messages when you do not specify SHOW-PROGRAM.

The value of the Print field is overridden whenever an ADSPRINT environment variable exists on the PC.

This field sets the PRINT-FILE profile option.

Store the file name used by the Store command to store responses for a character-based consultation. BACKUP-ENABLE must be on. The Store field is overridden if an ADSRERUN environment variable exists on the PC.

This field sets the STORE-FILE profile option.

Auto Store the file name used by the Store command to store the responses to a character-based consultation. BACKUP-ENABLE must be on. For AionDS, the default is ADS#####.ANS

The Auto Store option is overridden if an ADSSTORE environment variable exists on the PC

This field sets the AUTO-STORE-FILE profile option

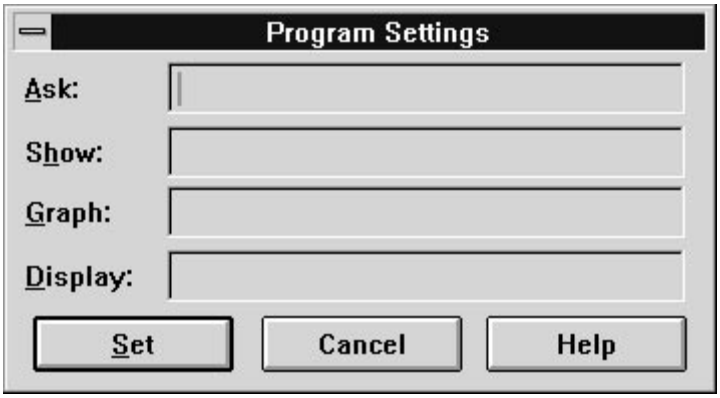
Changing run program settings

Overview

The run program settings are available in GUI AionDS but only apply to running character-based knowledge bases.

What to do

1 To change the programs that replace parts of AionDS, select Settings.Run Programs. The Run Programs Settings window displays.



2 Change the desired settings.

3 Click Set.

Result: The current run program settings are replaced.

Ask

file name that specifies the program that replaces the Consultation monitor when the user is asked for input

This field sets the ASK-PROGRAM profile option

☑ *Tip:* If you do not enter a program name for ASK, user input is disabled—simulating CAES

Show specifies the program that replaces the Consultation monitor during the display of a message

☑ *Tip:* If you do not enter a program name for Show, message displays are disabled for an entire consultation—simulating CAES

This field sets the SHOW-PROGRAM profile option

Graph specifies the program that AionDS uses instead of the default graphic interface to display a graph

This field sets the GRAPH-PROGRAM profile option

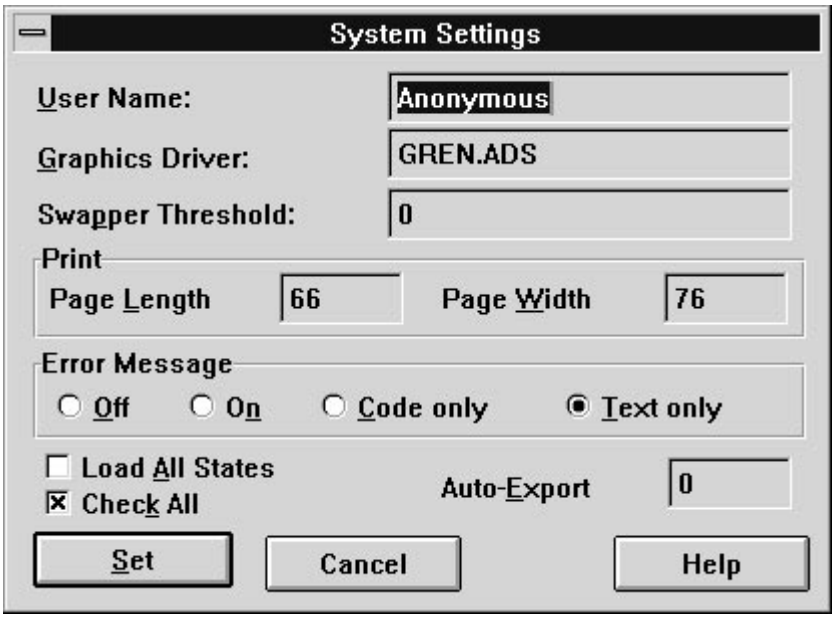
Display specifies the program to execute when AionDS sources a Group object. If you set Display to NUL, groups are not displayed

This field sets the DISPLAY-PROGRAM profile option

Changing system settings

What to do

1 To change the system settings, select Settings.System. The System Settings window displays.



2 Change the desired settings.

3 Click Set.

Result: The current system settings are replaced.

User Name

specifies the name of the user. The default is the name of the last user to perform a save or export. If the User Name is blank, AionDS prompts you for a user name during the first knowledge base operation

This field sets the USER-NAME profile option

Graphics Driver specifies the name of the graphics driver. For AionDS graphical development, use GRPM.ADS. Without this as the driver, AionDS graph objects are not displayed when running under AionDS. For other environments, use one of the following drivers

GRCO.ADS	(with the letter 'O', not zero) specifies the IBM Color Graphics Adapter (CGA) or compatible board. Graphs display in red, yellow, and green, on a blue background.
GRCOI.ADS	(with the letter 'O', not zero) specifies the IBM Color Graphics Adapter (CGA) or compatible board. This driver is recommended for monochrome monitors. Graphs display in purple, white, and blue, on a black background.
GREN.ADS	specifies the Enhanced Graphics Adapter (EGA), or compatible board, with at least 128KB on-board memory. This driver is only used with enhanced color display monitors. Graphs display in 16 colors.
GREN0.ADS	(with zero, not the letter 'O') specifies the Enhanced Graphics Adapter (EGA), or compatible board, with at least 128KB on-board memory. This driver is only used with enhanced color display monitors. Graphs display in 16 colors with a blue background.
GRENL.ADS	specifies the Enhanced Graphics Adapter (EGA) or compatible board. This driver is used with either color display monitors or enhanced color display monitors. Graphs display in 16 colors.
GRENB.ADS	specifies the Enhanced Graphics Adapter (EGA) or compatible board. Use this driver with monochrome monitors. Graphs display in black and white.

GRVGA.ADS	specifies the Video Graphics Array (VGA) or compatible board and a VGA-compatible monitor. Graphs display in 16 colors.
GRVGA0.ADS	(with zero, not the letter 'O') specifies the Video Graphics Array (VGA) or compatible board and a VGA-compatible monitor. Graphs display in 16 colors with a blue background.
GRMCGA.ADS	specifies the Multi-Color Graphics Array (MCGA) or compatible board and an MCGA-compatible monitor. Graphs display in black and white.

This field sets the GRAPHICS-DRIVER profile option

Swapper Threshold defines the number of bytes (in multiples of 1024K bytes) that is the upper limit of the amount of physical memory that will be used to store the knowledge base in AionDS. After this threshold is reached, parts of the knowledge base not in use are swapped to disk. If Swapper Threshold is 0, AionDS dynamically determines the amount of available memory and uses all of it. The default value is 0

This field sets the SWAPPER-THRESHOLD profile option

Print Page Length specifies the number of lines to print on a page. The default is 66. This field sets the PRINT-PAGE-LENGTH profile option

Page Width specifies the width of the print output. The default is 76. This field sets the PRINT-PAGE-WIDTH profile option

If the width of the print output is greater than the specified Page Width, the output is wrapped onto the next line. A Page Width setting of zero specifies no wrapping.

