

**Rational Environment  
Reference Manual**

**Reference Summary**

Copyright © 1985, 1986, 1987 by Rational

Document Control Number: 8001A-21 (803-002302)

Rev. 2.0, September 1985

Rev. 3.0, November 1985

Rev. 4.0, July 1986

Rev. 5.0, July 1987(Delta)

This document subject to change without notice.

Note the Reader's Comments form on the last page of this book, which requests the user's evaluation to assist Rational in preparing future documentation.

Ada is a registered trademark of the U.S. Government (Ada Joint Program Office).

Rational and R1000 are registered trademarks and Rational Environment and Rational Subsystems are trademarks of Rational.

Rational  
1501 Salado Drive  
Mountain View, California 94043

## How to Use This Book

The Reference Summary book of the *Rational Environment Reference Manual* is intended to be used as a quick reference to the resources provided by the Rational Environment™. The information is intended for experienced users of the Rational Environment. The *Rational Environment Basic Operations* and the *Rational Environment User's Guide* provide a better introduction and quick reference for new users.

Products other than the Rational Environment (for example, Rational Networking—TCP/IP or Rational Target Build Utility) are not documented in the Reference Summary. If you have purchased such products, however, the documentation for them typically contains quick-reference pages and tabs that are intended to be inserted into this Reference Summary.

The Reference Summary provides the following information, organized into tabbed sections:

- **World !:** A map of the library system of the Rational Environment. The map includes an indication of the manual/book in which each unit is documented.
- **World !Commands:** Specifications for the units that provide the interactive command interfaces to the facilities provided by the Environment. The specifications are organized alphabetically by simple name.
- **World !Io:** Specifications for the I/O packages defined in Chapter 14 of the *Reference Manual for the Ada® Programming Language*. It also contains the specifications for other I/O packages provided by the Environment. The specifications are organized alphabetically by simple name.
- **World !Lrm:** Specifications for the predefined units required by the *Reference Manual for the Ada Programming Language* that are provided by the Environment. The specifications are organized alphabetically by simple name.
- **World !Tools:** Specifications for the software components provided by the Environment and the programmatic interfaces to the resources provided by the Environment. The specifications are organized alphabetically by simple name.
- **Abbreviations:** Definitions of the predefined abbreviations provided for frequently used Environment commands and unit names.
- **Model Definitions:** Definitions of the predefined models that can be used to initialize the imports (links) of newly created worlds and subsystem views.

- **Project Tools:** An empty section intended as a repository for quick-reference information describing locally developed tools.
- **Release Information:** An empty section intended as a repository for release notes distributed by Rational.
- **Symbols and Switches:** Information on library-naming syntax and special symbols (including wildcards and substitution characters), search pattern characters, and session and library switches.
- **System Programming:** An empty section intended as a repository for specifications of units used locally for system programming that are not included in any of the above sections.

## Organization of the Reference Manual

The *Rational Environment Reference Manual* (Reference Manual for brevity) includes the following volumes (see accompanying illustration):

- |    |   |
|----|---|
| 1  | Reference Summary<br>Keymap<br>Master Index         |
| 2  | Editing Images (EI)<br>Editing Specific Types (EST) |
| 3  | Debugging (DEB)                                     |
| 4  | Session and Job Management (SJM)                    |
| 5  | Library Management (LM)                             |
| 6  | Text Input/Output (TIO)                             |
| 7  | Data and Device Input/Output (DIO)                  |
| 8  | String Tools (ST)                                   |
| 9  | Programming Tools (PT)                              |
| 10 | System Management Utilities (SMU)                   |
| 11 | Project Management (PM)                             |

Each *volume* of the Reference Manual contains one or more *books* separated by large colored tabs. Each book contains information on particular features or areas of application in the Environment. The abbreviation for the name of each book (for example, EI for Editing Images) appears on the binder cover and spine, and this abbreviation is used in page numbers and cross-references. The books grouped into one volume are not necessarily logically related.

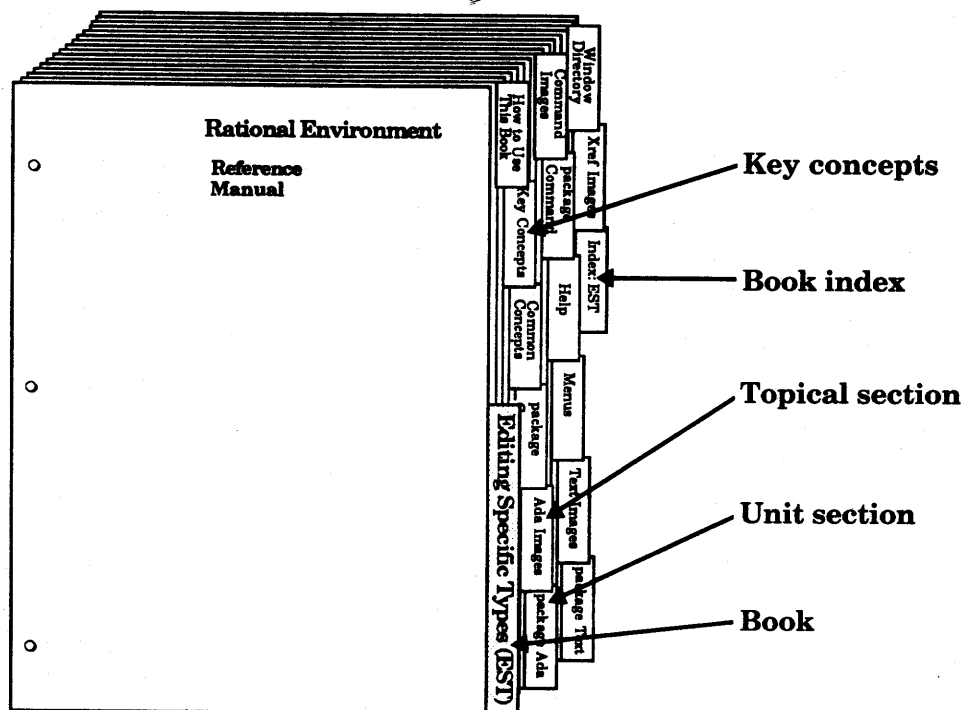
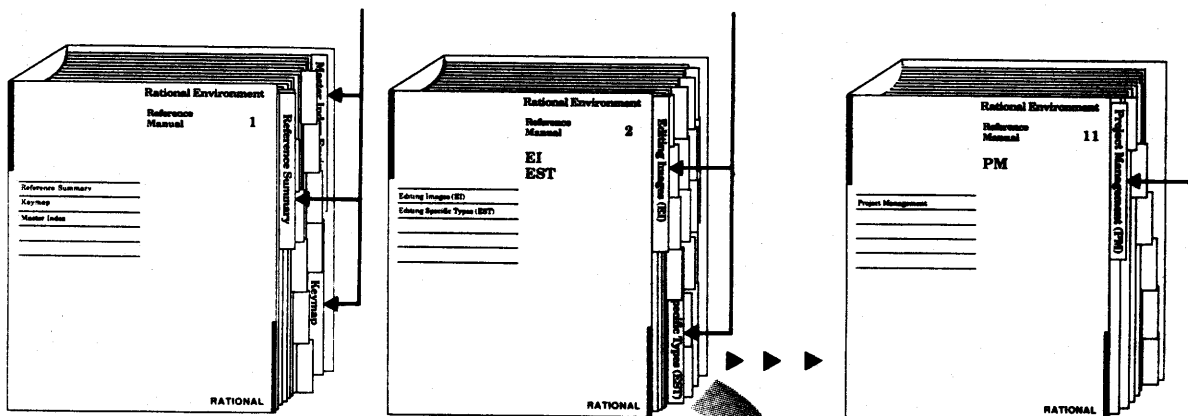
# Organization of the *Rational Environment Reference Manual*

11 volumes containing 14 books

Volume 1: 3 books

Volume 2: 2 books

Volume 11: 1 book



A sample book

The Reference Manual provides reference information organized to efficiently answer specific questions about the Rational Environment. The *Rational Environment User's Guide* complements this manual, providing a user-oriented introduction to the facilities of the Environment. Products other than the Rational Environment (for example, Rational Networking—TCP/IP or Rational Target Build Utility) are documented in individual manuals, which are not part of the Reference Manual.

### Volume 1

Volume 1, intended to be used as a quick reference to the resources provided by the Environment, contains the following books:

- **Reference Summary:** The Reference Summary contains the full Ada specification for each unit in the standard Environment. The unit specifications are organized by their pathnames. The World ! section provides a list of the units in the library system of the Environment and the manual/book in which they are documented.
- **Keymap:** The Rational Environment Keymap presents the standard Environment key bindings, organized by topic and by command name. The topical section includes both a quick reference for commonly used commands and a more detailed reference for key bindings.
- **Master Index:** The Master Index combines all of the index information for each of the books in the Reference Manual.

### Volumes 2-11

Each book in Volumes 2-11 begins with a colored tab on which the name of the book appears. Each book typically contains the following sections:

- **Contents:** The table of contents provides a complete list of all the units in the book and their reference entries.
- **Key Concepts section:** Most of the books contain a section describing key concepts that pertain to all of the Environment facilities documented in that book. This section is located behind its own tab after the table of contents.
- **Unit sections:** Each of the commands, tools, and so on has a declaration within an Ada compilation unit (typically a package) in the Environment library system. For each unit, there is a section that contains reference entries for the declarations (for example, procedures, functions, and types) within that unit. Each section is preceded by a tab.

The sections for units are alphabetized by the simple names of the units. For example, the section for package !Tools.String-Utilities is alphabetized under String-Utilities.

For many units, introductory material and/or examples specific to the unit appear after the section tabs.

Within the section for a given unit, the reference entries describing the unit's declarations are organized alphabetically after the section introduction. Appearing at the top of each page in a reference entry are the simple name of the given declaration and the fully qualified pathname of the enclosing unit.

- **Explanatory/topical sections:** Like the unit sections, explanatory/topical sections are preceded by tabs, and they are alphabetized with the unit sections. The topical sections, such as Help, located in Editing Specific Types (EST), discuss Environment facilities.
- **Index:** Preceded by a tab, the Index appears as the last section of each book. It contains entries for each unit or declaration, along with additional topical references. Each book index covers only the material documented in that particular book. The Master Index (in Volume 1) provides entries for the information documented in all the books within the Reference Manual.

Italic page numbers indicate the page on which the primary reference entry for a declaration appears; nonitalic page numbers indicate key concepts, defined terms, cross-references, and exceptions raised.

## Suggestions for Finding Information

The following suggestions may help you in finding various kinds of information in the documentation for Rational's products.

### Learning about Environment Facilities

If you are a novice user starting to use the Environment, consult the *Rational Environment User's Guide*.

If you are familiar with the Environment but are interested in learning about the Environment's library-management commands, for example, you might start by scanning the specifications for these units in the Reference Summary to get an idea of the kinds of things these tools can do. You should also look at the Key Concepts for the particular book, which describes important concepts and gives examples.

It may also be useful to glance through the introductions provided for some of the units in the book. These introductions, located immediately after the tabs for the units, often contain helpful examples.

### Finding Information on a Specific Item

If you know the name of the item and the book in which it is documented, consult either the table of contents or the index for that book. You can also turn through the pages of the book using the names and pathnames of the reference entries to locate the entry you want. Remember that the reference entries for a unit are organized alphabetically within the unit, and the units are organized alphabetically by simple name within the book.

If you know the simple name of the entry but do not know the book in which it is documented, look in the Master Index (in Volume 1) to find the book abbreviation and page number.

If you know the pathname of the entry but do not know the book in which it is documented, the World ! section of the Reference Summary (in Volume 1) provides a map of the units in the library system of the Environment and the books in which they are documented.

If you cannot find an item in the Master Index, the item either is not documented or is documented in the manuals for a product other than the Rational Environment (for example, Rational Networking—TCP/IP or Rational Target Build Utility). If you know the pathname, consult the World ! section of the Reference Summary to determine whether that item is documented and in which manual.

### Using the Index

The index of each book contains entries for each unit and its declarations, organized alphabetically by simple name. When using the index to find a specific item, consult the italic page number for the primary reference for that item. Nonitalic page numbers indicate key concepts, defined terms, cross-references, and exceptions raised.

### Viewing Specifications On-Line

If you know the pathname of a declaration and want to see its specification in a window of the Rational Environment, provide its pathname to the Common-Definition procedure—for example, Definition ("!Commands.Library");. If you know the simple name of the unit in which the declaration appears, in most cases you can use searchlist naming as a quick way of viewing the unit—for example, Definition ("\Library");.

### Using On-Line Help

Most of the information contained in the reference entries for each unit is available through the on-line help facilities of the Environment. Press the **Help on Help** key or consult the *Rational Environment User's Guide* or the *Rational Environment Reference Manual*, EST, Help, for more information on using this on-line help facility.

### Cross-Reference Conventions

The following conventions are used in cross-references to information:

- **Specific page/book:** For references to a specific place in a specific book, the book abbreviation is followed by the page number in the book (for example, LM-322). If the book abbreviation is omitted, the current book is implied (for example, the page numbers in the table of contents for a book do not include the book prefix).
- **Declaration in same unit:** References to the documentation for a declaration in the same unit are indicated by the simple name of the desired declaration. For example, within the reference entry for the Library.Copy procedure, a reference to the Library.Move procedure would be simply "procedure Move." Note that if there are nested packages in the unit, references to nested declarations use qualified pathnames.
- **Declaration in different unit, same book:** References to the documentation for a declaration in another unit are indicated by the qualified pathname of the desired declaration. For example, within the reference entry for the Library.Copy procedure, a reference to the Compilation.Delete procedure would be "procedure Compilation.Delete."



- **Declaration in different book:** References to the documentation for a declaration in another book are indicated by the addition of the abbreviation for that book. For example, within the reference entry for the Library.Copy procedure, a reference to the Editor.Region.Copy procedure in the Editing Images book would be "EI, procedure Editor.Region.Copy."

References to specific declarations in the library system of the Rational Environment (not the documentation for them) are typically indicated by fully qualified pathnames—for example, "procedure !Commands.Library.Copy." When the context is clear, however, a shorter name will be used. If the unit in which the declaration appears is undocumented, you may want to see its explanatory comments to understand what it does. To see these comments, either look at the unit's specification in the Reference Summary or view it on-line using the Rational Environment.

### **Feedback to Rational: Reader's Comments Form**

Rational wants to make its documentation as useful and error-free as possible. Please provide us with feedback. The last page of each book contains a Reader's Comments form that you can use to send us comments or to report errors. You can also submit problem reports and make suggestions electronically by using the SIMS problem-reporting system. If you use SIMS to submit documentation comments, please indicate the manual name, book name, and page number.