Error Message specifies the content of the display when an error occurs. The default radio button is Text only

Select one of the following radio buttons:

On	both the AionDS error code and error text display
Off	AionDS error information is never displayed. This value is not recommended
Code Only	only the AionDS error code displays
Text Only	only the AionDS error text displays. AES_ERRORNUM is set to the numbers listed in the Aion Development System Messages and Codes

When this field is set to On, Off, or Code Only, AES_ERRORNUM is set to the number of the message in the message file

This field sets the ERROR-MESSAGE profile option

Load All States controls whether AionDS loads all contexts (states, classes, and vocabularies) when knowledge base execution begins. The default is Off (the check box is deselected)

If you deselect the check box, only the entry state is loaded when you run the knowledge base—other contexts are loaded as they are needed

If you select the check box, all contexts are loaded when knowledge base execution begins

This field sets the LOAD-ALL-STATES profile option

Check All controls whether invalid objects are checked when the Save or Run commands are executed

If you deselect the check box, invalid objects are not checked

If you select the check box, invalid objects are checked

This field sets the CHECK-ALL profile option

Auto Export is a number that specifies the number of Save commands that AionDS executes before you are prompted to export your knowledge base. The default is 0. With this default, you are never prompted to export your knowledge base

This field sets the AUTO-EXPORT profile option

Profile options not available from the Settings pull-down menu

Overview

The following profile options cannot be changed using the Settings pull-down menu. Use a text editor to edit the profile. The following profile options are valid only when you are editing character-based applications.

Backup	Mouse-Support**
Command-Preference	Password-Program
Confirm-Delete*	Rerun
Debug-Split	Rule-Type*
Edit-Interrupt	Save-Deleted-Lines
Flush-Typeahead	SQL separators
Initial-Wrap-Mode	Sorted-Export
Insert-Mode	Tab-Width
Join-At-End	Value-Can-Change*
Keep-Entry-Mode	600-Export
Match-parenthesis	

* This profile option is also valid when you are editing knowledge bases with a graphical user interface.

** This profile option is also valid when you are running character-based applications.

BACKUP

specifies whether AionDS records user-sourced answers during a consultation. The ability to back up and rerun a consultation both rely on recorded answers. You can specify one of the following values:

No	answers are not recorded
Yes	answers are recorded

If you answer yes, AionDS saves answers in memory. For very long consultations, saving information slows the consultation.

COMMAND-PREFERENCE

controls the initial command mode for each monitor

This option can have one of the following values:

COMMAND	initial command mode is to put the cursor on the command line
MENU	initial command mode is to put the cursor in the Options menu

CONFIRM-DELETE

specifies whether to AionDS to confirm to delete an object before AionDS removes it.

ON	specifies that AionDS confirms before it deletes the object. This is the default option
OFF	specifies that AionDS takes action before immediately without confirmation

On the mainframe, it is sometimes useful to avoid confirmation when you want to delete many objects by specifying them in the List Objects monitor.

DEBUG-SPLIT

specifies the division of the screen between the Consultation monitor and the Debug monitor in AionDS on the PC. The default is OFF.

This option can have one of the following values:

OFF	screen is not split between the Consultation monitor and the Debug monitor
ON	screen is split between the Consultation monitor and the Debug monitor. The first 25 lines of the screen are reserved for the Consultation monitor; the remaining lines display the Debug monitor

EDIT-INTERRUPT specifies the action AionDS takes when you enter an editor. The default is ON.

This option can have one of the following values:

OFF	cursor is positioned in the Windows menu or on the command line when you enter an editor
ON	cursor is positioned in an editor window when you enter an editor

FLUSH-TYPEAHEAD controls whether characters typed before AionDS is ready to process them are saved in AionDS on the PC. The default is ON.

This option can have one of the following values:

OFF	characters that are typed ahead are saved and processed when AionDS is ready for them
ON	characters cannot be typed ahead

INITIAL-WRAP-MODE controls whether words are wrapped to the next line in AionDS on the PC. The default is ON.

This option can have one of the following values:

OFF	words cannot be wrapped; characters are not accepted after you fill up the line
ON	words are wrapped to the next line when you fill up the original line

INSERT-MODE controls how characters are inserted in AionDS on the PC. The default is ON.

This option can have one of the following values:

OFF	characters are inserted at the cursor, replacing any existing characters. This mode is also known as typeover mode
ON	characters are inserted to the left of the cursor; existing characters are not replaced

JOIN-AT-END controls where text strings are joined in AionDS on the PC. The default is OFF.

This option can have one of the following values:

OFF	text is joined at the position of the cursor
ON	text is joined at the end of the line

KEEP-ENTRY-MODE controls whether the current insert mode is kept after a function key is pressed in AionDS on the PC. The default is ON.

This option can have one of the following values:

OFF	insert mode is returned to the setting of INSERT-MODE after a function key is pressed
ON	insert mode is not returned to the setting of INSERT-MODE after a function key is pressed

MATCH-PARENTHESIS controls whether AionDS checks for unmatched parentheses when you edit a text property. This option can have one of the following values:

ON	AionDS notifies you of unmatched parentheses. This is the default on the PC
OFF	AionDS does not check. This is the default on the mainframe

MOUSE-SUPPORT specifies whether to activate a mouse.

ON	specifies mouse support is activated
OFF	specifies no mouse support

The default is OFF. If you change this option, restart AionDS for your change to take effect.

PASSWORD-PROGRAM specifies the program AionDS uses to replaces the security component of AionDS when a protected knowledge base is loaded.

RERUN specifies rerunning a consultation after recording user-sourced answers during a consultation. Rerunning a consultation relies on the answers recorded by using the BACKUP option.

RULE-TYPE specifies the text to be put in the Rule Type property when a rule object is first created. There is no default.

This option can have any of the Rule Type attributes

📖 *For more information:* See Chapter 13, “Rule,” in the Aion Development System *General Reference*.

SAVE-DELETED-LINES controls whether deleted lines are kept in the paste buffer in AionDS on the PC. The default is OFF.

This option can have one of the following values:

OFF	deleted lines are not saved in the past buffer
ON	deleted lines are saved in the paste buffer

SEPARATORS

DECIMAL	specifies a single character to display as a decimal point in a real number. This field sets the DECIMAL-SEPARATOR profile option
THOUSANDS	specifies a single character to display to separate thousands (000) in a number. This field sets the THOUSANDS-SEPARATOR profile option
SQL DECIMAL SEPARATOR	specifies a single character to display as a decimal point in a real number. The character is only used when AionDS substitutes a real number for a parameter or slot in an SQL statement

SORTED-EXPORT causes Export to output a sequenced file. Sorted export lets change/source management systems control changes made to the knowledge base at the Export file level.

TAB-WIDTH controls the number of positions to move the cursor when the TAB key is pressed in AionDS on the PC. The default is 4.

VALUE-CAN-CHANGE specifies the default choice for the Value-Can-Change property of parameters and slots. This option can have one of the following values:

- FALSE
- TRUE
- TRUE/ASK

600-EXPORT causes Export to output Version 6.0- and Version 6.1-compatible files. These files can be processed by the AionDS Version 6.0 and 6.1 systems.

Appendix
Building and Running Character-Based Applications in AionDS

Introduction

This appendix describes building and running character-based applications in the GUI AionDS environment.

Note: In AionDS/Win and AionDS/2, you cannot build and run a character-based interface in the GUI environment. You must use a separate character-based system.

In this appendix

Building character-based applications

Overview

You can use AionDS to build character-based applications to run on the PC and mainframe. A **character-based application** is a program with a user interface that displays screens using characters as the basic display unit—such as in a DOS or OS/2 environment. A **GUI application** is a program with a user interface that displays windows using graphics as the basic display unit—such as in a Windows or PM environment.

Although you build objects in the same manner as described in chapters 3 and 5, use the Screen editor to build screens displayed in a character-based application—you cannot use the Window editor (described in Chapter 6, "Working with the Window Editor") to build character-based screens. This section describes how to build and edit screens using the Screen editor.

Caution: To import an existing character-based knowledge base or build a new character-based application, set the User Interface field in the Profile monitor to `character`. See Chapter 3, "Working at the Knowledge Base Level," in the AionDS *Users Guide* (*Character-based)* manual for details about opening or importing a character-based knowledge base. See Chapter 10, "Customizing AionDS," in the AionDS *Users Guide* (*Character-based)* manual for details about the Profile monitor. The Screen editor used in character-based applications is not available in a GUI application. You cannot build the screens displayed in a character-based application.

Porting to the mainframe

You can port character-based knowledge bases developed on the PC to the mainframe.

📖 *For more information:* See Appendix A, "Porting Knowledge Bases," in the AionDS *General Reference* for details about porting an application built on the PC to the mainframe.

Authorize command

In GUI AionDS, there is no Authorize command nor any equivalent to it. When you import a character-based knowledge base, any restrictions placed on the knowledge base by the Authorize command are retained, but cannot be changed.

📖 *For more information:* See AionDS *Users Guide (Character-based)*.

Procedure for building screens

Procedure

To build a screen for an object, perform the following procedure:

1. Open the Screen editor of the object to build a screen.

   📖 *For more information:* See "Opening the Screen editor" on page A-7.

2. Build a Display object for that object.

   📖 *For more information:* See "Creating new Displays" on page A-11.

3. Build the windows for the Display object.

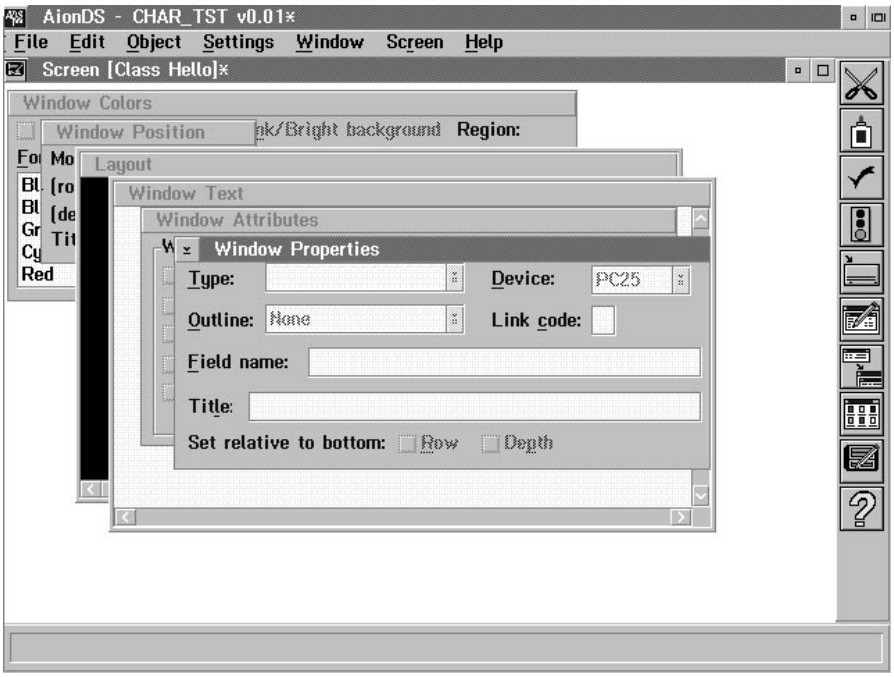
   📖 *For more information:* See "Creating new windows" on page A-18.

The Screen editor

Overview

The Screen editor is used to design the layout of windows in a Display object. **Display objects** represent customized screens. **Windows** are areas in a customized screen that you use to display or enter information, or use as options that you can activate. The Screen editor allows you to see and modify the way that the windows of the Display object appear on the screen.

The following is an example of the Screen editor.



Screen editor and Display objects

When using a Screen editor, you are really making changes to Display objects. The Screen editor provides a consolidated place to manipulate all windows owned by all Displays attached to a particular object. Use the Window editor to create and edit windows that you place in a display.

Suppose you add a new Display object to a Parameter that doesn't have any Display objects attached. AionDS creates a new Display object in the knowledge base. You can edit this Parameter's Display using the Screen editor or the Display object editor—just like any other object.

Screen editor for Groups and Instances

Although a screen is defined by a single object, for Groups and Instances the Screen editor represents several components that make up the object. The displays are not attached directly to Groups and Instances. The Screen editor for a Group object represents the Displays attached to the members of the Group. The Screen editor for an Instance represents the Displays attached to its slots.

Opening the Screen editor

Overview

In order to perform any Screen editor actions described in this section, have a Screen editor window open and active. The description of Screen editor actions assumes that you have already opened a Screen editor for an object.

You can design a screen for the following objects:

Parameter	Instance
Slot	Message
Group	Class
Display	Type

What to do

To open a Screen editor for an object, perform the following steps:

1. Select the object to design a screen.
2. Select Object.Screen to open the Screen editor for that object. If you select an object that cannot have a Display attached, the Screen command is grayed out.

   ☯ *Alternative:* Select the Edit Screen button from the Button bar.

Result: The object's Screen editor displays and the Screen pull-down menu displays in the menu bar.

Summary of Screen editor commands

Overview This is a summary of the commands available in the Screen pull-down menu.

Display sub-menu This is a summary of the commands in Screen.Display:

New	creates and attaches a new Display to the current object in the Screen editor
Attach	attaches existing Displays to an object
Detach	detaches a Display from an object
Build	builds a default Display for the current object in the Screen editor. The default Display depends on the type of object
List	toggles the Displays window on and off. The Displays window shows a list of all of the displays attached to the current object in the Screen editor

Window sub-menu This is a summary of the commands in Screen.Window:

New	creates a new window in the Display currently being edited in the Screen editor. The New Window dialog displays This command is available only if a Display has already been associated with an object (via New, Attach, Build, and so on)
List	toggles the Windows dialog on and off. The Windows dialog shows a list of all windows in all Displays attached to the current object in the Screen editor
Attributes	lets you control the attributes of the window currently selected. The Window Attributes dialog displays

Colors	lets you control the colors of the window currently selected. The Window Colors dialog displays
Position	toggles the Window Position dialog on and off. The Window Position dialog shows you the position of the mouse cursor in the Layout window as well as the coordinates of the currently selected Display window
Properties	lets you control the properties of the window currently selected. The Window Properties dialog displays
Text	lets you control the fixed text of the window currently selected. The Window Text dialog displays. This option only applies to Text windows

Other Screen commands

This is a summary of the rest of the commands that are available in the Screen menu:

Layout	toggles the Layout window on and off. The Layout window is the place where you draw the screen and look at its layout
Refresh	refreshes the Layout window, reflecting its current attributes
Hide	hides a window or Display that you select from the Windows dialog or Displays window
Show	shows a window or Display that you select from the Windows dialog or Displays window
Settings	displays the Screen Settings window where you control the display characteristics of the Screen editor

Tile	arrange all open Screen editor windows so that they appear side by side and top to bottom within the Screen editor
Cascade	arrange all open Screen editor windows so that they appear layered one on top of another with only the title bar showing
Close All	close all open Screen editor windows. This choice does not save or discard the contents of the Screen editor windows

Saving the appearance of windows

To save the appearance of the various windows in the Screen editor (including the Displays window), select Settings.Set window template.