RATIONAL

MAP OF THE RATIONAL ENVIRONMENT LIBRARY SYSTEM

Name	Kind	Book Abbreviation For Documentation
<u>! : Library (World):</u>		
Commands	: Library (World);	
Compiler_Interface	: Library (World);	
Implementation	: Library (World);	
Io	: Library (World);	
Local	: Library (World);	
Lrm	: Library (World);	
Machine	: Library (World);	
Model	: Library (World);	
Software_Catalogs	: Library (World);	SW Library Catalog
Tools	: Library (World);	
Users	: Library (World);	
<u>!Commands : Library (World):</u>		
Abbreviations	: Library (World);	Reference Summary
Access_List	: Ada (Pack_Spec);	LM
Action_Utillities	: Ada (Pack_Spec);	n/a
Activity	: Ada (Pack_Spec);	PM
Ada	: Ada (Pack_Spec);	EST
Archive	: Ada (Pack_Spec);	LM
Cmvc	: Ada (Pack_Spec);	PM
Cmvc_Maintenance	: Ada (Pack_Spec);	PM
Command	: Ada (Pack_Spec);	EST
Common	: Ada (Pack_Spec);	EST
Compilation	: Ada (Pack_Spec);	LM
Daemon	: Ada (Pack_Spec);	SMU
Debug	: Ada (Pack_Spec);	DEB
Diana_Tree	: Ada (Pack_Spec);	n/a
Disk_Space	: Ada (Pack_Spec);	n/a
Editor	: Ada (Pack_Spec);	EI
File_Utillities	: Ada (Pack_Spec);	LM
Ftp	: Ada (Pack_Spec);	FTP
Job	: Ada (Pack_Spec);	SJM
Library	: Ada (Pack_Spec);	LM
Links	: Ada (Pack_Spec);	LM
Log	: Ada (Pack_Spec);	SJM
Message	: Ada (Pack_Spec);	SMU
Network	: Ada (Pack_Spec);	TRL
Operator	: Ada (Pack_Spec);	SMU
Program	: Ada (Pack_Spec);	SJM
Queue	: Ada (Pack_Spec);	SMU
Scheduler	: Ada (Pack_Spec);	SMU
Search_List	: Ada (Pack_Spec);	SJM
Sims	: Library (Subsystem);	System Manager's G.
Switches	: Ada (Pack_Spec);	LM
System_Backup	: Ada (Pack_Spec);	SMU

```

System_Maintenance      : Library (Subsystem);
.Revn.Units.Check_Universe_Acls : Ada (Proc_Spec);      n/a
.Revn.Units.Find_Null_Acls   : Ada (Proc_Spec);      n/a
.Revn.Units.Set_Universe_Acls : Ada (Proc_Spec);      n/a
.Revn.Units.Show_Groups     : Ada (Proc_Spec);      n/a
.Revn.Units.Show_Identity   : Ada (Proc_Spec);      n/a
.Revn.Units.Show_Jobs       : Ada (Proc_Spec);      n/a
.Revn.Units.Show_Job_Names  : Ada (Proc_Spec);      n/a
.Revn.Units.Show_Locks     : Ada (Proc_Spec);      n/a
.Revn.Units.Show_Machine_Id : Ada (Proc_Spec);      n/a
.Revn.Units.Show_Memory_Hogs : Ada (Proc_Spec);      n/a
.Revn.Units.Show_Stats     : Ada (Proc_Spec);      n/a
.Revn.Units.Show_Tasks     : Ada (Proc_Spec);      n/a
Tape                    : Ada (Pack_Spec);      SMU
Telnet                  : Ada (Pack_Spec);      TEL
Terminal                : Ada (Pack_Spec);      SMU
Text                    : Ada (Pack_Spec);      EST
Transport_Route        : Ada (Pack_Spec);      TRL
What                    : Ada (Pack_Spec);      SJM
Work_Order              : Ada (Pack_Spec);      PM

```

!Io : Library (World):

```

Device_Independent_Io   : Ada (Pack_Spec);      n/a
Direct_Io               : Ada (Gen_Pack);      DIO
Io                      : Ada (Pack_Spec);      TIO
Io_Exceptions           : Ada (Pack_Spec);      DIO, TIO
Object_Set              : Ada (Pack_Spec);      n/a
Pipe                    : Ada (Pack_Spec);      n/a
Polymorphic_Io         : Ada (Pack_Spec);      n/a
Polymorphic_Sequential_Io : Ada (Pack_Spec);      DIO
Sequential_Io           : Ada (Gen_Pack);      DIO
Tape_Specific           : Ada (Pack_Spec);      n/a
Terminal_Specific       : Ada (Pack_Spec);      n/a
Text_Io                 : Ada (Pack_Spec);      TIO
Window_Io               : Ada (Pack_Spec);      DIO

```

!Lrm : Library (World):

```

Calendar                : Ada (Pack_Spec);      PT
--Standard              : Ada (Pack_Spec);      PT
System                  : Ada (Pack_Spec);      PT
Unchecked_Conversion    : Ada (Gen_Func);      PT
Unchecked_Deallocation  : Ada (Gen_Proc);      PT

```

!Machine : Library (World):

```

Accounting              : Library (World);
.Enabled                : File;
Cg_Data                 : Library (World);
Devices                 : Library (World);
Editor_Data             : Library (World);
.Facit_Commands        : Ada (Proc_Spec);
.Facit_Commands        : Ada (Proc_Body);      Facit Keymap
.Facit_Keys            : File;
.Facit_User_Commands   : File;
.Help_Data              : Library (Directory);
.Rational_Commands     : Ada (Proc_Spec);
.Rational_Commands     : Ada (Proc_Body);      Rational Keymap
.Rational_Keys         : File;
.Rational_User_Commands : File;
.Session_Switch_Help   : File;

```

.Visible_Key_Names	: Ada (Pack_Spec);	
.Vt100_Commands	: Ada (Proc_Spec);	Rational Keymap
.Vt100_Commands	: Ada (Proc_Body);	
.Vt100_Keys	: File;	
.Vt100_User_Commands	: File;	
Error_Logs	: Library (World);	
Groups	: Library (World);	
Initialize	: Ada (Proc_Spec);	
Initialize	: Ada (Proc_Body);	System Manager's G.
Initialize_Cross_Compilers	: Ada (Proc_Spec);	
Initialize_Cross_Compilers	: Ada (Proc_Body);	
Initialize_Daemons	: Ada (Proc_Spec);	
Initialize_Daemons	: Ada (Proc_Body);	
Initialize_Housekeeping	: Ada (Proc_Spec);	
Initialize_Housekeeping	: Ada (Proc_Body);	
Initialize_Network	: Ada (Proc_Spec);	
Initialize_Network	: Ada (Proc_Body);	
Initialize_Servers	: Ada (Proc_Spec);	
Initialize_Servers	: Ada (Proc_Body);	
Initialize_Site	: Ada (Proc_Spec);	
Initialize_Site	: Ada (Proc_Body);	
Initialize_Terminals	: Ada (Proc_Spec);	
Initialize_Terminals	: Ada (Proc_Body);	
Machine_Name	: File;	
Operator_Capability	: File (Text);	
Queues	: Library (World);	
Release	: Library (World);	
.Current.Activity	: File (Activity);	
.Current.Commands	: Library (World);	
.Login	: Ada (Proc_Spec);	
.Login	: Ada (Proc_Body);	
Search_Lists	: Library (World);	
.Default	: File;	
Shutdown_Help_File	: File (Text);	
Tcp_Ip_Host_Id	: File (Text);	
Temporary	: Library (World);	
Transport_Name_Map	: File (Text);	
Users	: Library (World);	
User_Acl_Suffix	: File (Text);	
User_Default_Acl_Suffix	: File (Text);	
 <u>!Model : Library (World):</u>		
R1000	: Library (World);	Reference Summary
R1000_Implementation	: Library (World);	Reference Summary
R1000_Portable	: Library (World);	Reference Summary
 <u>!Tools : Library (World):</u>		
Access_List_Tools	: Ada (Pack_Spec);	LM
Ada_Object_Editor	: Ada (Pack_Spec);	n/a
Ada_Text	: Ada (Pack_Spec);	n/a
Allows_Deallocation	: Ada (Gen_Func);	PT
Bit_Operations	: Ada (Pack_Spec);	n/a
Bounded_String	: Ada (Pack_Spec);	ST
Ci	: Library (Subsystem);	
.Revn.Units.Ci	: Ada (Pack_Spec);	System Manager's G.
.Revn.Units.Commands	: Library (Directory);	
.Run	: Ada (Pack_Spec);	System Manager's G.
.Show	: Ada (Pack_Spec);	System Manager's G.
.Typ	: Ada (Proc_Spec);	System Manager's G.

Compatibility	: Library (Subsystem);	
.Revn.Units.Check	: Ada (Pack_Spec);	PM
Concurrent_Map_Generic	: Ada (Gen_Pack);	PT
Debug_Tools	: Ada (Pack_Spec);	DEB
Diana_Object_Editor	: Ada (Pack_Spec);	n/a
Directory_Tools	: Ada (Pack_Spec);	n/a
Disk_Daemon	: Ada (Pack_Spec);	n/a
Hash	: Ada (Pack_Spec);	PT
Library_Object_Editor	: Ada (Pack_Spec);	n/a
Link_Tools	: Ada (Pack_Spec);	n/a
List_Generic	: Ada (Gen_Pack);	PT
Map_Generic	: Ada (Gen_Pack);	PT
Networking	: Library (Directory);	
.Byte_Defs	: Ada (Pack_Spec);	TRL
.Byte_String_Io	: Ada (Pack_Spec);	n/a
.Exos_8010_3_Sd	: File;	
.File_Transfer	: Ada (Pack_Spec);	FTP
.Ftp_Defs	: Ada (Pack_Spec);	FTP
.Ftp_Name_Map	: Ada (Pack_Spec);	FTP
.Ftp_Product	: Ada (Pack_Spec);	FTP
.Ftp_Profile	: Ada (Pack_Spec);	FTP
.Ftp_Server	: Ada (Pack_Spec);	n/a
.Host_Id_Io	: Ada (Pack_Spec);	TRL
.Interchange	: Ada (Pack_Spec);	RPC
.Interchange_Defs	: Ada (Pack_Spec);	RPC
.Network_Product	: Ada (Pack_Spec);	TRL
.Rpc	: Ada (Pack_Spec);	RPC
.Rpc_Access_Utilityies	: Ada (Pack_Spec);	RPC
.Rpc_Client	: Ada (Pack_Spec);	RPC
.Rpc_Product	: Ada (Pack_Spec);	RPC
.Rpc_Server	: Ada (Pack_Spec);	RPC
.Tcp_Ip_Boot	: Ada (Proc_Spec);	TRL
.Tcp_Ip_Dump	: Ada (Proc_Spec);	n/a
.Telnet_Product	: Ada (Pack_Spec);	n/a
.Telnet_Profile	: Ada (Pack_Spec);	TEL
.Transfer_Generic	: Ada (Gen_Pack);	FTP
.Transport	: Ada (Pack_Spec);	TRL
.Transport_Defs	: Ada (Pack_Spec);	TRL
.Transport_Interchange	: Ada (Pack_Inst);	RPC
.Transport_Name	: Ada (Pack_Spec);	TRL
.Transport_Server	: Ada (Gen_Pack);	RPC
.Transport_Server_Job	: Ada (Pack_Spec);	RPC
.Transport_Stream	: Ada (Pack_Spec);	RPC
Object_Editor	: Ada (Pack_Spec);	n/a
Parameter_Parser	: Ada (Gen_Pack);	n/a
Profile	: Ada (Pack_Spec);	SJM
Queue_Generic	: Ada (Gen_Pack);	PT
Random	: Ada (Pack_Spec);	n/a
Rpc_Servers	: Library (Subsystem);	
.Revn.Units.Queue_Service	: Ada (Pack_Spec);	n/a
Script	: Ada (Pack_Spec);	n/a
Set_Generic	: Ada (Gen_Pack);	PT
Simple_Status	: Ada (Pack_Spec);	PT
Stack_Generic	: Ada (Gen_Pack);	PT
String_Map_Generic	: Ada (Gen_Pack);	ST
String_Table	: Ada (Pack_Spec);	ST
String_Utilityies	: Ada (Pack_Spec);	ST

System_Availability	: Library (Subsystem);	System Manager's G.
.Revn.Units.Log_Reader	: Ada (Pack_Spec);	
.Revn.Units.Outage_Information	: Ada (Pack_Spec);	
.Revn.Units.Sample_Logs	: Library (Directory);	
.Revn.Units.Show_Error_Log	: Ada (Proc_Spec);	
.Revn.Units.System_Information	: Ada (Pack_Spec);	
.Revn.Units.System_Report	: Ada (Pack_Spec);	
System_Utilities	: Ada (Pack_Spec);	SMU
Table_Formatter	: Ada (Gen_Pack);	ST
Table_Sort_Generic	: Ada (Gen_Proc);	PT
Tape_Tools	: Ada (Pack_Spec);	n/a
Target_Build_Utility	: Library (Subsystem);	
.Revn.Units.Target_Builder	: Ada (Gen_Pack);	TBU
.Revn.Units.Vax_Builder	: File (Text);	TBU
Time_Utilities	: Ada (Pack_Spec);	PT
Unbounded_String	: Ada (Gen_Pack);	ST
Unchecked_Conversions	: Ada (Pack_Spec);	PT
Xref_Utility	: Library (Subsystem);	
.Revn.Units.Commands.Xref	: Ada (Pack_Spec);	LM

RATIONAL



```

package Access_List is
    subtype Name is String; -- an object name
    Read : constant Character := 'R'; -- objects and worlds
    Write : constant Character := 'W'; -- objects only
    Delete : constant Character := 'D'; -- worlds only; same bit as W
    Create : constant Character := 'C'; -- worlds only
    Owner : constant Character := 'O'; -- worlds only

    subtype Acl is String;
    -- String representations of access lists have the following syntax:
    -- Acl ::= Acl_Entry [' ', Acl_Entry]*
    -- Acl_Entry ::= Group '=', Access
    -- Group ::= Identifier
    -- Access ::= Acc_Type+
    -- Acc_Type ::= 'R' | 'W' | 'D' | 'C' | 'O' |
    --             'r' | 'w' | 'd' | 'c' | 'o'
    -- Examples: "Phil => R, TRW => rv", "Public=>RCOD"

    procedure Display (For_Object : Name := "<CURSOR>");
    -- Display the access list of the specified object(s).
    -- Output and error messages are sent to current output.

    procedure Set (To_List : Acl := "Network_Public => RMCOD";
                  For_Object : Name := "<SELECTION>";
                  Response : String := "<PROFILE>");
    -- Set the access list for the specified object(s).
    -- Setting the access list requires "Owner" access to the containing world.
    -- Sends messages to a log that is under control of the Response parameter.

    procedure Default_Display (For_World : Name := "<CURSOR>");
    -- Display the default acl of the specified world(s) in an output window.
    -- Error messages are sent to the window in case of any error.
    -- Wildcards in the name are allowed.
    -- Non-world objects are filtered out of the display.
    -- A null display is produced if no worlds are referenced.

    procedure Set_Default (To_List : Acl := "Network_Public => RW";
                           For_World : Name := "<SELECTION>";
                           Response : String := "<PROFILE>");
    -- Set the default ACL for the specified world(s).
    -- Owner access to each world is required.
    -- Sends messages to a log that is under control of the Response parameter.
    -- A log is written indicating success or errors.
    -- Wildcards are allowed in the name.
    -- Any non-world objects referenced are ignored.
    -- A summary of the number of objects affected is included in the log.

    procedure Add (To_List : Acl := "Network_Public => RMCOD";
                   For_Object : Name := "<SELECTION>";
                   Response : String := "<PROFILE>");
    -- Add the access list to the existing value for the specified object(s).
    -- Changing the access list requires "Owner" access to the containing world.

```

Access\_List, !Commands

```

-- Sends messages to a log that is under control of the
-- Response parameter.

procedure Add_Default (To_List : Acl := "Network_Public => RW";
                       For_World : Name := "<SELECTION>";
                       Response : String := "<PROFILE>");
    -- Add the default ACL to the existing value for the specified world(s).
    -- Owner access to each world is required.
    -- Sends messages to a log that is under control of the Response parameter.
    -- A log is written indicating success or errors.
    -- Wildcards are allowed in the name.
    -- Any non-world objects referenced are ignored.
    -- A summary of the number of objects affected is included in the log.

pragma Subsystem (Os_Commands);
pragma Module_Name (4, 3507);

end Access_List;

```

C-1

```

with Action;
with Directory;
with Machine;

package Action_Uutilities is

  procedure Display_Action (Id : Action.Id);
  procedure Display_Action (Id : Integer);
  -- displays either not in progress or put_task_info (creating task_id)
  -- the second form converts the integer to an action.id and
  -- invokes the first form

  procedure Lock_Information (Version : Directory.Version);
  procedure Lock_Information
    (Name : Directory.Maming_Name := "<Image>";
     Version : Directory.Version_Name := Directory.All_Versions);
  -- displays the following information
  -- actions (if any) that have a read lock on the version
  -- action (if any) that has an update lock on the version
  -- action (if any) that has an overwrite lock on the version
  -- request queue of [task,action,mode] triples waiting for the object
  -- the second form does name resolution and then calls the first form

  procedure Display_Task (Task_Id : Machine.Task_Id);
  -- shows the user, session and job for the specified task

  procedure Display_Object (Version : Directory.Version);
  procedure Display_Object
    (Class : Natural; Instance : Natural; Host : Machine.Id);
  -- displays the name and version of the specified object
  -- the second form construct a directory.version and calls the first form

  procedure Lock_Information
    (Class : Natural; Instance : Natural; Host : Machine.Id);

  pragma Subsystem (Os_Commands);
  pragma Module_Name (4, 3933);

end Action_Uutilities;

```

Action\_Uutilities, !Commands

```

package Activity is

  pragma Subsystem (Commands);
  pragma Module_Name (4, 3940);

  subtype Activity_Name is String;

  -- An Activity is a managed object that maintains a map between
  -- subsystems and pairs of views. The pair consists of a spec view and
  -- a load (non-spec) view of the subsystem. An activity name is a
  -- string name for the managed object. The view pair can be specified
  -- indirectly by associating a subsystem in one activity with another
  -- activity, which then maps the subsystem to a pair of views.

  -- In these Activity commands, the default Activity is the object
  -- selected in the accompanying window, the object associated with the
  -- accompanying window, or, as a last resort, The_Current_Activity.

  type Creation_Mode is (Differential, Exact_Copy, Value_Copy);

  -- When a subsystem is copied from one Activity to another, the entry
  -- in the destination activity can be created in three ways:

  -- Differential : In the destination activity, the subsystem is mapped
  -- to the source Activity.

  -- Exact_Copy : In the destination activity, the subsystem is mapped
  -- to the same object it mapped to in the source
  -- activity; this may be either a view or an activity.

  -- Value_Copy : In the destination activity, the subsystem is mapped
  -- to the view currently associated with the subsystem
  -- in the source activity.

  subtype Subsystem_Name is String;
  -- String name of a World directory.

  subtype View_Simple_Name is String;
  subtype View_Name is String;
  -- View_Name = Subsystem_Name & '.' & View_Simple_Name

  -- A View is a world whose enclosing world is a Subsystem world.
  -- Any number of directories may come between a view and its subsystem.
  -- Hence, the view's subsystem is implicit in the full name of the
  -- view. The simple name of the view is used where the name of the
  -- subsystem is easily derived from other parameters.

  subtype View_Or_Activity_Name is String;
  -- An activity can be used to indirectly specify a view.

  subtype Unit_Name is String;
  -- The string name for an Ada library unit nested within a view of a
  -- subsystem.

  function Nil return Activity_Name;

```

Activity, !Commands

```

-- The name of the canonical activity with no subsystems;
-- the empty activity.

procedure Current (Response : String := "<PROFILE>");
-- Prints the name of the activity currently associated with the
-- running job; if no Activity has been associated with the job, it
-- then returns the Activity currently associated with the running
-- session.

function The_Current_Activity return Activity_Name;
-- returns the name of the current activity; as defined above.

procedure Set (The_Activity : Activity_Name := "<ACTIVITY>";
              Response : String := "<PROFILE>");
-- Makes The_Activity the current activity for the running job only.

procedure Set_Default (The_Activity : Activity_Name := "<ACTIVITY>";
                      Response : String := "<PROFILE>");
-- Makes Activity the current activity for the session. If the job's
-- activity is nil, set that as well.

procedure Enclosing_View (Unit : Unit_Name := "<IMAGE>";
                          Response : String := "<PROFILE>");
-- Prints the name of the enclosing view (either a load or spec view);

function The_Enclosing_View
  (Unit : Unit_Name := "<IMAGE>") return View_Name;
-- The name of the enclosing view (either a load or spec view);

procedure Enclosing_Subsystem (View : View_Name := "<IMAGE>";
                               Response : String := "<PROFILE>");
-- Prints the name of the subsystem that encloses the View, which may
-- be either a Spec or Load view.

function The_Enclosing_Subsystem
  (View : View_Name := "<IMAGE>") return Subsystem_Name;
-- The name of the subsystem that encloses the View, which may
-- be either a Spec or Load view.

procedure Create (The_Activity : Activity_Name := ">>ACTIVITY NAME<<";
                 Source : Activity_Name := Activity.Nil;
                 Mode : Creation_Mode := Activity.Exact_Copy;
                 Response : String := "<PROFILE>");
-- Create a new Activity object. If the Source activity is not Nil,

```

```

-- its contents are copied to the new activity according to the
-- specified Mode.

procedure Add (Subsystem : Subsystem_Name := "<CURSOR>";
             Load_Value : View_Or_Activity_Name := Activity.Nil;
             Spec_Value : View_Or_Activity_Name := Activity.Nil;
             The_Activity : Activity_Name :=
               Activity.The_Current_Activity;
             Mode : Creation_Mode := Activity.Exact_Copy;
             Response : String := "<PROFILE>");
-- Add a subsystem to an existing Activity. If the load or spec values
-- are activities, the mapping is created according to the specified
-- mode. The Load_Value and Spec_Value names are resolved in the
-- context of the given Subsystem, so that View_Simple_Names may be
-- used.

procedure Remove (Subsystem : Subsystem_Name := "<SELECTION>";
                 The_Activity : Activity_Name :=
                   Activity.The_Current_Activity;
                 Response : String := "<PROFILE>");
-- Remove a subsystem from an Activity.

procedure Set_Spec_View (Spec_View : View_Or_Activity_Name := "<CURSOR>";
                       Subsystem : Subsystem_Name := "";
                       Mode : Creation_Mode := Activity.Differential;
                       The_Activity : Activity_Name :=
                         Activity.The_Current_Activity;
                       Response : String := "<PROFILE>");
-- If Spec_View designates a view, associates the given view as the spec
-- view for the subsystem that contains the view.

-- If Spec_View designates an activity, associates the spec view defined
-- in the given source activity as the new spec view of the given
-- subsystem in the destination Activity. The mapping is created
-- according to the given Mode.

-- The Spec_View parameter is resolved in the context established by the
-- Subsystem parameter. The subsystem is derived from the Spec_View
-- parameter if it denotes a view, otherwise the Subsystem parameter
-- must be given.

procedure Set_Load_View (Load_View : View_Or_Activity_Name := "<CURSOR>";
                        Subsystem : Subsystem_Name := "";
                        Mode : Creation_Mode := Activity.Differential;
                        The_Activity : Activity_Name :=
                          Activity.The_Current_Activity;
                        Response : String := "<PROFILE>");
-- If Load_View designates an activity, associates the given View as the
-- load view for the subsystem that contains the view.

-- If Load_View designates an activity, associates the load view defined
-- in the given source activity as the new load view of the given
-- subsystem in the named Activity. The mapping is created according to the

```

```
-- given Mode.
-- The Load_View parameter is resolved in the context established by the
-- Subsystem parameter. The subsystem is derived from the Load_View
-- parameter if it denotes a view, otherwise the Subsystem parameter
-- must be given.
```

```
procedure Display (Subsystem : Subsystem_Name := "?";
  Spec_View : View_Name := "?";
  Load_View : View_Name := "?";
  Mode : Creation_Mode := Activity_Value_Copy;
  The_Activity : Activity_Name :=
    Activity_The_Current_Activity;
  Response : String := "<PROFILE>");
-- Display the mappings between subsystems and views defined by the
-- given activity. Only the mappings that match the patterns given in
-- the Subsystem, Spec_View, and Load_View parameters are listed. (The
-- default is to list all mappings in the activity.) In the Value_Copy
-- mode, all indirect references are resolved and only the resolution
-- is displayed. In the Exact_Copy mode, indirect mappings are not
-- resolved and the name of the source activity is displayed. In the
-- Differential mode, the indirect mappings are resolved and both the
-- resolution and the original indirect activity are displayed.
```

```
procedure Edit (The_Activity : Activity_Name := "<ACTIVITY>");
-- Invoke the Activity object editor on the given Activity.
```

```
procedure Insert (Subsystem : Subsystem_Name := ">>SUBSYSTEM NAME<<";
  Spec_View : View_Or_Activity_Name := "";
  Load_View : View_Or_Activity_Name := "");
```

```
-- Inserts the specified subsystem mapping into the activity associated
-- with the command window. (The current activity is brought up in an
-- Activity window and modified if the command is not associated with
-- an Activity window). The given names may specify a view or another
-- activity. If the subsystem name is omitted, it is inferred from the
-- view names.
```

```
procedure Change (Spec_View : View_Or_Activity_Name := "";
  Load_View : View_Or_Activity_Name := "");
```

```
-- The selected subsystem mapping is changed to the new values given in
-- the Views specification. Valid only in an Activity window.
```

```
procedure Write (File : Activity_Name := ">>ACTIVITY NAME<<");
```

```
-- Copies the current content of the Activity window to the designated
-- File. Valid only in an Activity window.
```

```
procedure Visit (The_Activity : Activity_Name := "<ACTIVITY>");
```

```
-- Same as Edit, except that if the command is given on an activity
-- window, the new activity is displayed in that window rather than in
-- a new one.
```

```
procedure Merge (Source : Activity_Name := ">>ACTIVITY NAME<<");
```

```
Subsystem : Subsystem_Name := "?";
Spec_View : View_Name := "?";
Load_View : View_Name := "?";
Mode : Creation_Mode := Activity_Exact_Copy;
Target : Activity_Name := "<ACTIVITY>";
Response : String := "<PROFILE>");
```

```
-- The subsystem mappings defined in the Source Activity that match the
-- given subsystem and view patterns are copied to the Target activity
-- according to the specified Creation mode. New subsystems are added
-- to the Target activity if necessary; Existing subsystem mappings are
-- replaced. The default Target activity is the current selection/image.
```

```
end Activity;
```

```

package Ada is
  procedure Code_Unit;
  -- Bring the unit corresponding to current image to the coded state.
  -- May involve coding subunits, parent unit, or corresponding visible
  -- part, but no closure operation is performed. If the operation
  -- succeeds, the unit will be read-only.

  procedure Install_Unit;
  -- Bring the unit corresponding to current image to the installed
  -- state. Will install no other units; may reduce subunits or parent
  -- unit to installed, but no closure operation is performed. If the
  -- operation succeeds, the unit will be read-only.

  procedure Source_Unit;
  -- Bring the unit to source state such that its library declaration has
  -- the appropriate name and the image is read-only.

  procedure Withdraw (Name : String := "<IMAGE>");
  -- Edit the indicated unit, removing its declaration from the library.

  procedure Diana_Edit (Name : String := "<CURSOR>");
  -- Show a read-only image of the internal form of the Diana tree
  -- corresponding to the image given.

  procedure Install_Stub;
  procedure Make_Inline;
  -- Make a separate subunit body into an inline unit body

  procedure Make_Separate;
  -- Make an inline subunit body be a separate subunit body

  procedure Other_Part (Name : String := "<IMAGE>";
    In_Place : Boolean := False);
  -- If a new window is required, In_Place indicates that the current
  -- frame should be used.

  procedure Replace_Id (Old_Id : String := ">>OLD NAME<<";
    New_Id : String := ">>NEW NAME<<");
  -- For the current selection, change all occurrences of Old_Id into
  -- occurrences of New_Id. Only changes Ada identifier references that
  -- match exactly.

  procedure Show_Usage (Name : String := "<CURSOR>";
    Global : Boolean := True;
    Limit : String := "<ALL_WORLDS>";
    Closure : Boolean := False);
  -- Show uses of the indicated item.
  -- Global => mark units other than the one indicated.
  -- Limit specifies the range of units if Global is true.
  -- Closure causes Show_Usage to find indirect references, e.g. renames.

  procedure Show_Unused (In_Unit : String := "<IMAGE>";
    Check_Other_Units : Boolean := True);
  -- Show the declarations in a unit that are not referenced

  procedure Create_Body (Name : String := "<IMAGE>");
  -- Create a body declaration corresponding to the indicated
  -- declaration or visible part.