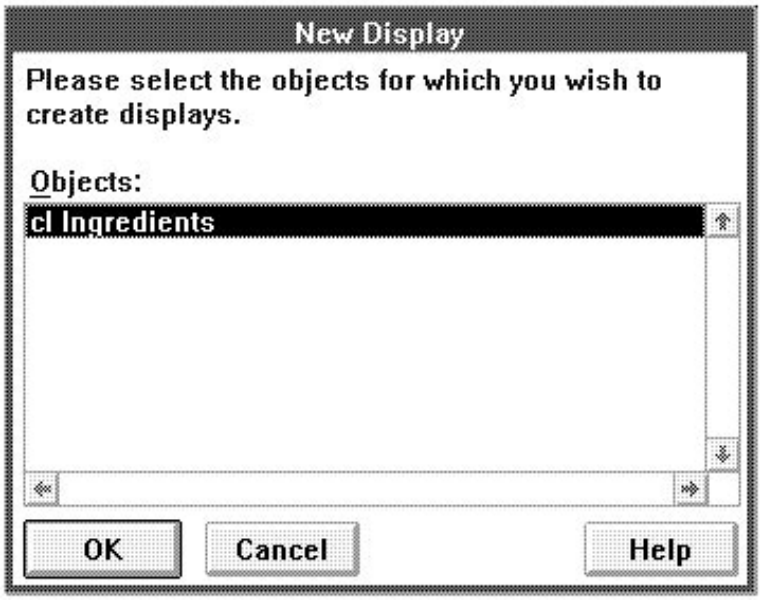
📖 *For more information:* See Chapter 10, "Customizing AionDS."

Creating new Displays

What to do

To create a new Display to attach to the object in the Screen editor, perform the following steps:

1 Select Screen.Display.New. The New Display window displays.



2 In the Objects list box, select the object for which to create a display. When editing a screen for Group or Instance, you can select multiple objects for which to create a display.

3 Click OK.

Result: AionDS creates Displays for the selected objects. The name of the Display object is the same as the name of the first object in the Objects list box.

Alternative

1. To create a Display with a different name than the attached object, create a Display object with Object.New.
2. Select Screen.Display.Attach to attach the Display to another object.

   📖 *For more information:* See “Attaching Displays to objects” on page A-13.

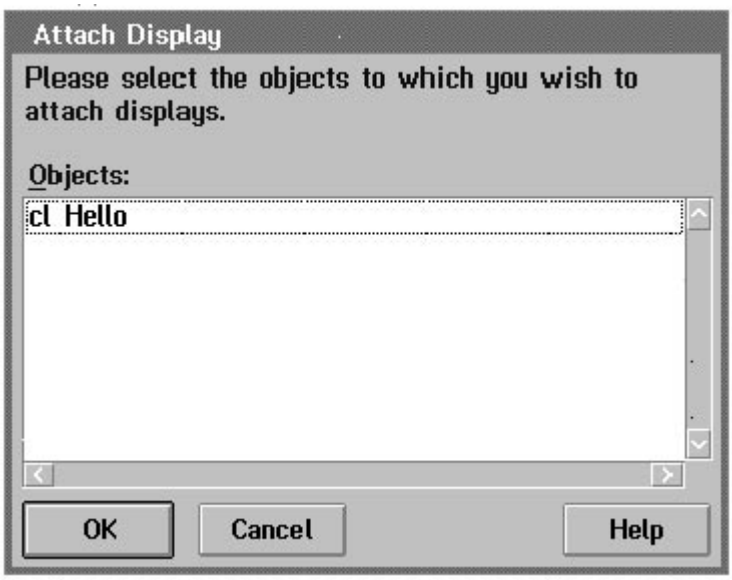
Result: The Display is created and attached to the specified object.

Attaching Displays to objects

What to do

To attach one or more existing Displays to the object in the Screen editor, perform the following steps:

1 Select Screen.Display.Attach. The Attach Display window displays.



2 In the Objects list box, select the objects to which to attach a Display.

3 Click OK. The Attach window dialog displays.

4 In the Attach window, there is a list of valid displays you can attach to the selected object. Select one or more displays to attach to the object.

5 Click OK.

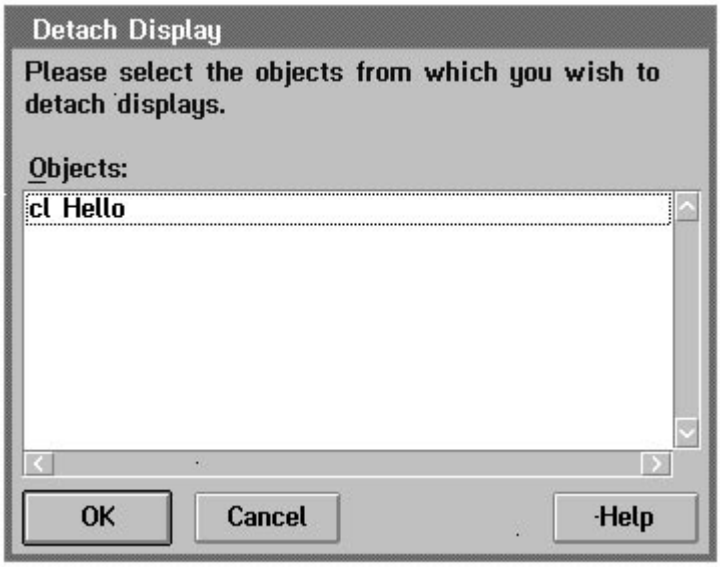
Result: The selected displays are attached to the selected object.

Detaching Displays from objects

What to do

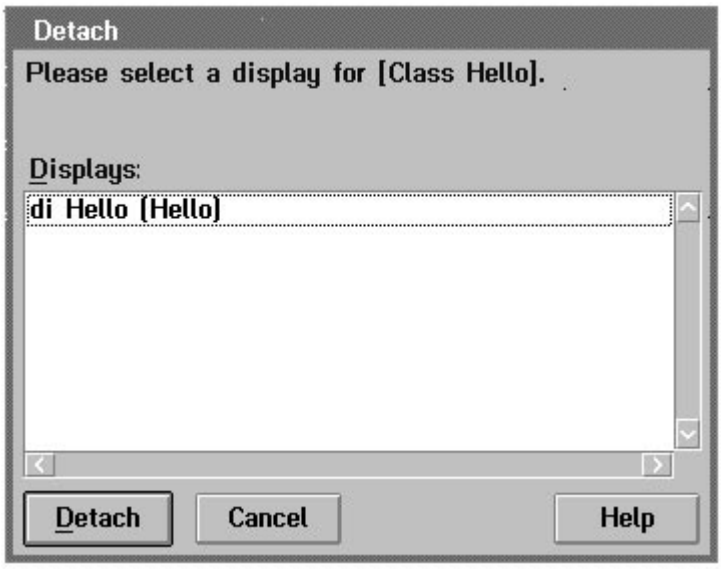
To detach one or more Displays from the current object, perform the following steps:

1 Select Screen.Display.Detach. The Detach Display window displays.



2 In the Objects list box, select the object from which to detach a Display.

3 Click OK. The Detach dialog displays.



4 In the Detach dialog, there is a list of valid Displays you can detach from the selected object. Select one or more displays.