```

```

  procedure Create_Private (Name : String := "<IMAGE>");
  -- Create a private part declaration for each private type that still
  -- requires one.

  procedure Get_Errors;
  -- Restore the error underlining from the last compile, semantimize,
  -- etc.

  procedure Insert_Blank_Line (Repeat : Positive := 1);
  -- Insert repeat blank lines before the current line

  procedure Delete_Blank_Line (Repeat : Positive := 1);
  -- Delete repeat blank lines at the current cursor

  pragma Subsystem (Command);
  pragma Module_Name (4, 209);
end Ada;

```

with Machine;  
package Archive is

```
procedure Save (Objects : String := "<IMAGE>";
               Device : String := "R1000";
               Response : String := "MACHINE.DEVICES.TAPE_0";
               Profile : String := "<PROFILE>");
-- Save a set of objects (files, Ada units, etc.) to a tape or directory
-- such that they may be restored to their original form at a later time
-- or on another system.
-- The Objects parameter specifies the primary objects to be saved. It
-- can be any naming expression. By default, the current image is saved
-- unless there is a selection on that image, in which case the selected
-- object is saved. Normally, the specified object(s) and all contained
-- objects are archived; this feature can be disabled.
-- The Options parameter specifies the type of tape to be written and
-- options to control what is saved. The Options parameter for each of
-- the Archive operations is written as a sequence of option
-- names separated by spaces or commas. Options with arguments are
-- given as an option name followed by an equal sign followed by a
-- value.
-- The save options are:
-- FORMAT = R1000 | R1000_LONG | ANSI
-- R1000
-- Writes an ANSI tape with the data file followed by the index
-- file. The images of the objects being saved are written
-- directly to the tape. This is the default.
-- R1000_LONG
-- like R1000 format but the data file is written to one ANSI tape
-- and the index file to a second ANSI tape.
-- ANSI
-- Writes the data to a temporary file and then writes both index
-- and data file to a tape using ANSI tape facilities.
-- LABEL=(any balanced string)
-- An identifying string written at the head of the archived data.
-- The label parameter allows the user to specify a string that
-- will be put at the front of the index file. When a restore is
-- done the label specified to the restore procedure will be
-- checked against the one on the save tape.
-- NONRECURSIVE
-- Save only the objects resolved to by the Objects parameter. Do
-- not recursively save objects that are inside of other objects.
-- The default is to save the objects mentioned in the Objects
-- parameter and all objects contained in them.
-- To save a world and a subset of its contents one can say:
-- Save (Objects => "[HJL?,"[HJL.ABC?,"[HJL.DEF?]", ...
-- Options => "R1000 NONRECURSIVE");
```

Archive, !Commands

```
-- AFTER=<time_expression>
-- Only objects changed after the time represented by
-- <time_expression> will be archived. The <time_expression>
-- should be acceptable to the time_utilities.value function.
--
-- COMPATIBILITY_DATABASE (CDB) [=Subsystems]
-- Causes the full compatibility database for each subsystem
-- specified to be archived. If no subsystems are specified with
-- the option, the Objects parameter specification is used instead.
-- The NONRECURSIVE option does not affect the interpretation of the
-- CDB specification even when it is obtained from the Objects
-- parameter.
--
-- When Ada units in a subsystem are archived, the relevant
-- portions of the subsystem Compatibility Database is
-- automatically archived with them. Therefore, this option is
-- required only in special situations, primarily when one needs to
-- "sync up" a primary and a secondary subsystem.
--
-- To archive just Compatibility Databases, use
-- Save ("Subsystems", "CDB");
--
-- To archive compatibility databases with other objects, use
-- Save ("Other Stuff", "CDB=Subsystems");
--
-- The "Subsystems" and "Other Stuff" specifications will usually
-- describe disjoint sets of objects.
--
-- PREFIX=<naming pattern>
-- A naming pattern that is saved with the archived objects, which
-- can be recalled as the For_Prefix when the data is Restored.
-- When set to an appropriate value, the restorer need not know
-- exactly the names of the archived objects to be able to restore
-- them to a new place. If this option is not given, the value
-- used is derived from the Objects parameter and CDB
-- option (if present) by expanding context-sensitive characters
-- (such as ` and $), expanding indirect file references, and
-- removing all attributes.
--
-- For downward compatibility the following options are provided.
--
-- GAMMA0
-- write a tape which can be read on a Gamma0 system.
--
-- GAMMAL
-- write a tape which can be read on a Gamma1 system.
--
-- VERSION=<archive_version_number>
-- write a tape that can be read by a version of source
-- earlier than the current one. The argument is a three digit
-- integer. For example, version=210.
--
-- The Device parameter can be set to the name of a directory. In this
-- case the index and data files are written to that directory. The
-- tape format option is irrelevant in this case.
```

C-6

```

procedure Restore (Objects : String := "?";
                  Use_Prefix : String := "+";
                  For_Prefix : String := "+";
                  Options : String := "R1000";
                  Device : String := "MACHINE.DEVICES.TAPE_0";
                  Response : String := "<PROFILE>");

-- Restore an object or a set of objects from an Archive Tape.

-- If the archive is on a tape then the tape format option given to
-- Restore should be the same as that given during the save. If the
-- archive is in a directory then the device parameter on the restore
-- should be set to that directory.

-- The Objects parameter may be any wildcard pattern specifying the
-- objects to be restored.

-- For example:
--   USERS.HJL.CLI.TEST
--   [USERS.HJL.TESTS:@, USERS.HJL.LOGS.ABC]

-- The pattern in the Objects parameter is compared against the full
-- names of the saved objects. The objects whose names match the Objects
-- parameter specification are restored. If the name denotes an Ada
-- unit all of its parts are restored from the tape. If the name denotes
-- a world or directory all of its subcomponents are restored.

-- The Use_Prefix and For_Prefix parameters provide a simple means for
-- changing the names of the archived objects when they are restored.

-- If the Use_Prefix is the special default value, "+", the For_Prefix
-- is ignored and the objects are restored using the names they had when
-- they were saved.

-- If the Use_Prefix is not "+", it must specify the name of an object
-- into which the archived objects can be restored. The name for a
-- restored object is derived from the name of the archived object by
-- replacing the shortest portion of the name matched by the For_Prefix
-- with the value of the Use_Prefix. If the For_Prefix is "+", the
-- archived objects are restored using the Default_Prefix stored with
-- the archived data.

-- For example:
--   Restore (Objects => "A.B.C.D.E",
--            Use_Prefix => "X.Y",
--            For_Prefix => "A.B.C");

-- will restore to X.Y.D.E.

-- If the name of the archived object does not have the For_Prefix as a
-- prefix, it is restored under its original name.

-- The For_Prefix may contain wildcard characters (#, @, ?) and the
-- Use_Prefix parameter may contain substitution characters (@ or #
-- only). (Not implemented in DO)

-- For example:

```

```

-- Restore (Objects => "[A.B.TEST1, ID.E.F.TEST2]"
-- For_Prefix => "?@"
-- Use_Prefix => "C.D.@");

-- will restore to C.D.TEST1 and C.D.TEST2

-- If the object named by the prefix of the target name of an object
-- being restored doesn't exist, that object will be created as a set of
-- nested worlds. So, for example, if the For_Prefix is A.B and the unit
-- being restored is then A.B.X.Y.Z and ...X.Y hasn't been saved on
-- the tape then A, A.B, A.B.X, A.B.X.Y will be created as worlds.

-- The following options are allowed in the Options parameter:

-- FORMAT and LABEL: options as in the save option.

-- COMPATIBILITY_DATABASE, (CDB) [=Subsystems>]
-- Specifies that the Compatibility Databases for just the named
-- subsystems are to be restored.

-- NONRECURSIVE
-- prevents subcomponents of libraries and Ada units from being
-- implicitly restored. for example:

-- Restore
-- (Objects => "[USERS.HJL, USERS.HJL.CLI, IUSERS.HJL.CLI.@]",
-- Options => "R1000 NONRECURSIVE");

-- will restore only the named objects and not their substructure.

-- OVERWRITE = ALL_OBJECTS | NEW_OBJECTS | UPDATED_OBJECTS | CHANGED_OBJ
-- ALL_OBJECTS
-- All specified objects are restored. This is the default.

-- NEW_OBJECTS
-- Only specified objects that don't already exist on the target
-- machine are restored.

-- UPDATED_OBJECTS
-- Only specified objects that already exists on the target are
-- restored, but only if the update time of the archived object
-- is greater than the update time on the target object.

-- CHANGED_OBJECTS
-- Restore both new and updated Objects.

-- PROMOTE
-- After they are restored, any Ada units will be promoted to the
-- state they were in when they were archived.

-- REPLACE
-- Given an object that is being restored that already exists
-- on the target, this option will cause the restore operation
-- (1) to unfreeze the target object if it is frozen.
-- (2) If the target object is an installed or coded Ada unit
-- with clients, it is demoted to source using Compilation.
-- Demote with the "<ALL_WORLDS>" parameter.

```

```

-- (3) if the parent library into which an object is being
-- restored is frozen, the parent will be unfrozen to restore
-- the object then refrozen.
--
-- OBJECT_ACL=<acl_value>
-- WORLD_ACL=<acl_value>
-- DEFAULT_ACL=<acl_value>
-- Specifies the Access Control List for restored objects
-- (OBJECT_ACL) and worlds (WORLD_ACL) and the default ACL for
-- restored worlds (DEFAULT_ACL). The value is either an ACL
-- specification or the special values INHERIT or ARCHIVED.
-- ARCHIVED means to use the ACL archived with the object and is
-- the default for all three ACL options. INHERIT means to use the
-- standard inheritance rules for new versions of objects.
--
-- BECOME_OWNER
-- Modify the ACL of all restored objects such that the restorer
-- becomes the owner of the restored object.
--
-----
-- procedure List (Objects : String := "?";
-- Options : String := "R1000";
-- Device : String := "MACHINE.DEVICES.TAPE_0";
-- Response : String := "<PROFILE>");
--
-- Produce a listing of the names of the objects on an Archive tape.
--
-- The Objects parameter specifies the objects to be listed. Wildcards
-- are permitted, so if Objects = "?", the default, then all Objects are
-- listed.
--
-- The Options parameters are:
--
-- FORMAT and LABEL
-- as in the Save options.
--
-----
-- procedure Copy (Objects : String := "<IMAGE>";
-- Use_Prefix : String := "+";
-- For_Prefix : String := "+";
-- Options : String := "";
-- Response : String := "<PROFILE>");
--
-- Copy objects from one location to another, including between
-- machines on the same network.
--
-- The Objects parameter specifies where the objects are to be gotten
-- from as in an Archive.Save. The Use_Prefix/For_Prefix parameters
-- specify where the objects are to go as in Archive.Restore.
--
-- Each name consists of an (optional) machine name followed directly
-- by a Objects parameter. A machine name has the form Iname.
-- the Objects part of the source name is like that given to the save
-- operation.
--
-- The Use_Prefix and the For_Prefix function as in the Restore command.
--
-- If the Use_Prefix parameter is "+" or just a machine name, then the

```

```

-- source Objects are moved to the same place on the destination machine
-- as specified by the source. The For_Prefix parameter is ignored.
--
-- If neither Objects nor Use_Prefix have a machine name then the
-- objects are copied from the source to the Use_Prefix on the
-- current machine.
--
-- The Options parameter has the following options.
--
-- AFTER=<time_expression>
-- as in the save operation.
--
-- COMPATIBILITY_DATABASE, CDB
-- NONRECURSIVE
-- as in the save operation.
--
-- PROMOTE, REPLACE,
-- BECOME_OWNER,
-- OBJECT_ACL, WORLD_ACL, DEFAULT_ACL
-- as in the restore operation.
--
-- Examples of calls:
--
-- Copy (Objects => "IUSERS.HJL.CLI",
-- Use_Prefix => "IIM1");
--
-- Will copy the CLI directory in IUSERS.HJL on the
-- current machine to machine M1 IUSERS.HJL.CLI.
--
-- Copy (Objects => "IIM2IUSERS.JMK.CLI");
--
-- Will copy IUSERS.JMK.CLI on M2 to IUSERS.JMK.CLI on the
-- current machine.
--
-- Copy (Objects => "IIM3IUSERS.HJL.CLI.CMD",
-- Use_Prefix => "IUSERS.JMK",
-- For_Prefix => "IUSERS.HJL.CLI");
--
-- Will copy the file IUSERS.HJL.CLI.CMD on M3 to
-- IUSERS.JMK.CMD on the current machine.
--
-- note when repositioning Objects it is necessary to give a
-- for_prefix which is a prefix of the Objects part of the
-- source parameter.
--
-- Copy (Objects => "IIM1IUSERS.HJL.ILFORD",
-- Use_Prefix => "IIM2IAGEA",
-- For_Prefix => "IUSERS.HJL");
--
-- Will copy IUSERS.HJL.ILFORD from machine M1 to
-- machine M2 IAGEA.ILFORD
--
-- Copy (Objects => "IUSERS.HJL.CLI",
-- Use_Prefix => "IIM1",
-- Options => "REPLACE AFTER=12/25/85");
--
-- Will copy those files which have changed since 12/25/85 in
-- IUSERS.HJL.CLI on the current machine to machine M1 IUSERS.HJL.CLI
-- Any existing files with the same names will be overwritten.

```



```

-----
procedure Server;
-- start the archive server;

procedure Status (For_Job : Machine.Job_Id);
-- Prints information about the status of the Archive job specified.
-- Can be the job number of an Archive Server or of a job running
-- Archive.Copy, Archive.Restore, or Archive.Save.

pragma Subsystem (Archive);
pragma Module_Name (4, 3546);
end Archive;

```

```

with Compilation;
with System_Uutilities;

package Cmvc is

-- All CMCV commands raise Profile.Error if any error is detected
-- and Profile.Propagate or Profile.Raise_Error is true
-----

-- Some of the following reservation commands take the name of an object
-- that appears in more than one view. The naming expression
-- imumble.subsystem.[view1, view2, view3].units.object
-- is useful for such times.

procedure Check_Out (What_Object : String := "<CURSOR>";
Comments : String := "";
Allow_Demotion : Boolean := False;
Allow_implicit_Accept_Changes : Boolean := True;
Expected_Check_In_Time : String := "<TOMORROW>";
Work_Order : String := "<DEFAULT>";
Response : String := "<PROFILE>");

-- Check out reserves one or more objects (specified by What_Object) so
-- that they may be modified in only one view. All of the
-- objects specified must belong to the same working view.
-- An object must be 'controlled' to be reserved (see Make_Controlled),
-- a warning is issued for objects that are not controlled.

-- The reservation spans all of the views that share the
-- same reservation token for the element.

-- This command implicitly accepts changes in the checked out object,
-- updating the value of the object to correspond to the most
-- recent generation of that element/reservation token pair.

-- The Comments field is stored with the notes for the object.
-- If What_Object is a set, the comment is stored with all of them.

-- Expected_Check_In accepts any string that Time_Uutilities.Value
-- will accept.

procedure Check_In (What_Object : String := "<CURSOR>";
Comments : String := "";
Work_Order : String := "<DEFAULT>";
Response : String := "<PROFILE>");

-- Release the reservation on the object. What_Object may
-- specify a set of objects. This command only applies to
-- the controlled objects in the set and will note any
-- objects that are not controlled.

-- Comments are treated as in Check_Out

procedure Accept_Changes (Destination : String := "<CURSOR>";
Source : String := "<LATEST>";
Allow_Demotion : Boolean := False;

```

```

-- the libraries in such a way the relocation operates most effectively,
-- preventing compilation in many cases when changes move between views.

procedure Abandon_Reservation (What_Object : String := "<SELECTION>";
  Comments : String := "";
  Work_Order : String := "<DEFAULT>";
  Response : String := "<PROFILE>");

-- Forget about a check_out of some object, or set of objects.
-- This reverts the objects back to last checked in version.
-- This operation is an "undo" for Check_Out, except that it
-- does not undo the Implicit Accept_Changes that goes with
-- a Check_Out.

procedure Revert (What_Object : String := "<SELECTION>";
  To_Generation : Integer := -1;
  Make_Latest_Generation : Boolean := False;
  Comments : String := "";
  Work_Order : String := "<DEFAULT>";
  Response : String := "<PROFILE>");

-- Replace the contents of the specified object with the contents
-- of the specified generation. The operation is equivalent to an
-- Accept_Changes from a configuration containing the specified
-- generation.

-- If Make_Latest_Generation is true, then the operation is equivalent to
-- a Check_Out, a copy of the specified generation into the object, and
-- a Check_In.

-- Generation of -n means n generations back; thus -1 => the previous
-- generation.
-----

-- The following commands allow the creation and interrogation of
-- a note scratchpad for each element. Descriptive information
-- regarding what is being changed, why, or whatever, can be put
-- into the scratchpad.

procedure Get_Notes (To_File : String := "<WINDOW>";
  What_Object : String := "<CURSOR>";
  Response : String := "<PROFILE>");

-- Copy the notes from the object. If To_File is the default, then
-- a new I/O window is created and the notes are copied into this window.
-- The first line of this window is the name of the object, which is
-- used by Put_ and Append_Notes to put the notes back. The notes
-- displayed are those that go with the generation of the object pointed
-- at. See Cmcv_History for ways of getting notes and other information
-- on a range of generations

-- The next three commands require the object in question to be
-- checked out.

procedure Put_Notes (From_File : String := "<WINDOW>";
  What_Object : String := "<CURSOR>";
  Response : String := "<PROFILE>");