5 Click OK.

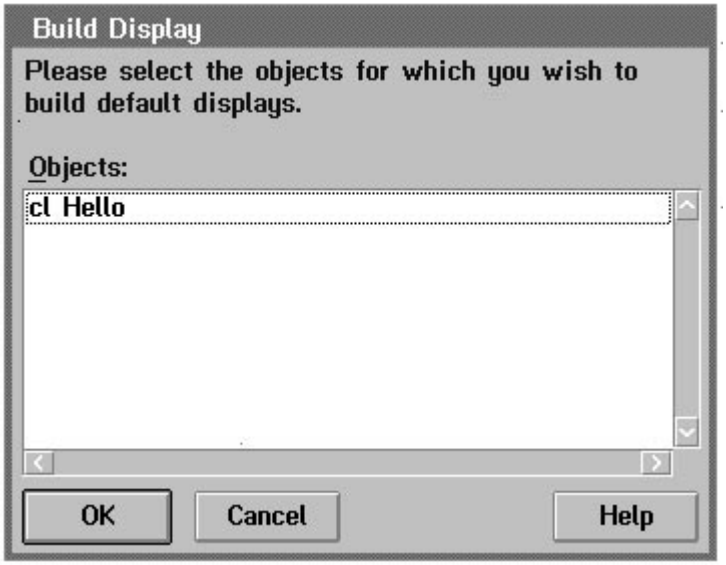
Result: The selected Displays are detached from the selected object.

Building default Displays

What to do

To build a default display for the current object, perform the following steps:

1 Select Screen.Display.Build. The Build Display window displays.



2 In the Objects list box, select the object for which to build a display.

3 Click OK.

Result: AionDS builds the default display based on the object type in the Screen editor. AionDS has default displays for the following objects:

- messages
- parameters
- slots
- groups
- instances
- classes
- types

Caution: If the display already contains windows, they are deleted before Build adds default windows.

Listing the Displays in the current screen

What to do

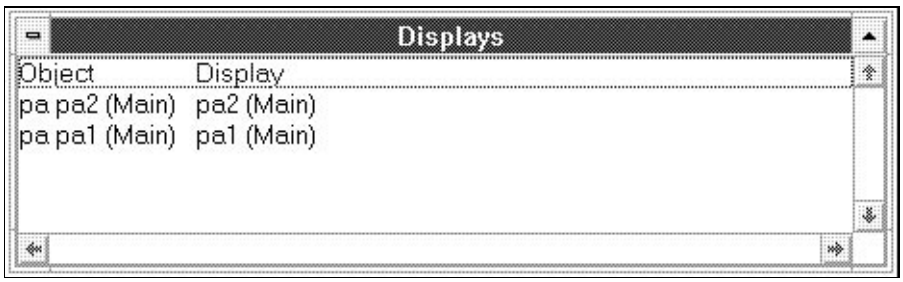
To list the Displays in the current screen,

- Select Screen.Display.List.

Result: The Displays window displays.

What you see

The Displays window shows the names of the Displays used in the current Screen editor. It also shows the names of the objects that are part of the screen being edited.



The example Displays window is for a Group called GROUP1. GROUP1 has two parameters and their attached Displays as members: PA1 and PA2. The Display object PA1 is attached to parameter PA1 and the Display object PA2 is attached to PA2.

Creating new windows

Overview

To create a new window, specify the type of window, its title, and Displays of which it should be a part.

Window types

Windows types are the following:

Window	Function
ANSWER	retrieves an answer to the prompt
BROWSE	displays a system file (such as a trace file) during runtime
BROWSEFILE	displays the file specified in the File Name property of a message
CERTAINTY	displays certainty information, if any
CHOICES	displays choices, if any
COMMAND-LINE	used to enter commands
CONFIRM-MENU	holds the choices for the “Abort?” question
CONSTRAINT	displays constraints on property values
DESCRIPTION	holds a description for the Explanation facility
EXPLAIN	displays the explain text
FUNCTION-KEYS	holds function key assignments

Continued

Continued

Window	Function
GRAPHIC	not currently used
INFORMATION	used by the Explanation facility
INPUT	holds a file name for STORE or PRINTWINDOW commands. It is also used to hold certainty values for certainty sets of records
KDL	holds KDL text for the Explanation facility
MESSAGE-LINE	holds AionDS consultation messages
MESSAGE	holds the message text
OBJECTS-MENU	holds object names, such as the objects you can use with the backup command
OPTIONS-MENU	holds options
PROMPT	displays the prompt
SELECTION	used by the Explanation facility
SELECTIONINPUT	accepts selection input for constrained strings and constrained string sets. It is especially useful on the mainframe because it separates the input from the menu of possible answers this window is used with the SELECT-INPUT profile option. This profile option specifies the characters that cause choices to be selected

Continued

Continued

Window	Function
SUMMARY	used by the Explanation facility
TEXT	holds text for display. Text windows can contain literal text or TSL text. You can also use text windows to create three-dimensional effects. Use the CREATE command or press F4 or PF4 to create windows
TEXTEDITINGKEYS	displays the current function key assignments for alternate key tables. It is only available on the mainframe where there are three tables of function key assignments

What to do

To create a new window in the current Display, perform the following steps:

1 Select Screen.Window.New. The New Window dialog displays.

2 In the Type field, indicate the window type. Click on the arrow to display a list of valid window types. Valid window types are those appropriate to the current display.

   For example, if the display is attached to a slot, you can create prompt windows, answer windows, and so on. However, if the display is attached to a message, you can create message windows but you cannot create prompt or answer windows.

3 In the Title field enter a title in the default title location of the window.

4 The Displays list box indicates which Displays the new window will be a part of. The Displays available from this list include any that are part of the screen being edited.

5 Click OK.

Result: AionDS creates a window as part of the display you indicated. If the Layout window is open, it shows this new window as part of the screen. The Windows dialog lists the new window.

📖 *For more information:* See "Listing and selecting windows in the current screen" on page A-22.

For more information

To modify a window using the Screen pull-down menu commands, see following sections:

"Modifying window attributes" on page A-24

"Modifying the selected window properties" on page A-27

"Modifying the selected window's text properties" on page A-34

"Modifying window colors" on page A-34

To move, resize, copy, cut, and paste windows in the Layout window, see following sections:

"Moving and resizing windows in the Layout window" on page A-39

"Cutting, copying, pasting, and deleting windows" on page A-38

Listing and selecting windows in the current screen

List windows

To list windows,

- Select Screen.Window.List.

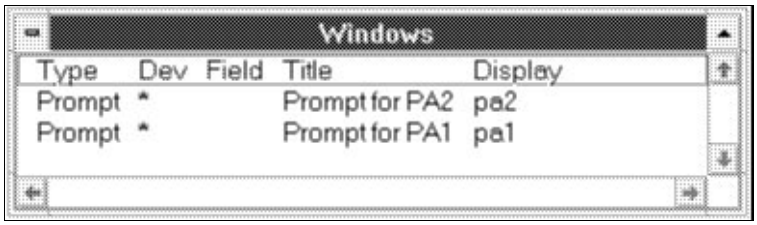
Result: The Windows dialog displays.

What you see

The Windows dialog shows the names of the windows belonging to the screen. If there is more than one Display attached to the object (for example, if you are looking at the Screen editor for a Group) each display and its windows are listed in this dialog.

☑ *Tip:* The title bar of the Screen editor shows an exclamation mark (!) if any windows are invalid.

The following example of the Windows dialog is for a Group called GROUP1. GROUP1 has member parameters PA1 and PA2 and their attached displays. The Display object PA1 is attached to parameter PA1 and owns the Prompt for PA1 window. The Display object PA2 is attached to PA2 and owns the Prompt for PA2 window.