```

```

Comments : String := "";
Work_Order : String := "<DEFAULT>";
Response : String := "<PROFILE>");

-- This operation updates the Destination to reflect changes
-- (objects that have been checked in) specified by Source.

-- The Destination is either a view or a set of objects (all in
-- one view). Specifying the view is equivalent to specifying
-- all the objects in the view. Uncontrolled objects in the
-- destination are ingored except that a note is issued.

-- The Source is either "<LATEST>", a view, a configuration,
-- or a set of objects all in one view.

-- If the Source is "<LATEST>", the destination objects
-- will be updated to the most recently checked in version.
-- If the most recent generation of a source object is currently
-- checked out, the previous generation is used and a warning
-- is issued.

-- If the Source is a view and the Destination is a view, this command
-- is basically "Make the Destination view look exactly like the
-- Source view". Every controlled object in the source is copied
-- to the destination and the configuration in the destination
-- is updated. This includes new objects which did not previously
-- exist in the destination. If the destination has a more recent
-- version than the source, the destination will not be updated and
-- a warning is issued. In particular, if objects are checked out in
-- the destination, they will not be changed.

-- If objects are checked out in the source this operation
-- will use the previously checked in version of the object and
-- a warning will be issued.

-- If the Source is a view and the Destination is a set of objects,
-- the destination objects are updated to the corresponding objects
-- in the source view, as above.

-- If the source is a configuration it is identical to having the
-- source be a view except that the configuration specifies the
-- versions to use and they may be older (less up to date) than
-- the ones in the destination. Thus if the source is a configuration
-- then destination objects may "go backwards", while this will not
-- happen if the source is a view.

-- If the source is a set of objects and the destination is a view,
-- the corresponding objects in the destination view are updated
-- to the source objects.

-- A common way of using Accept_Changes is to use the default parameters
-- during normal development to accept changes made in other subpaths.
-- Then periodically an integration view (in the path) is updated by
-- first accepting all relevant subpaths into the integration view
-- (accept_changes (destination => integration_view, source =>
  active_subpath_working_view)).
-- Then this integration view is compiled (and tested). The subpaths are
-- then re-synchronized by accepting the integration view (source =>
  integration_view, destination => destination_subpath_working_view).

-- In addition to synchronizing the source, this protocol updates

```

```

-- Replace the notes for the specified object. If the I/O window
-- was created by Get_Notes, the window (first line) contains the name
-- of the object to write back into, and What_Object is ignored.

procedure Append_Notes (Note : String := "<WINDOW>";
                       What_Object : String := "<CURSOR>";
                       Response : String := "<PROFILE>");

-- Append the specified text to the notes. If Note is <IMAGE_TEXT>,
-- the associated window must have been created by Get_Notes or
-- Create_Empty_Note_Window; in this case What_Object is ignored.
-- If Note is a string, then that string is appended to the object
-- selected by What_Object. If the content of Note is prepended with a
-- ', Note is interpreted as a text file name, and the content of
-- that file is appended to the selected object.

procedure Create_Empty_Note_Window (What_Object : String := "<CURSOR>";
                                   Response : String := "<PROFILE>");

-- Create an empty window (with no underlying directory object)
-- to be used for constructing notes for the specified object.
-- Typically, Append_Notes is used to actually add the text
-- to the object's notes.

-----

procedure Make_Controlled
(What_Object : String := "<CURSOR>";
 Reservation_Token_Name : String := "<AUTO_GENERATE>";
 Join_With_View : String := "<NONE>";
 Comments : String := "";
 Work_Order : String := "<DEFAULT>";
 Response : String := "<PROFILE>");

-- Make the object or objects specified by What_Object be subject to
-- reservation. The objects must be in a working view and not
-- already controlled. All objects must be in the same subsystem.
-- If Join_With_View is specified, the objects are joined with the
-- object in that view, using the reservation token specified by that view.
-- If no view is specified, the reservation token name is used if provided,
-- else the development path name of the view containing the object is
-- used as the reservation token name.

procedure Make_Uncontrolled (What_Object : String := "<CURSOR>";
                             Comments : String := "";
                             Work_Order : String := "<DEFAULT>";
                             Response : String := "<PROFILE>");

-- Make an object or objects uncontrolled.
-- This means the objects are no longer subject to reservation
-- (in the enclosing view).

procedure Sever (What_Objects : String := "<SELECTION>";
                New_Reservation_Token_Name : String := "<AUTO_GENERATE>";
                Comments : String := "";
                Work_Order : String := "<DEFAULT>";
                Response : String := "<PROFILE>");

```

```

-- Make the object(s) in the given working view(s) have a separate
-- reservation. This command severs the relationship between views
-- for objects. When done, the views specified in this command will
-- have their own reservation to share. All other views (not
-- specified) will share a different reservation.
-- A specific reservation token name can be provided, if desired.

procedure Join (What_Object : String := "<SELECTION>";
              To_Which_View : String := ">>VIEW_NAME<<";
              Comments : String := "";
              Work_Order : String := "<DEFAULT>";
              Response : String := "<PROFILE>");

-- Make object in two or more working views share a reservation. The
-- objects in the views must be identical (textually) and controlled
-- for this command to succeed.

-----

procedure Merge_Changes (Destination_Object : String := "<SELECTION>";
                        Source_View : String := ">>VIEW_NAME<<";
                        Report_File : String := "";
                        Fail_If_Conflicts_Found : Boolean := False;
                        Comments : String := "";
                        Work_Order : String := "<DEFAULT>";
                        Response : String := "<PROFILE>");

-- Merge two versions of the same object together, leaving the result
-- in destination object. In order for this command to succeed, the
-- Source_View and the view containing the Destination_Object must
-- have been copied from some common view sometime in the past, and
-- the configuration for that view must still exist.
-- Destination_Object must refer to the last generation; all changes must
-- have been accepted.

-- The command writes a report showing what it did, as well as changing
-- the destination object. If the report_file name is "", the report
-- is written to Get_Simple_Name (Destination_Object) & "Merging_Report".
-- Conflicts are defined to be regions of change in the source and
-- destination that directly overlap, ie the same line(s) have been
-- changed in both objects. If Fail_If_Conflicts_Found is true,
-- no updating is done, but the report file is left.
-- If it is desired to rejoin the two objects after the merge, then
-- check out the Merge source object, copy the Merge Destination_Object
-- into the source, then Join the objects.

-----

function Imported_Views (Of_View : String := "<CURSOR>";
                        Include_Import_Closure : Boolean := False;
                        Include_Importer : Boolean := False;
                        Response : String := "<WARN>") return String;

-- return a string suitable for name resolution that names the union of
-- all of the imports specified by the view(s) Of_View. These views
-- are in no particular order.

```

-- It controlling of spec views is desired, use Make\_Controlled after  
-- creating the views. But be forewarned that checking out a spec  
-- where an implicit accept is required will probably obsolete all  
-- of the spec's clients.

-----  
IMPORTS  
-----  
CVC supports selective importing of units when views are imported.  
This is accomplished using Imports\_Restrictions and  
Exports\_Restrictions.

Exports\_Restrictions are subsets of exported Ada units controlled  
by the exporting view (spec view). The subset is determined by the  
contents of a text file in the Exports directory of the view. This  
file contains Naming expressions which, when resolved against the  
Units directory, produce a list of objects that are exported by  
that subset.

Imports\_Restrictions are further restrictions on what Ada units are  
to be imported. The restriction specifies which export restriction  
to use (if any), a list of Ada units (using simple names) to  
exclude, and a list of units to rename. A restriction is a text  
file, in the Imports directory, with the same name as the subsystem  
containing the view being imported. Each line of the file  
specifies one thing. The form of the lines are:

EXPORT RESTRICTION=>restriction\_name  
Specify the name of the export restriction. No blanks are  
allowed. If more than one restriction is specified, the  
union of all of the restrictions is used.  
Object\_Name Link\_Name  
Import Object\_Name but make a link with Link\_Name (a rename)  
"Object\_Name  
Dont import Object\_Name  
Object\_name  
Import Object\_Name and use Object\_Name for the link name  
Import all Objects, except those removed above  
In all cases, the names provided above are simple names, ie no '.'s  
in them.

-----  
SELECTING VIEWS  
-----  
In the following commands, wherever a view is called for, a naming set  
can be used. A text file containing the names of configurations  
or views can also be used. However, you must use the leading '  
convention supported by Naming. Also, configuration names can be  
used in place of views anywhere, assuming that the view represented  
by the configuration still exists.

-----  
SPEC VIEWS  
-----  
Spec views in CVC are by default uncontrolled. The reason for this  
is to allow free changing of specs in the load views, accepting the  
changes back and forth, then incrementally making the changes in the  
spec views.

procedure Release (From\_Working\_View : String := "<CURSOR>";  
Release\_Name : String := "<AUTO\_GENERATE>";  
Level : Natural := 0;  
Views\_To\_Import : String := "<INHERIT\_IMPORTS>";  
Create\_Configuration\_Only : Boolean := False;  
Compile\_The\_View : Boolean := True;  
Goal : Compilation\_Unit\_State := Compilation.Coded;  
Comments : String := "";  
Work\_Order : String := "<DEFAULT>";  
Volume : Natural := 0;  
Response : String := "<PROFILE>");

-- Create a new release view in the subsystem. If Release\_Name is  
-- "<AUTO\_GENERATE>", the view will have the same name prefix as the  
-- working view, with \_n\_ appended as appropriate given the level.  
-- Otherwise Release\_Name must be the simple name of the new release.

-- Since the new view is a release, it is frozen. If From\_Working\_View  
-- names multiple views, each named working view is released as  
-- above, and the imports are adjusted so that the new releases  
-- reference each other as appropriate instead of the working views.  
-- Views\_To\_Import specifies, perhaps by indirection through an activity,  
-- a set of views to be used as imports by the new view(s). This allows  
-- changing imports during a release. Imports already adjusted during  
-- the releasing of working views will be left alone, otherwise  
-- subsystems currently imported will be reimported. In other words,  
-- if this were an import command, Only\_Change\_Imports would be true.  
-- If Compile\_The\_View is true, the compiler is run before the views  
-- are frozen, trying to promote the units to the indicated Goal.  
-- The views are frozen even if compilation fails.

-- This command creates a configuration object named  
-- SUBSYSTEM.state.configurations.release\_name. It also creates an  
-- import description file in the same place, named release\_name &  
-- "imports". This import description file lists the configuration  
-- objects for all views that are imported. It is maintained by  
-- all commands that modify or adjust the imports. These two objects  
-- are used to reconstruct views from configurations.

-- A controlled text object (state.release\_history) is used by this  
-- command. Release enters the comments supplied with the command  
-- into the notes for this object. Feel free to check out and modify  
-- this object to further describe what is going on. This object is joined  
-- across all of the releases and the working view of a subpath.  
-- Furthermore, the object is checked out and in by the release command  
-- in order to mark the time of the release.

```

procedure Copy (From_View : String := "<CURSOR>";
New_Working_View : String := ">>SUB/PATH NAME<<";
View_To_Modify : String := "",
View_To_Import : String := "<INHERIT_IMPORTS>";
Only_Change_Imports : Boolean := True;
Join_Views : Boolean := True;
Reservation-Token_Name : String := "",
Construct_Subpath_Name : Boolean := False;
Create_Spec_View : Boolean := False;
Level_For_Spec_View : Natural := 0;
Model : String := "<INHERIT_MODEL>";
Remake_Demoted_Units : Boolean := True;
Goal : Compilation.Unit_State := Compilation.Coded;
Comments : String := "";
Work_Order : String := "<DEFAULT>";
Volume : Natural := 0;
Response : String := "<PROFILE>");

```

```

-- Create a new working view. Working views are named Mumble_Working,
-- where mumble is supplied as New_Working_View. If Join_Views is
-- true, the two views share reservations of the all of the controlled
-- objects in the two views. If false, reservations aren't shared
-- across the views for any objects. If From_View names multiple views, a
-- copy is made for each of those views and, if the originals
-- import each other (computed using the subsystem, not the view),
-- the copies will (try) to import the new views of those subsystems.
--
-- If Join_Views is false, new reservation tokens are created for all
-- of the controlled objects. The default is to use the name supplied
-- as the >>SUBPATH_NAME<<.
--
-- View_To_Import supplies a set of views to be processed according to
-- the value of Only_Change_Imports. If Only_Change_Imports is true,
-- a copied view always inherits the source view's imports. After the
-- copy, the imports specified by View_To_Import are applied against the
-- new view, replacing any inherited import if needed.
--
-- If Only_Change_Imports is false, then either the imports are inherited
-- from the source, or the complete set of imports specified by
-- by View_To_Import is imported into the copy.
--
-- View_To_Modify specifies the set of working views that are to have
-- their imports changed to refer to the new copy(s). The
-- View_To_Modify views are also changed to refer to the views specified
-- by View_To_Import. For this import operation, Only_Change_Imports
-- is forced to true.
--
-- Construct_Subpath_Name cause Copy to construct the target view name
-- by appending New_Working_View to the prefix of the source view name
-- up to the first '_' (See paths and subpaths below).
--
-- Remake demoted units, if true, indicates that ada units that were
-- demoted during the copy process are to be recompiled. They are
-- compiled to the level indicated by Goal. Units are not compiled
-- to a state higher than they were in the source.
--
-- Goal further indicates the desired state of all of the units after
-- copy. No unit will be in a state higher than specified by goal, but
-- might be in a lower state. For example, a source unit that is copied
-- will remain source, regardless of Goal, but a Coded unit will be

```

```

-- demoted if Goal is installed or less.
--
-- The order of the copy and import operations is:
--
-- 1. Create the new view.
-- 2. If Inherit_Imports, bring along the old imports
-- 3. Import the new views into the new views, forcing
-- Only_Change_Imports => True
-- 4. If not Inherit_Imports, import the specified views
-- into the new views.
-- 5. Import the new views + View_To_Import into Views_To_Modify,
-- forcing Only_Change_Imports => true
--
-- Spec views are created by copying the units if the source is a load
-- view, otherwise using Relocation. Spec views are created with all
-- objects uncontrolled. If level_for_spec_view = natural'last, the
-- spec view is given the name supplied as new_working_view, otherwise
-- a name is generated as 'New_Working_View & Release_Numbers & "_spec"'.
--
-- It is recognized that this is a complicated command. Using the
-- procedures below (which are effectively renames) might make more
-- sense if the methodology in use permits it (Path, Subpath, etc).

```

-----  
PATHS AND SUBPATHS  
-----

```

-- The following procedures support the notion of paths and subpaths.
-- A Path is a logically connected series of releases in which all
-- controlled objects are joined together. In other words, there is
-- no branching within a path. A Subpath is an extension of the
-- path, allowing multiple developers to make changes and test
-- without getting in each others way. However, controlled objects
-- in the subpaths are joined with the path; people in two subpaths
-- cannot independently change the same object. In addition, a path
-- and its subpaths share the same model, which means they share
-- the same Target_Key and Initial links.
--
-- In Delta, paths and subpaths are identified by string name conventions.
-- The name of the path is the view name up to the first '_'. The
-- subpath extension is the name from this '_' to the '_Working'. Thus
-- Rev9_Cbh_Working has a path name of Rev9 and subpath extension of
-- Cbh.
--
-- Multiple paths are used when multiple targets are involved, or when
-- objects are to be changed independently. For example, assume that
-- a version of a product has been shipped, and is in maintenance, and
-- that development is progressing on a new version. It is likely that
-- the old and new versions would be separate paths, since the objects
-- would have to be independently changed (these paths would not be
-- 'joined').
--
-- In the multiple target case, the paths might be created joined.
-- Using the above scenario, assume that the release that has been shipped
-- works on two targets, but most or all of the code is target
-- independent. Then the two paths, one for each target, would be
-- created joined together, then have the objects that are not common
-- 'sever'ed.

```

```

procedure Make_Path
  From_Path : String := "<CURSOR>";
  New_Path_Name : String := ">>PATH NAME<<";
  View_To_Modify : String := "";
  View_To_Import : String := "<INHERIT_IMPORTS>";
  Only_Change_Imports : Boolean := True;
  Model : String := "<INHERIT_MODEL>";
  Join_Paths : Boolean := True;
  Remake_Demoted_Units : Boolean := True;
  Goal : Compilation.Unit_State := Compilation.Coded;
  Comments : String := "";
  Work_Order : String := "<DEFAULT>";
  Volume : Natural := 0;
  Response : String := "<PROFILE>";

procedure Make_Subpath (From_Path : String := "<CURSOR>";
  New_Subpath_Extension : String := ">>SUBPATH<<";
  View_To_Modify : String := "";
  View_To_Import : String := "<INHERIT_IMPORTS>";
  Only_Change_Imports : Boolean := True;
  Remake_Demoted_Units : Boolean := True;
  Goal : Compilation.Unit_State := Compilation.Coded;
  Comments : String := "";
  Work_Order : String := "<DEFAULT>";
  Volume : Natural := 0;
  Response : String := "<PROFILE>");

-- The Subpath_Extension is appended to the path name of the source
-- view (From_Path). From_Path can actually name the path or any
-- subpath of the path. The '_' between the path and subpath extension
-- is automatically provided.

procedure Make_Spec_View
  (From_Path : String := "<CURSOR>";
  Spec_View_Prefix : String := ">>PREFIX<<";
  Level : Natural := 0;
  View_To_Modify : String := "";
  View_To_Import : String := "<INHERIT_IMPORTS>";
  Only_Change_Imports : Boolean := True;
  Remake_Demoted_Units : Boolean := True;
  Goal : Compilation.Unit_State := Compilation.Coded;
  Comments : String := "";
  Work_Order : String := "<DEFAULT>";
  Volume : Natural := 0;
  Response : String := "<PROFILE>");

-- Make a spec view for a path. Spec_View_Prefix is the string that
-- replaces the path and subpath name. For example, if creating a
-- spec view from a subpath named rev9_cbh_working, with
-- Spec_View_Prefix => Env9, the result will be Env9_n.Spec, assuming
-- level => 0 and two levels are specified by the model. N is a
-- number automatically generated from the current release number for
-- the path/subpath. If level = natural'last, the name supplied as
-- Spec_View_Prefix is used for the name of the view, with no suffixes

```

```

Into_View : String := "<CURSOR>";
Only_Change_Imports : Boolean := False;
Import_Closure : Boolean := False;
Remake_Demoted_Units : Boolean := True;
Goal : Compilation.Unit_State := Compilation.Coded;
Comments : String := "";
Work_Order : String := "<DEFAULT>";
Response : String := "<PROFILE>";

-- Imports spec or combined views as appropriate into the specified
-- view(s). The import specification can be a set of view names,
-- in which case all views are imported, unless only_change_imports is
-- true. In this case only subsystems that were imported sometime in
-- the past are reimported. All others are ignored.

-- The import description file mentioned in the release command is
-- brought up to date by this command.

-- If View_To_Import is "", then the imports of Into_View are refreshed.
-- This means the various imported views are examined, and any new
-- Ada specs are imported in to the current view.

-- It is useful to invoke Import with Views_To_Import = Into_View and
-- Only_Change_Imports is true. This will cause a set of views to be
-- changed to import each other.

procedure Remove_Import (View : String := ">>VIEW NAME<<";
  From_View : String := "<CURSOR>";
  Comments : String := "";
  Work_Order : String := "<DEFAULT>";
  Response : String := "<PROFILE>");

-- remove references to a previously imported view.

procedure Remove_Unused_Imports (From_View : String := "<CURSOR>";
  Comments : String := "";
  Work_Order : String := "<DEFAULT>";
  Response : String := "<PROFILE>");

-- Search through all of the Ada units in the view and examine the
-- withs. If no units in some imported view are referenced, remove
-- that import.

-- This command generates warnings if units in spec or combined
-- views are referenced, but the view isn't imported. Errors are
-- generated if units in load views are referenced.

procedure Replace_Model (New_Model : String := ">>NEW MODEL NAME<<";
  In_View : String := "<CURSOR>";
  Comments : String := "";
  Work_Order : String := "<DEFAULT>";
  Response : String := "<PROFILE>");

-- Replace the model with the new one. All units must be source.
-- This command gets the switch file from the new model (if one
-- was provided), readjusts the maximum levels (which affects future
-- releases), and rebuilds the links.

```

```

procedure Import (View_To_Import : String := "<REGION>";

```

```

Work_Order : String := "<DEFAULT>";
Response : String := "<PROFILE>";

-- Destroy a subsystem. There must be no views in the subsystem

-----
procedure Build (Configuration : String := ">>CONFIGURATION NAME<<";
View_To_Import : String := "";
Model : String := "R1000";
Goal : Compilation.Unit_State := Compilation.Installed;
Limit : String := "<WORLDS>";
Comments : String := "";
Work_Order : String := "<DEFAULT>";
Volume : Natural := 0;
Response : String := "<PROFILE>");

-- Rebuild a view from history. If Configuration_Object_Name refers to
-- a text file, that file is assumed to contain a list of configuration
-- object names to be built.

-- If View_To_Import = "", and if a text file exists with the name "same
-- as configuration_object" & "_imports", that text file is opened
-- after the views are built and imports are constructed from the views
-- or configuration objects named in that file. Please note that copy,
-- initial, import, and remove_import will create and maintain such a
-- text file, so it is probably there.

-----
HISTORY COMMANDS
-----

-- The following commands display history information, in various
-- formats, of Cwvc controlled objects

procedure Show_History (For_Objects : String := "<CURSOR>";
Display_Change_Regions : Boolean := True;
Starting_Generation : String := "<CURSOR>";
Ending_Generation : String := "";
Response : String := "<PROFILE>");

-- Display the history for the specified objects. If a view is
-- specified, all of the controlled objects in that view are displayed.
-- This history includes notes, checked_out and _in information, and
-- optionally the actual changes

-- If display_change_regions is true, the differences between a
-- generation and the previous one (n-1, n) are displayed. The display
-- is in the form of regions where changes occurred similar to that
-- produced by File_Uilities.Difference(Compressed_Output=>True)

-- The first generation to display is determined by looking up
-- the object in the view(s) specified by Starting_Generation. If
-- Starting_Generation = "", the display starts at generation 1.

-- The last generation to display is determined by Ending_Generation.
-- If E..G.. is "", the last displayed is the latest one. If E..G..
-- is the name of a view, the generation specified by that view is
-- used as the last.

```

```

type Subsystem_Type_Enum is (Spec_Load, Combined, System);

procedure Initial (Subsystem : String := ">>SUBSYSTEM NAME<<";
Working_View_Base_Name : String := "Rev1";
Subsystem_Type : Subsystem_Type_Enum := Cwvc.Spec_Load;
View_To_Import : String := "";
Model : String := "R1000";
Comments : String := "";
Work_Order : String := "<DEFAULT>";
Volume : Natural := 0;
Response : String := "<PROFILE>");

-- Build a new subsystem of the specified type. Also create a working
-- view and import as specified. This command can be used to create
-- an empty view in an existing subsystem.

-----
procedure Information (For_View : String := "<CURSOR>";
Show_Model : Boolean := True;
Show_Whether_Frozen : Boolean := True;
Show_View_Kind : Boolean := True;
Show_Creation_Time : Boolean := True;
Show_Imports : Boolean := True;
Show_Referencers : Boolean := True;
Show_Unit_Summary : Boolean := True;
Show_Controlled_Objects : Boolean := False;
Show_Last_Release_Numbers : Boolean := False;
Show_Path_Name : Boolean := False;
Show_Subpath_Name : Boolean := False;
Show_Switches : Boolean := False;
Show_Exported_Units : Boolean := False;
Response : String := "<PROFILE>");

-- Show various things about a view. Please see Cwvc_History for
-- ways of extracting other information about the controlled objects
-- in the view.

-----
procedure Destroy_View (What_View : String := "<SELECTION>";
Demote_Clients : Boolean := False;
Destroy_Configuration_Also : Boolean := False;
Comments : String := "";
Work_Order : String := "<DEFAULT>";
Response : String := "<PROFILE>");

-- Destroy a view. If Demote_Clients is false, the view can have no
-- referencing views (clients); if it does, the destroy fails. If
-- Demote_Clients is true, the view is "remove_import"ed from those
-- clients (which might cause lots of obsolescence), then the view is
-- destroyed. The configuration object for the view is left behind
-- in its normal place (see Release, above) so the view can be
-- reconstructed using "Build"

-----
procedure Destroy_Subsystem (What_Subsystem : String := "<SELECTION>";
Comments : String := "");

```

```

procedure Show_History_By_Generation
(For_Objects : String := "<CURSOR>";
 Display_Change_Regions : Boolean := True;
 Starting_Generation : Natural := 1;
 Ending_Generation : Natural := Natural'Last;
 Response : String := "<PROFILE>");
-- In this case, All_Units means all of the units in the current
-- view. Naming a view means all units in that view.
procedure Show_All_Uncontrolled (Object_Or_View : String := "<CURSOR>";
 Response : String := "<PROFILE>");
-- List objects that are not controlled. Produces output only if an
-- object listed (or one in the units directory if a view is supplied)
-- is not under CMCV control
procedure Show_Image_Of_Generation (Object : String := "<CURSOR>";
 Generation : Integer := -1;
 Output_Goes_To : String := "<WINDOW>";
 Response : String := "<PROFILE>");
-- Reconstruct an image of some generation of the specified object.
-- The default (-1) indicates back up one generation from that of
-- Object. Negative numbers are relative to the generation of Object,
-- positive numbers are actual generation numbers.
-- The result is written to current output unless a file name is
-- supplied in Output_Goes_To.
-----
-- The following commands produce a report showing objects that
-- meet some criteria. This report shows the following information
-- about each object.
-- Object Name Generation Where Chkd Out By Who Expected Check In
-----
UNITS.FOO 5 of 8 VIEW Yes MTD Apr 7, 1987
-----
-- Object name is the element name (the name from the view down)
-- Generation is a pair. The first number is the generation of
-- the object used to lookup the element. The second number is
-- the highest generation produced.
-- Where is either the view containing a copy of the last generation
-- if the object is not checked out, or the view in which the object
-- is checked out. In the case where the object is not checked out,
-- it is possible that there is no representative object, in which
-- case this field is blank.
-- Chkd Out is 'Checked Out'. If this is yes, 'By Who' and
-- 'Expected Check In' provide more information.
-----
procedure Show (Objects : String := "<CURSOR>";
 Response : String := "<PROFILE>");
-- Produce the information described above for the listed objects.

```

```

-- Also produces a report for each object showing which views
-- contain elements sharing a reservation token with the object.
procedure Show_All_Checked_Out (In_View : String := "<CURSOR>";
 Response : String := "<PROFILE>");
-- Look through all of the controlled objects in the supplied view, and
-- display information about them if they are checked out anywhere
procedure Show_Checked_Out_In_View (In_View : String := "<CURSOR>";
 Response : String := "<PROFILE>");
-- Display information about all of the objects checked out in the
-- view pointed at (or in)
procedure Show_Checked_Out_By_User
(In_View : String := "<CURSOR>";
 Who : String := System.Utilities.User_Name;
 Response : String := "<PROFILE>");
-- Display information about any object in the view that is checked out
-- be the user given. This command will find the object even if it is
-- checked out in some other view, as long as it is controlled in the
-- view referred to.
procedure Show_Out_Of_Date_Objects (In_View : String := "<CURSOR>";
 Response : String := "<PROFILE>");
-- Display information about all objects in the view that are not
-- at the latest revision.
procedure Show_All_Controlled (In_View : String := "<CURSOR>";
 Response : String := "<PROFILE>");
-- Display information about all controlled objects in this view
-----
ARCHIVE COMMANDS
-----
procedure Make_Code_View (From_View : String := "<CURSOR>";
 Code_View_Name : String := "";
 Comments : String := "";
 Work_Order : String := "<DEFAULT>";
 Volume : Natural := 0;
 Response : String := "<PROFILE>");
-- Make a code view with the given name. From_View must only
-- name load and/or combined views. If a load view is provided, no
-- specs are copied; all specs are copied for combined views.
-- This operation fails if any unit isn't coded, or any spec exists
-- for which a body is required and one doesn't exist.
-----
pragma Subsystem (Cmcv);
pragma Module_Name (4, 3704);
end Cmcv;

```



```

package Cmcv_Maintenance is
  procedure Expunge_Database (In_Subsystem : String := "<CURSOR>";
    Response : String := "<PROFILE>");

  -- Free up space in the Database by first finding all configurations
  -- in the database that no longer have objects and destroying them,
  -- then destroying all elements and join sets (with all of their
  -- generations) that are no longer referenced.

  procedure Delete_Unreferenced_Leading_Generations
    (In_Subsystem : String := "<CURSOR>";
    Response : String := "<PROFILE>");

  -- Not yet implemented

  procedure Convert_Old_Subsystems (Which : String := "<SELECTION>";
    Response : String := "<PROFILE>");

  -- Convert all of the views in a subsystem to CMVC subsystems. This
  -- command can convert more than one subsystem per call.

  procedure Check_Consistency (Views : String := "<CURSOR>";
    Response : String := "<PROFILE>");

  -- Verify that all of the views are consistent with the CMVC invariants.
  -- Checks that:
  -- The configurations all exist and are correct.
  -- There are no dangling controlled objects.
  -- The imports are ok, and that all of the imported subsystems
  -- record the reference.
  -- Various other things.

  -- User level commands for manipulating the compatibility database (CDB)
  -- associated with subsystems.

  procedure Display_Cdb (Subsystem : String := "<CURSOR>";
    Show_Units : Boolean := False;
    Response : String := "<PROFILE>");

  -- Displays a summary of the information in the CDB. If "show_units"
  -- is true, then a summary of information for the units currently
  -- known in the subsystem is also displayed.

  procedure Make_Primary (Subsystem : String := "<SELECTION>";
    Moving_Primary : Boolean := False;
    Response : String := "<PROFILE>");

  -- Makes the subsystem into a primary subsystem with its own read/write
  -- CDB. If the subsystem was a primary this operation is a no-op. If
  -- the subsystem is a secondary then a new subsystem_id is assigned.
  -- If "moving_primary" is set to true, then the location of the
  -- primary for this subsystem is being moved and the current subsystem_id
  -- will be used. When moving a primary the user must make sure
  -- that the original primary is either destroyed or converted into
  -- a secondary to prevent corruption of the CDB.