The Windows dialog lists the windows attached to the current display. You cannot edit the contents of this property directly. Instead, use the Screen editor or the Window Properties editor to change the contents of the individual windows.

The following fields are shown:

Type	identifies the type of window
Dev	displays the Device-Type property value of the window
Field	identifies the record or field name
Title	identifies the title of the window
Display	identifies the window's attached display. This is useful when the Screen editor shows groups of displays

Selecting a window

To select a window from the Windows dialog,

- Click on the window.

☯ *Alternative:* In the Layout window, click on a window to select it.

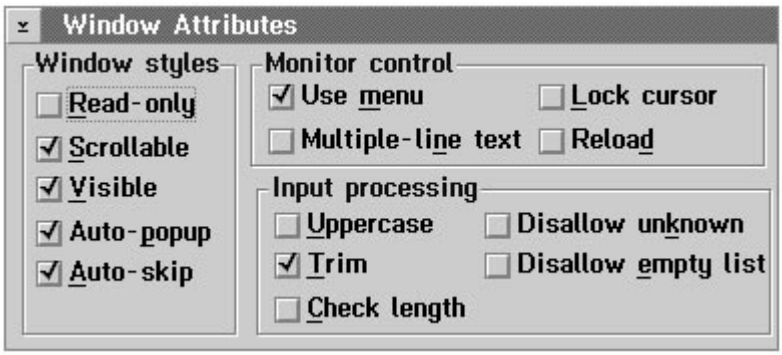
Modifying window attributes

Overview

The Attributes window controls the behavior of the currently selected window.

What to do

1. Select a window using the Windows dialog or the Layout window.
2. Display the Window Attributes dialog by selecting Screen.Window. Attributes. The Windows Attributes dialog displays.



3. Select the desired options in this dialog.
4. Select Screen.Refresh.

   ☯ *Alternative:* Select File.Save.

Result: The Layout window displays the window with its new attributes.

Window Attributes options

The Window Attributes dialog contains the following groups of check boxes:

Window Styles, Monitor Control, and Input processing.

Window Styles

Read-only	text in the window cannot be modified by the user
Scrollable	text in the window scrolls
Visible	user input to the window is printed on the screen. Most windows have this checked. However, a window in which the user enters a password might have this attribute unchecked—to hide the password
Auto-popup	window pops up when the cursor enters the window
Auto-skip	only valid on the mainframe. The cursor automatically skips to the next window when the last character is entered

Monitor control

Use menu	answer choices are presented as a menu
Lock cursor	cursor stays in the window until an answer is entered
Multiple-line text	multiple lines of text can be entered
Reload	contents of the window are updated whenever the screen is refreshed. Assignment to a field in a record does not cause the window to be reloaded unless the assignment causes the record to be processed. The window is not reloaded if the window is modified

Input processing

Uppercase	input is converted to uppercase
Trim	AionDS trims leading or trailing blanks from input
Check length	length of user input is checked
Disallow unknown	user cannot press F5 or PF5 to enter UNKNOWN
Disallow empty list	empty list or set cannot be entered

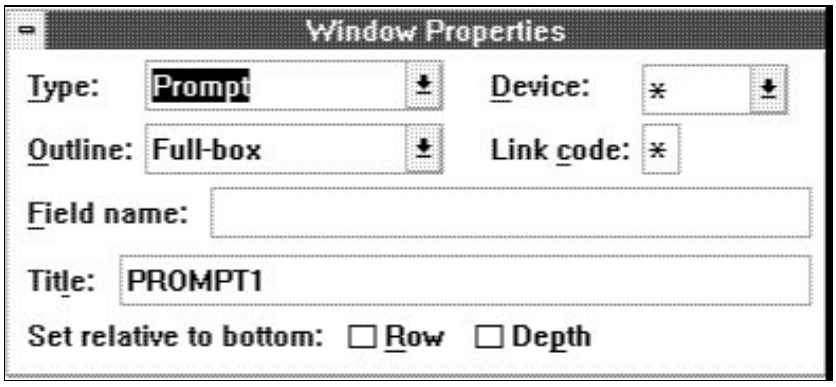
Modifying the selected window properties

Overview

The Window Properties dialog lets you control the properties of the window currently selected.

What to do

1. Select a Display window using the Windows dialog or the Layout window.
2. Display the Window Properties dialog by selecting Screen.Window.Properties. The Windows Properties dialog displays.



3. Change the fields in this dialog. See following sections for information about each field.
4. Select Screen.Refresh.

   ☯ *Alternative:* Select File.Save.

   Result: The Layout window displays the window with its new attributes.

Type

Purpose

Identifies the purpose of the window, such as to display a prompt, answer, or constraint. The window type determines several characteristics of a window, such as when it displays and whether the user can enter information in the window

Valid values

You can specify one of the following values. These types are explained in the "Creating new windows" on page A-18.

ANSWER	EXPLAIN	OPTIONS-MENU
BROWSE	FUNCTIONKEYS	PROMPT
BROWSEFILE	GRAPHIC	SELECTION
CERTAINTY	INFORMATION	SELECTIONINPUT
CHOICES	INPUT	SUMMARY
COMMAND-LINE	KDL	TEXT
CONFIRM-MENU	MESSAGE	TEXTEDITINGKEYS
CONSTRAINT	MESSAGE-LINE	
DESCRIPTION	OBJECTS-MENU	

Outline

Purpose

The Outline field specifies the kind of border you want around the window. For example, you can specify FULL-BOX to place a full box around the window or NONE to hide the outline.

Valid values

This property can have one of the following values:

Value	Description
BOTTOM-LEFT	lines display on the bottom and the left side of the field
BOTTOM-LINE	line displays on the bottom of the field

Continued

Continued

Value	Description
BOTTOM-RIGHT	lines display on the bottom and the right side of the field
FULL-BOX	lines display on the bottom
TOP-LINE	an outline displays only at the top of the window
HORIZONTAL-LINES	lines display on the top and bottom of the field
LEFT-LINE	a line displays on the left side of the field
NONE	no lines display around the field
OPEN-BOTTOM	lines display on the top and the left and right sides of the field
OPEN-LEFT	lines display on the top bottom and right side of the field
OPEN-RIGHT	lines display on the top bottom and left side of the field
OPEN-TOP	lines display on the bottom and left and right sides of the field
RIGHT-LINE	a line displays on the right side of the field
TOP-LEFT	lines display on the top and left side of the field
TOP-LINE	a line displays on the top of the field
TOP-RIGHT	lines display on the top and the right side of the field
UNDERLINE	no lines display around the field, but the field within the window is underlined
VERTICAL-LINES	lines display on the left and right sides of the field

Device

Purpose

The Device field specifies the type of monitor on which this window displays. The default is an asterisk (*).

Valid values

The table below lists the Device specifications, the terminals to which they apply, and the number of rows and columns in each.

Device	Terminal	Rows	Columns	Descriptions
*	All	Any	Any	the window displays on every monitor or terminal
PC*	PC	Any	Any	the window displays on any PC monitor
PC25	PC	25	80	the window displays only on monitors supporting 25 lines and 80 columns
PC43	PC	43	80	the window displays only on monitors supporting 43 lines and 80 columns
PC50	PC	50	80	the window displays only on monitors supporting 50 lines and 80 columns
M*	3270	Any	Any	the window displays on any mainframe terminal
M2	3270	24	80	the window displays only on terminals supporting 24 lines and 80 columns

Continued

Continued

Device	Terminal	Rows	Columns	Descriptions
M4	3270	43	80	the window displays only on terminals supporting 43 lines and 80 columns
M5	3270	27	132	the window displays only on terminals supporting 27 lines and 132 columns

Link-code

Purpose

Specifies a group of windows that scroll together

Valid values

This option can contain any alphanumeric character. An asterisk (*) in the Link-code property indicates no link. No link is the default.