```

```

  procedure Make_Secondary (Subsystem : String := "<SELECTION>";
    Response : String := "<PROFILE>");

  -- Makes the subsystem into a secondary with the same subsystem_id.

  procedure Destroy_Cdb (Subsystem : String := "<SELECTION>";
    Limit : String := "<WORLDS>";
    Effort_Only : Boolean := True;
    Response : String := "<PROFILE>");

  -- Destroys the CDB and all remnants of it in compiled units.
  -- This includes demoting ALL units in the subsystem to source
  -- and deleting all code-only views. If "effort-only" is set
  -- to true, then the effects of the operation are computed
  -- and displayed.

  procedure Update_Cdb (From_Subsystem : String := "<ASSOCIATED_PRIMARY>";
    To_Subsystem : String := "<SELECTION>";
    Response : String := "<PROFILE>");

  -- Moves the CDB from one subsystem to another using the network
  -- if necessary. Both subsystems must have the same subsystem_id.

  procedure Repair_Cdb (Subsystem : String := "<SELECTION>";
    Verify_Only : Boolean := True;
    Delete_Current : Boolean := False;
    Response : String := "<PROFILE>");

  -- Will rebuild the CDB to be consistent with the currently compiled
  -- units in the subsystem. If "verify_only" is true then the CDB
  -- will not be changed, but will be checked for consistency with
  -- the currently compiled units. If "verify_only" is false and
  -- "delete_current" is true then the current CDB will be deleted
  -- and then rebuilt. If the "verify_only" is false and
  -- "delete_current" is false then existing entries in the CDB
  -- will be verified and missing entries will be added.

  procedure Display_Code_View (View : String := "<CURSOR>";
    Verbose_Unit_Info : Boolean := False;
    Show_Map_Info : Boolean := False;
    Response : String := "<PROFILE>");

  pragma Subsystem (Cmcv);
  pragma Module_Name (4, 3707);
end Cmcv_Maintenance;

```

```

package Command is
  procedure Diana_Edit (Name : String := "<IMAGE>");
  procedure Spawn;
  procedure Debug;

  pragma Subsystem (Command);
  pragma Module_Name (4, 2212);

end Command;

```

```

package Common is

  procedure Abandon (Window : String := "<IMAGE>");
  -- Release all locks, and delete the associated window.
  -- This causes the loss of any editing changes.

  procedure Clear_Underlining;
  -- Remove underlining marks left on the image by previous commands.

  procedure Commit;
  -- Make changes to the image permanent

  procedure Complete (Menu : Boolean := True);
  -- Make the current image complete. Provides syntactic and semantic
  -- completion, as possible.
  -- Menu => bring up a menu window for ambiguous references

  procedure Create_Command;
  -- Go to the command window for the current image, creating one if
  -- necessary.

  procedure Definition (Name : String := "<CURSOR>";
    In_Place : Boolean := False;
    Visible : Boolean := True);
  -- Bring up the appropriate image to show the designated object.
  -- Do not make the image modifiable. If a new window is required
  -- In_Place indicates that the current frame should be used. Visible
  -- controls how names that resolve to both a visible part and a body
  -- should be resolved. Visible causes the visible part to be pre-
  -- ferred; not Visible brings up the body if that is possible

  procedure Edit (Name : String := "<IMAGE>";
    In_Place : Boolean := False;
    Visible : Boolean := True);
  -- Bring up the appropriate image to show the designated object.
  -- Attempt to make the image modifiable.
  -- In_Place and Visible are as in Definition.

  procedure Enclosing (In_Place : Boolean := False;
    Library : Boolean := False);
  -- Bring up the image for the object enclosing this one.
  -- In_Place is as in Definition.
  -- Library => the resulting image should be a Library; e.g. for Ada
  -- subunits, go to the enclosing directory rather than parent body.

  procedure Elide (Repeat : Positive := 1);
  -- Reduce the level of detail presented by the number of levels
  -- specified. Attempts to expand beyond maximum level have no effect.
  -- It is not expected that Elide will reorder the presentation.

  procedure Expand (Repeat : Positive := 1);
  -- Increase the level of detail presented by the number of levels
  -- specified. Attempts to expand beyond maximum level have no effect.
  -- It is not expected that Expand will reorder the presentation.

  procedure Explain;
  -- Provide additional information about the indicated object.
  -- The additional information may take the form of more detailed
  -- display or error message explanation. If more detailed infor-

```

```

-- mation is supplied, repeated applications cause the display to
-- cycle through the available presentations. For Ada, provides
-- text of messages associated with underlinings.

procedure Format;
-- Format the current image appropriately for its image type.

procedure Revert;
-- Restore the image to the state of the underlying object.
-- This causes the loss of any editing changes.

procedure Release_Window (Window : String := "<IMAGE>");
-- Make changes to the designated image permanent (if applicable),
-- release all locks, and delete the associated window

procedure Semanticize;
-- Perform semantic checking on the image.

procedure Sort_Image (Format : Integer := 1);
-- Sort the display according to the given format. Format numbering is
-- specific to the object type. It is assumed that if format 1 sorts by
-- increasing values that format -1 will sort by decreasing values of
-- the same key. Clearly not relevant to all object types.

procedure Demote;
-- Bring the image to the next lower state.

procedure Promote;
-- Bring image to the next higher state.

procedure Redo (Repeat : Positive := 1);
-- Inverse of Undo

procedure Undo (Repeat : Positive := 1);
-- restore the contents of the image to the previous consistent state

procedure Insert_File (Name : String := "<REGION>");
-- Insert the contents of the indicated file into the current image

procedure Write_File (Name : String := ">>FILE NAME<<");
-- Write the contents to the named text file

package Object is
  procedure Insert;
  procedure Copy;
  procedure Delete;
  procedure Move;
  procedure Previous (Repeat : Positive := 1);
  procedure Next (Repeat : Positive := 1);
  procedure Parent (Repeat : Positive := 1);
  procedure Child (Repeat : Positive := 1);
  procedure First_Child (Repeat : Positive := 1);
  procedure Last_Child (Repeat : Positive := 1);
end Object;

pragma Subsystem (Object_Editor);
pragma Module_Name (4, 2215);

end Common;

```

```

with Action;
package Compilation is

  subtype Name is String;
  subtype Unit_Name is String;

  -- All names are resolved in the established naming context for the job.

  -- A parameter of type Unit_Name may designate a set of Ada units,
  -- Worlds, Directories, or Activities. If a world or directory is
  -- designated, all Ada units contained by that world or directory are
  -- operated on. If an activity is given, all Ada units in the views
  -- specified by the Activity are operated on.

  type Unit_State is (Archived, Source, Installed, Coded);

  subtype Change_Limit is String;

  -- Parameters of type Change_Limit control which units an operation is
  -- allowed to change in order to perform its task. Three special values
  -- are predefined:

  Same_Directories : constant Change_Limit := "<DIRECTORIES>";
  Current_Directory : constant Change_Limit := Same_Directories;

  -- Only units in the same directories as the units specified to the
  -- operation are allowed to change.

  Same_Worlds : constant Change_Limit := "<WORLDS>";
  Same_World : constant Change_Limit := Same_Worlds;

  -- Only units in the same worlds as the units specified to the operation
  -- are allowed to change.

  All_Worlds : constant Change_Limit := "<ALL_WORLDS>";

  -- A unit in any world may be changed.

  -- A Change_Limit parameter may also be a string name that designates a
  -- set of worlds, directories or activities. Only units in the
  -- designated worlds or directories are allowed to change. The set of
  -- worlds designated by an activity is the set of views referenced by
  -- that activity.

  procedure Demote (Unit : Unit_Name := "<SELECTION>";
    Goal : Unit_State := Compilation.Source;
    Limit : Change_Limit := "<WORLDS>";
    Effort_Only : Boolean := False;
    Response : String := "<PROFILE>");

  -- All units that must be demoted in order to demote the specified
  -- unit will be demoted if possible. Any messages are appended to the
  -- log file.

  procedure Parse (File_Name : Name := "<REGION>";
    Directory : Name := "#";
    List : Boolean := False;

```

```

Limit : Change_Limit := "<WORLDS>";
Response : String := "<PROFILE>";

-- Deletes and expunges all versions of the named unit and its subunits.
-- Wildcard notation may be used to specify more than one unit to be
-- destroyed. The Threshold is the number of objects to be destroyed per
-- unit specified.

procedure Compile (File_Name : Name := "<REGION>";
                  Library : Name := "q";
                  Goal : Unit_State := Compilation.Installed;
                  List : Boolean := False;
                  Source_Options : String := "";
                  Limit : Change_Limit := "<WORLDS>";
                  Response : String := "<PROFILE>");

-- Parses and promotes the units in the given file_name(s) (wildcards
-- allowed) to the given Goal state in the given Library according to
-- the Chapter 10 LRM rules for libraries. If List is true a source
-- listing with interleaved error messages will be generated to the log
-- file.

procedure Dependents (Unit : Unit_Name := "<IMAGE>";
                    Transitive : Boolean := False;
                    Response : String := "<PROFILE>");

-- Displays the installed units that depend on (with) the given unit(s);

procedure Atomic_Destroy (Unit : Unit_Name;
                        Success : out Boolean;
                        Action_Id : Action.Id := Action.Null_Id;
                        Limit : Change_Limit := "<WORLDS>";
                        Response : String := "<PROFILE>");

-- Deletes and expunges all versions of the named unit and its subunits.
-- Wildcard notation may be used to specify more than one unit to be
-- destroyed. The operation succeeds only if all designated units can
-- be destroyed.

procedure Load
  (From : String; To : String; Response : String := "<PROFILE>");
-- Produce a Loaded_Main program from the main program specified by From.
-- Put the result at To.

procedure Set_Target_Key (The_Key : String := "?";
                        To_World : String := "<IMAGE>";
                        Response : String := "<PROFILE>");

-- Assign the target key to the specified world. Once a key has
-- been assigned to a world, the assignment can be changed only if
-- the new key and the old key differ only in the front end/back end
-- policy sub-components. The default Key string, "?", causes a
-- list of all available keys to be displayed.

procedure Show_Target_Key (For_World : String := "<IMAGE>";
                        Response : String := "<PROFILE>");

```

```

Source_Options : String := "";
Response : String := "<PROFILE>";

-- The named file must contain Ada source for a compilation. After it
-- is parsed, the library compilation units are placed in the designated
-- Directory. LIST => true generates a listing of the input file into
-- the log file. Wildcards in the File_Name are supported.

type Promote_Scope is (Single_Unit, Unit_Only, Subunits_Too,
                    All_Parts, Load_Views);

procedure Promote (Unit : Unit_Name := "<IMAGE>";
                 Scope : Promote_Scope := Compilation.Subunits_Too;
                 Goal : Unit_State := Compilation.Installed;
                 Limit : Change_Limit := "<WORLDS>";
                 Effort_Only : Boolean := False;
                 Response : String := "<PROFILE>");

-- Attempts to promote the units designated by the Unit parameter to the
-- designated Goal. The operation is a no-op if the units are already at
-- or beyond the goal state.

-- Unless the Scope is Single_Unit, Promote will attempt to promote the
-- ancestor units of, the visible part of, and any units with'ed by the
-- designated units before promoting the designated units. The with'ed
-- units must exist in the libraries specified by the Limit parameter.

-- Promotion of other units is NOT attempted; specifically: promotion of
-- siblings is NOT attempted. If a designated unit is a visible part,
-- promotion of the body is NOT attempted.

-- Specifying Scope => Subunits_Too will cause subunits of the
-- designated units to be promoted. Specifying Scope => All_Parts is
-- equivalent to the Make procedure described below.

-- Semantic messages are attached to the tree. Semantic and other
-- messages are appended to the end of the Log_File.

procedure Make (Unit : Unit_Name := "<IMAGE>";
              Scope : Promote_Scope := Compilation.All_Parts;
              Goal : Unit_State := Compilation.Coded;
              Limit : Change_Limit := "<WORLDS>";
              Effort_Only : Boolean := False;
              Response : String := "<PROFILE>") renames Promote;

-- Same as Promote except that an attempt is made to promote the
-- secondary units of each visible part promoted.

procedure Delete (Unit : Unit_Name := "<SELECTION>";
                Limit : Change_Limit := "<WORLDS>";
                Response : String := "<PROFILE>");

-- Demotes and deletes the default version of the named unit and its
-- subunits.

procedure Destroy (Unit : Unit_Name := "<SELECTION>";
                 Threshold : Natural := 1;

```

```

-- Displays in the log the target key currently assigned to the
-- indicated world.
function Get_Target_Key (For_World : String := "<IMAGE>") return String;
-- returns the image of the target key assigned to the indicated
-- world.

pragma Subsystem (Commands);
pragma Module_Name (4, 3936);

end Compilation;

```

```

with Calendar;
package Daemon is

-- There are five types of Daemon tasks controlled by this package, their
-- characteristics and default scheduling:
--
-- Snapshot. Erequent. "1 minute slowdown. Hourly.
--
-- Action. Erequent, unobtrusive. Every two hours.
--
-- Weekly. Unobtrusive. Weekly at 2:30 AM.
-- Code_Segment Group Session Tape Terminal User
--
-- Daily. Variable, possibly significant interruption.
-- Nightly at 3:00 AM.
-- Ada DDB Directory Error_Log File Disk
--
-- Disk. Daily or as needed. Prolonged slowdown.
-- Last portion of the Daily run
--
-- If no other action is taken, all clients will be scheduled at a
-- frequency and time normally appropriate. These schedules can be
-- changed to suit specific needs. Note that Disk is included in the
-- Daily category and will be run with the other Daily Daemons.
--
-- Clients that interfere with normal operations warn all users.
--
-- There is a group of clients referred to as Major_Clients that are
-- expected to be of interest in monitoring the state of the machine:
-- Snapshot, Action, Disk, Ada, DDB, Directory, and File.
Major_Clients : constant String := "+";

procedure Run (Client : String := "Snapshot";
Response : String := "<PROFILE>");
-- Cause the named Client to run the specified operation immediately;
-- Has no effect on the next scheduled run of Client.

procedure Schedule (Client : String := ">>CLIENT NAME<<";
Interval : Duration;
First_Run : Duration := 0.0;
Response : String := "<PROFILE>");

-- Sets the interval at which the Client operation will take place.

procedure Quiesce (Client : String := ">>CLIENT NAME<<";
Additional_Delay : Duration := 86_400.0;
Response : String := "<PROFILE>");
-- Reschedule the Client not to run at the next scheduled time.
-- Equivalent to Schedule with a new First_Run, but the same Interval.
-- Defaults to a 1-day delay; use Duration'Last for indefinite delay.

procedure Status (Client : String := "+");
-- Print a formatted display of current status for given Client
-- Matches on prefix of Client name, "" is prefix of all clients
-- Major Clients (*): Actions, Ada, DDB, Directory, Disk, File, Snapshot
-- The Disk Client provides additional information when run separately.

procedure Warning_Interval (Interval : Duration := 120.0);
Daemon, !Commands