To establish a link between windows, use the same link code for each window. For example, if the answer windows for the NAME, ADDRESS, and DATE_OF_BIRTH slots all use link code A, the three windows are linked together.

Field Name

Purpose

serves to identify a window and to link a window to a record field

Valid values

Specifies the name of a field in a record that corresponds to the window for the Answer, Certainty, Choices, and Constraint windows of record parameters. If the record field name is not specified for a window, AionDS does not display a window when it displays the record during execution.

For other windows, the Field Name property contains a name to identify the window. You can use the Field Name property to identify windows for use with the CM statements in KDL.

📖 *For more information:* See Chapter 4 of the AionDS *Language Reference*.

Title

Purpose Contains the title of the window. Titles can include TSL

Titles can only be displayed on a single line. The Title property lets you enter more than one line of text because TSL statements might require more characters than the actual display.

The text in the Title property overwrites the window outline if it is placed on the same line the outline is on.

Set relative to bottom

Purpose Specifies whether or not to set a window relative to the bottom of the screen. You set the position of the window in the Layout window.

Row check box	sets the last row for the window or title relative to the bottom of the screen. The text portion of the window begins the distance (as shown in the Layout window) from the bottom of the screen and extends upward. The position of the window relative to the top of the screen might vary for different types of devices
Depth check box	sets the last row for the window relative to the bottom of the screen. The window extends to the distance (as shown in the Layout window) from the bottom of the screen

If the window displays on a different screen, it displays the same number of lines from the bottom of that screen. You should not specify relative positions for both the row and depth of the same window.

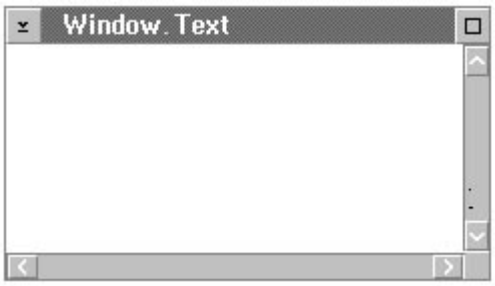
Modifying the selected window's text properties

Overview

The Window Text dialog lets you edit the fixed text of the window currently selected. The Window Type must be Text.

What to do

1. Select a text window using the Windows dialog or the Layout window.
2. Display the Window Text dialog by selecting Screen.Window.Text. The Window Text dialog displays.



3. Enter the text to appear in the window.
4. Select Screen.Refresh (or press F6).

   ☯ *Alternative:* Select File.Save.

Result: The Layout window displays the window with its new text.

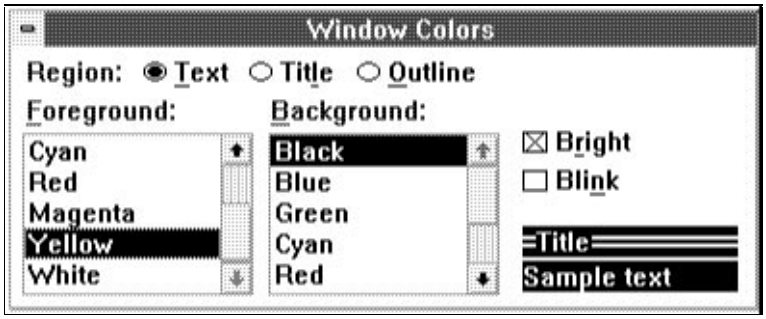
Modifying window colors

Overview

The Window Colors dialog controls the colors of the currently selected window.

What to do

1. Select a window using the Windows dialog or the Layout window.
2. Display the Window Colors dialog by selecting Screen.Window.Colors. The Window Colors dialog displays.



3. Indicate the area of the window in which to set colors by selecting a Region radio button. The other controls in the Window Colors dialog reflect the color characteristics of the selected region. For example, if the Title radio button is selected, the Foreground, Background, Bright, Blink, controls and the sample area reflect the colors of the selected window's title.

4 The Foreground and Background list boxes contain the colors available for the foreground and background of the window region selected by the Region radio buttons. Select one color from each list box for each region of the window.

   The sample area shows how text for the selected window will look based on the current selections in the Window Colors window. You do not directly edit this part of the Window Colors dialog—it reflects the current settings.

5 Use the Bright and Blink check boxes to indicate whether the color of a particular region is bright or blinking.

   On the PM desktop, Blink does not work the same as in a full-screen execution of AionDS on the PC or mainframe. On the PM desktop, Blink sets the background color to bright. For example, if the background color is yellow and blink is selected, the window has a bright yellow background on the PM desktop, but is normal yellow and blinking when you run the consultation in full-screen mode.

6 Select Screen.Refresh.

   ☯ *Alternative:* Select File.Save.

Result: The Layout window displays the window with its new colors.

Ordering windows

Overview

To indicate the order in which the windows display during a consultation, use the Edit.Cut and Edit.Paste options to paste the windows in the desired order. Windows display in the order that they are created.

☑ *Tip:* Windows that display later are placed on top of windows that were displayed earlier.

What to do

To reorder the windows, follow this procedure:

1. Select the window to move.
2. Cut the window to the clipboard by selecting Edit.Cut.Window.

3 Paste the window back into the screen by selecting Edit.Paste.Window. The Paste Window displays. Use the Paste Window to specify the order in which windows are drawn on the screen.

4 Select the Display in which the window is to be pasted.

5 Select the window that displays immediately after the one to be pasted. Do not select a window if you want to paste the window at the end of the window list.

6 Click OK.

Result: The window is pasted into the display in the specified order.

Cutting, copying, pasting, and deleting windows

Overview

The concepts of cut, copy, paste, and delete discussed in Chapter 5, "Working with Objects," also apply to the Display windows in the Layout window of the Screen editor. When you select any Display window or windows, the Edit pull-down menu becomes active for the Display window.

Moving and resizing windows in the Layout window

Overview

There are a number of ways to move and resize the windows in the Layout window. You can also move and resize window titles. To move windows in the Layout window, first select a window.

Use the mouse to select and move a window or windows. You can resize the windows by moving a window border with the mouse.

📖 *For more information:* See Chapter 6, "Working with the Window Editor."

Selecting a window title

To select a window title, perform the following steps:

1. Deselect the window.
2. Place the tip of the mouse over the title and click.

Deselecting a window title

To deselect a window title,

- Place the mouse over any of the following areas and click:
  - An area of the window that is not a part of the title
  - An area of the Layout window that is not part of a window

Moving the title of a window

To move the title of a window, perform the following steps:

1. Deselect the window.
2. Select the title.
3. Drag the title to the desired location.
4. Release the mouse button.

Resizing a title

You can only resize a title by changing the name. To resize a title, perform the following steps:

1. Select from the Screen.Window.Properties. The Window Properties dialog displays.
2. Change the text of the title in the Window Properties window.

Displaying the Window Position dialog

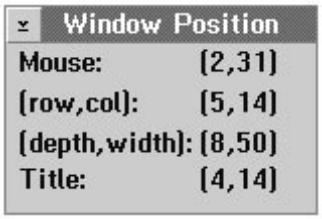
Overview

The Window Position dialog shows you the position of the mouse cursor in the Layout window and the size and position of the currently selected window.

What to do

- Display the Window Position dialog by selecting Screen.Window.Position.

  Result: The Window Position dialog displays.



Displaying the Layout window

What to do

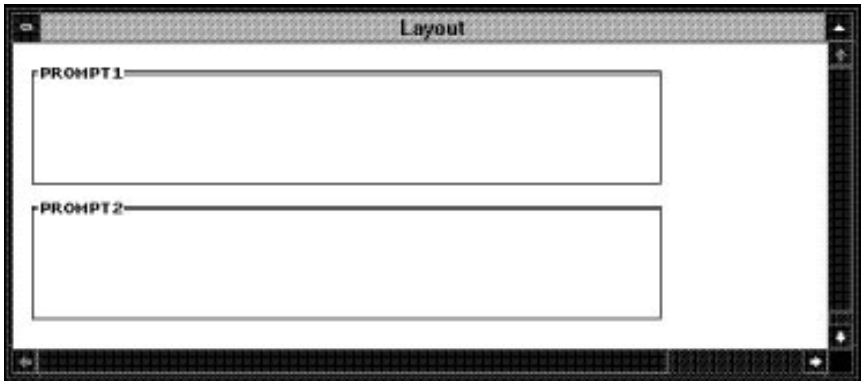
- Selecting Screen.Layout.

  Result: The Layout window displays.

What you see

The Layout window shows the appearance of the windows of all of the Displays that are attached to the object the Screen editor is associated with.

The example Layout window shown in the following example is from the Screen editor for a Group called GROUP1. GROUP1 has members PARAM1 and PARAM2 and their attached displays. The Display object PARAM1 is attached to parameter PARAM1 and owns the PROMPT1 window. The Display object PARAM2 is attached to PARAM2 and owns the PROMPT2 window.



Updating the Layout window

What to do

To show all current changes in the Layout window,

- Select Screen.Refresh (or press F6).

Result: All current changes display in the Layout window.

Hiding and showing selected windows or Displays

Overview

Sometimes, you want to hide the windows or Displays shown in the Layout window. You might want to do this so you can see a particular window you are designing. Or you might want to see what the screen would look like without the windows.

Hiding windows or displays

1. To hide windows or Displays, select the windows or Displays from one of the following places:
   - Displays window
   - Windows dialog
2. Select Screen.Hide.

Result: The selected windows and displays are hidden. All of the windows that the selected Display owns are also hidden. Hide does not delete windows or displays.

Showing a window or display

1. To show a window or display that is hidden, select it from one of the following:
   - Displays window
   - Windows dialog
2. Select Screen.Show.

Result: The selected window or display is shown.

☑ *Tip:* By default, the Explain window and window are not displayed initially. Use Show to display these windows.

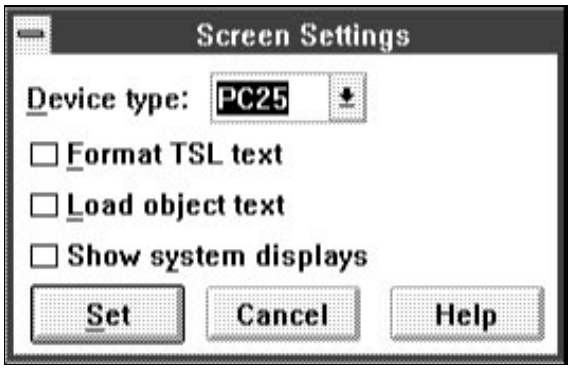
Modifying Layout window characteristics

Overview

The Screen Settings dialog controls the display characteristics of the Screen editor.

What to do

1 To open the Window Settings dialog, select Screen.Window.Settings. The Screen Settings dialog displays.



2 Use the Device Type pull-down list box to indicate the type of display to use. Valid entries are available by selecting one from the drop-down list or by entering a valid entry into the field.

3 Select the Format TSL text check box if the text in the Layout window should be formatted by the TSL commands.

4 Select the Load object text check box if the text from the object should be loaded into the Layout window.

5 Select the Show system displays check box if you want the Screen editor window to include system Displays in the Displays, Window, and Layout windows.

☑ *Tip:* Selecting all three check boxes previews what the screen looks like at runtime.

6 Click Set.

Result: The new settings are applied to the Screen editor.

Reorganizing the windows in the Screen editor

Overview

You can reorganize the open windows in the Screen editor. Select one of the following options from the Screen pull-down menu:

Tile	arrange all open Screen editor windows to appear side by side and top to bottom within the Screen editor
Cascade	arrange all open Screen editor windows in the active Screen editor to be layered one on top of another, with only the title bar showing
Close All	close all open Screen editor windows does not save or discard the contents of the Screen editor windows, it only hides them

Running character-based applications

Overview

You can use AionDS to run character-based applications. The character-based application is executed using the Consultation monitor. AionDS starts up the Consultation monitor automatically when you run a consultation.

Caution:

- In AionDS/PM, to run a character-based application you need not set the Interface field in the Environment Settings window to `GRAPHICAL`. However, if there are GUI calls in the character-based application, errors are reported.
- In AionDS/Win, to run a character-based application import the application with the User Interface field in the Profile monitor set to `CHARACTER`. Importing the application with the User Interface field set to `CHARACTER` adds the WindowObject and DDEObject hierarchies to the application. With these hierarchies, the character-based application can be executed in AionDS/Win.

The **Consultation monitor** is the runtime component of AionDS that displays information to and solicits information from the person running a consultation. The Consultation monitor can solicit information to resolve parameters and slots. It can also display parameters, slots, messages, graphs, classes, instances, or any object that needs to resolve one of these objects (such as a group object or a display object).

The appearance of the Consultation monitor varies, and is determined by the display object associated with the information being presented or solicited.

Customization

You can customize the Consultation monitor in the following ways:

- use the Screen editor to create displays and associate them with one or more slots, parameters, classes, or messages
- use the KDL command CM to add, remove, and modify windows at runtime and execute system commands based on information from the screen

📖 *For more information:* See Chapter 4 of the AionDS *Language Reference*.

What to do

To run a character-based knowledge base, perform the following steps:

1. Select File.Open. The Open Knowledge Base window displays.
2. Select the knowledge base and any appropriate options.

   📖 *For more information:* See Chapter 3, “Working at the Knowledge Base Level.”
3. Click Open. The knowledge base opens.
4. Select File.Run. The Run window displays.
5. Select the appropriate options.

   📖 *For more information:* See Chapter 3, “Working at the Knowledge Base Level.”
6. Click Run.

Result: The Consultation monitor displays and the knowledge base is executed.

If you enabled the debugger and selected the Running trace option, the Debugger is activated and the Run pull-down menu, Call Stack ,and Breakpoints windows display.

📖 *For more information:* See Chapter 9, “Working with the Debugger.”

Backup/Rerun

In the Run window, the following options in the Backup/Rerun group are available only when you run a character-based knowledge base:

Store Answers	specifies whether user-sourced answers are recorded during knowledge base execution. The ability to back up and the ability to rerun a knowledge base both rely upon answers being recorded If you select the check box, answers are recorded. AionDS saves answers in memory. For very long knowledge base executions, saving this information slows the speed of the execution If you deselect the check box, answers are not recorded This field sets the BACKUP-ENABLE profile option
Preload Answers	specifies whether to load and display answers to questions during the back up or rerun of a knowledge base If you deselect this check box, prior answers are not displayed. The knowledge base execution only stops to display messages and graphs If you select this check box, prior answers are loaded in the window and the object displays. The user can change the answer before continuing This field sets the PRELOAD-ANSWERS profile option

Index

6

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W