```

```

function Get_Warning_Interval return Duration;
-- Warning given before starting Daily clients to allow time to Quiesce.

function In_Progress (Client : String) return Boolean;
function Next_Scheduled (Client : String) return Calendar.Time;
function Last_Run (Client : String) return Calendar.Time;
function Interval (Client : String) return Duration;
procedure Get_Size (Client : String;
                   Size : out Long_Integer;
                   Size_After_Last_Run : out Long_Integer;
                   Size_Before_Last_Run : out Long_Integer);
-- Sizes are set to -1 if invalid

-- Control of the Disk Daemon

-- The Disk Daemon runs in response to a number of stimuli:
--
-- Daemon.Schedule Runs at priority 6; intended for machine idle.
-- Daemon.Run Runs at priority -1; background collection.
-- Daemon.Collect Runs at specified priority
-- over threshold Starts at priority 0 with escalation
--
-- Messages to all users are issued for each of the three explicitly
-- called collections. In addition, a message is sent when a Set_Priority
-- is called and it causes a change in priority.
--
-- A background task monitors over threshold situations and sends messages
-- of interesting events. Threshold_Warnings (False) allows an
-- installation-provided job to tailor policy.
--
-- Additional control over Disk operations is available in the
-- Disk_Daemon tools package.
--
subtype Volume is Integer range 0 .. 31;
subtype Collection_Priority is Integer range -1 .. 6;
-- -1 is the default and implies very low-level background activity
-- 0 guarantees progress in collection but has some effect on response
-- 6 causes collection to take over the machine

procedure Collect (Vol : Volume; Priority : Collection_Priority := 0);
-- If this call initiates a collection, it waits for its completion.

procedure Set_Priority (Priority : Collection_Priority := -1);
-- Set the priority of a currently running collection to Priority

procedure Threshold_Warnings (On : Boolean := True);
-- Cause messages to be sent when collection thresholds are passed.

-- Control of snapshot messages

--
--
procedure Snapshot_Warning_Message (Interval : Duration := 120.0);
procedure Snapshot_Start_Message (On : Boolean := True);
procedure Snapshot_Finish_Message (On : Boolean := True);
procedure Show_Snapshot_Settings;
procedure Get_Snapshot_Settings (Warning : out Duration;
                                Start_Message : out Boolean;
                                Finish_Message : out Boolean);

```

```

-- Control of the contents and permanence of the operations error log
--
--
type Condition_Class is (Normal, Warning, Problem, Fatal);
type Log_Threshold is (Console_Print, Log_To_Disk, Commit_Disk);

procedure Show_Log_Thresholds;
procedure Set_Log_Threshold (Kind : Log_Threshold; Level : Condition_Class);
function Get_Log_Threshold (Kind : Log_Threshold) return Condition_Class;

-- Options on client compactions.
--
-- Consistency checking does additional work to assure that the internal
-- state of the system is as it seems. This is normally only run when
-- there are suspected problems. Consistency checking slows operations
-- for which it is meaningful by between one and three orders of magnitude.
--
-- Access_List_Compaction is the process of removing non-existent groups
-- from the access lists of objects. This condition occurs when groups
-- are removed from the machine. Access_List_Compaction is only done
-- for Ada, Directory and File clients. All other clients requested will
-- be silently ignored. All three must be compacted for any old group
-- numbers to be freed.
--
-- The default is disabled. The default is restored after
-- the next appropriate daemon run has completed.

procedure Set_Consistency_Checking (Client : String := "");
On : Boolean := True;
Response : String := "<PROFILE>";

function Get_Consistency_Checking (Client : String := "") return Boolean;

procedure Set_Access_List_Compaction (Client : String := "");
On : Boolean := True;
Response : String := "<PROFILE>";

function Get_Access_List_Compaction (Client : String := "") return Boolean;

pragma Subsystem (Os_Commands);
pragma Module_Name (4, 3932);

end Daemon;

```

package Debug is

```
subtype Path_Name is String;
subtype Task_Name is String;
subtype Exception_Name is String;
subtype Hex_Number is String;
```

-- A Path\_Name is used to reference declarations, objects, statements,  
-- stack frames, tasks or types within program units.

-- Many commands take both a Path\_Name and a Stack\_Frame. Though  
-- the Path\_Name type allows the specification of a stack frame, the  
-- addition of the Stack\_Frame parameter as a numeric value makes it  
-- possible to specify the stack frame as a numeric argument from the  
-- keyboard. If both a Stack\_Frame and Path\_Name are specified, the  
-- Path\_Name will be interpreted as the string Stack\_Frame & Path\_Name.

-- Task\_Name may be either a hex number or string name for the task.  
-- Exception\_Name may be either a simple name for a predefined exception,  
-- or a pathname to an Ada identified.

-- A Task\_Name parameter of "all" specifies all tasks. A Task\_Name  
-- parameter of " " is interpreted as the control context task if explicitly  
-- set, otherwise, all tasks. Exceptions to this rule are the commands  
-- Run and Stack, for which a Task\_Name parameter of " " specifies the  
-- last task to stop if the control context is not explicitly set.

-- Commands to terminate debugging

```
procedure Debug_Off (Kill_Job : Boolean := False);
-- Debug_Off terminates debugging on the job. The job will run to
-- completion if Kill_Job is false. Otherwise, the job is terminated.

procedure Kill (Job : Boolean := True; Debugger : Boolean := False);
-- Kill can be used to kill either the job being debugged, or the
-- debugger itself.
```

-- Commands to query and modify program state

```
procedure Put (Variable : Path_Name := "<SELECTION>";
Stack_Frame : Integer := 0);
-- Display the value of the given object.
```

```
procedure Stack (For_Task : Task_Name := "";
Start : Integer := 0;
Count : Natural := 0);
```

-- Display Count stack frames for the specified task starting from frame  
-- Start.

```
procedure Modify (New_Value : String := "";
Variable : Path_Name := "<SELECTION>";
Stack_Frame : Integer := 0);
```

-- Modify the value of the given object.

-- Commands to display ADA source

Debug, !Commands

```
procedure Display (Location : Path_Name := "<SELECTION>";
Stack_Frame : Integer := 0;
Count : Natural := 0);
```

-- Display the source code for the given Location in the debugger window.  
-- If the Location specifies a subprogram, package, or task, display  
-- Count lines of source code including line numbers.

```
procedure Source (Location : Path_Name := ""; Stack_Frame : Integer := 0);
-- Like Definition, display the Location in an ada image.
```

-- Breakpoint handling commands; break 0 represents all breaks

```
procedure Break (Location : Path_Name := "<SELECTION>";
Stack_Frame : Integer := 0;
Count : Positive := 1;
In_Task : Task_Name := "");
```

-- Default\_Lifetime : Boolean := True);  
-- Set a break at the given location for the specified task. Count is  
-- the number of times the location is executed before the break is active.  
-- When Default\_Lifetime is true, the breakpoint is temporary or permanent  
-- as specified by the Permanent\_Breakpoints option; if false, its  
-- permanence is the opposite of the option.

-- The breakpoint will be given a unique number which can be used as the  
-- breakpoint parameter of the Remove and Activate commands.

```
procedure Remove (Breakpoint : Natural; Delete : Boolean := False);
-- Deactivate the given breakpoint. With delete false, the breakpoint  
-- can be installed again with the Activate command.
-- Use Show (Breakpoints) to display breaks.
```

```
procedure Activate (Breakpoint : Natural);
-- Install a previously removed breakpoint.
```

-- Commands to control all or individual tasks

```
procedure Stop (Name : Task_Name := "");
-- Stops execution of the specified task and keeps it stopped until  
-- started by a call to Execute or Run naming the task or "all".
```

```
procedure Execute (Name : Task_Name := "");
-- Starts execution of the specified task if stopped.
```

```
procedure Xecute (Name : Task_Name := "");
-- same as Execute.
```

```
procedure Hold (Name : Task_Name := "");
```

-- Stops execution of the specified task and put it in the held state  
-- until explicitly released by the command Release or a call to Execute or  
-- Run explicitly naming this task. The held state differs from the  
-- stopped state in that Execute ("all") will not run a held task.

```
procedure Release (Name : Task_Name := "");
```

-- Releases a task from the held state and moves it to the stopped  
-- state. The task can then be started by a call to Execute or Run naming  
-- the task or "all".

```

type Task_Category is
  (All_Tasks,
   -- all known tasks
  Blocked,
   -- tasks not in debugger, but not currently running
  Held,
   -- tasks held in debugger (Hold command)
  Not_Running,
   -- tasks not running for any reason
  Running,
   -- tasks that are currently ready to run
  Stopped);
   -- tasks stopped in the debugger (eg, at breakpoints)

procedure Task_Display (For_Task : Task_Name := "";
  Task_Set : Task_Category := Debug.All_Tasks);
  -- Display information about tasks in the given category.

type Stop_Event is
  (About_To_Return,
   -- stop after last statement of a subprogram
  Begin_Rendezvous,
   -- stop before first statement of accept body
  End_Rendezvous,
   -- stop after last statement of accept body
  Local_Statement,
   -- stop before next statement at same level
  Machine_Instruction,
   -- stop before next instruction
  Procedure_Entry,
   -- stop before first stat/decl of called proc
  Returned,
   -- stop before next statement in caller
  Statement);
   -- stop before next statement

procedure Run (Stop_At : Stop_Event := Debug.Statement;
  Count : Positive := 1;
  In_Task : Task_Name := "");
  -- Execute the specified task until the stop event has occurred
  -- Count times.

procedure Clear_Stepping (For_Task : Task_Name := "");
  -- Cancel any stepping operations for the given task.

  -- Exception handling commands

procedure Catch (Name : Exception_Name := "<SELECTION>";
  In_Task : Task_Name := "";
  At_Location : Path_Name := "");
  -- Stop execution when the specified exception is raised. Can be
  -- limited to a particular task or location. Name = "all" catches
  -- all exceptions; Name = "implicit" will catch implicitly raised
  -- exceptions.

procedure Propagate (Name : Exception_Name := "<SELECTION>";
  In_Task : Task_Name := "";
  At_Location : Path_Name := "");
  -- Request that execution not be stopped when the given exception is raised.

procedure Forget (Name : Exception_Name := "<SELECTION>";
  In_Task : Task_Name := "";
  At_Location : Path_Name := "");
  -- Cancel a catch or propagate request.

  -- Tracing commands

type Trace_Event is
  (All_Events,
   -- Produce message for all of below

```

```

Call,
  -- Message for each subprogram entry
  Exception_Raised,
  -- Message for each exception raised
  Machine_Instruction,
  -- Message for each statement/decl
  Propagate_Exception,
  -- Message for each frame popped by propagation
  Rendezvous,
  -- Message for each rendezvous start and end
  Statement);
  -- Message for each statement/decl

procedure Trace (On : Boolean := True;
  Event : Trace_Event := Debug.All_Events;
  In_Task : Task_Name := "";
  At_Location : Path_Name := "<SELECTION>";
  Stack_Frame : Integer := 0);
  -- Enable or disable tracing. Tracing displays information about
  -- the execution of the given_task when the specified Trace_Events
  -- occur.

procedure Trace_To_File (File_Name : String := ">>> FILE NAME <<>";
  -- Send trace output to the specified file. The null string
  -- causes output to go to the debugger window.

  -- History commands

procedure History_Display (Start : Integer := 0;
  Count : Integer := 0;
  For_Task : Task_Name := "");
  -- Display Count history entries for the given task. If Start is positive,
  -- it specifies the starting location from the newest entry; if negative,
  -- from the oldest entry.

procedure Take_History (On : Boolean := True;
  Event : Trace_Event := Debug.All_Events;
  For_Task : Task_Name := "";
  At_Location : Path_Name := "<SELECTION>";
  Stack_Frame : Integer := 0);
  -- Enable or disable history taking for the given task and location.

  -- Commands to query debugger state

type Context_Type is (Control, Evaluation);

procedure Context (Set : Context_Type := Debug.Control;
  To_Bo : Path_Name := "<SELECTION>";
  Stack_Frame : Integer := 0);
  -- Set either the control or evaluation context. Control context
  -- is generally used when a Task_Name parameter of " " is specified.
  -- The evaluation context is used as a prefix for unqualified location
  -- and object names.

type Option is
  (Addresses,
   -- Include machine information
  Break_At_Creation,
   -- Tasks stop before first decl
  Declaration_Display,
   -- Include declarations in program display
  Delete_Temporary_Breaks,
   -- Delete (vs deactivate) temp breakpoints
  Display_Creation,
   -- Trace message for each task creation
  Echo_Commands,
   -- Echo command in debugger window
  Freeze_Tasks,
   -- Stop all tasks when one stops

```



```

Include_Packages,
Interpret_Control_Words,
Kill_Old_Jobs,
Machine_Level,
No_History_Timestamps,
Optimize_Generic_History,
Permanent_Breakpoints,
Put_Locals,
Qualify_Stack_Names,
Require_Debug_Off,
Save_Exceptions,
Show_Location,
Timestamps);

-- Task display includes packages
-- Memory display for control stacks
-- Kill last debug job when next is begun
-- Allow certain machine level operations
-- History display option
-- No generic instance in history
-- Default breakpoints to permanent (vs temp)
-- Put displays locals as well as parameters
-- Use fully qualified names in stack display
-- Debug_Off needed before debug next job
-- Save exception-handling state across jobs
-- Display source in image when task stops
-- Include timestamps in command log

procedure Enable (Variable : Option; On : Boolean := True);
procedure Disable (Variable : Option; On : Boolean := False) renames Enable;
-- Enable or disable the specified option.

type Numeric is
(Display_Count,
Display_Level,
Element_Count,
First_Element,
History_Count,
History_Entries,
History_Start,
Memory_Count,
Pointer_Level,
Stack_Count,
Stack_Start);

-- Default for Count in Display command
-- Number of levels to expand Put command's data
-- Max elements of array for Put to display
-- Offset for start of Put's array display
-- Default for Count in History_Display
-- History buffer size
-- Default for Start in History_Display
-- Default for Memory_Dump Count parameter
-- Number of pointers to expand in Put's data
-- Default frame Count for Stack command
-- Default for Start in Stack command

procedure Set_Value (Variable : Numeric; To_Value : Integer);

procedure Flag (Variable : String := ""; To_Value : String := "TRUE");

type State_Type is (All_State, Breakpoints, Contexts,
Exceptions, Flags, Histories, Libraries,
Special_Types, Steps, Stops_And_Holds, Traces,
-- internal debugger state
Active_Items, Exception_Cache, Inner_State, Statistics);

procedure Show (Values_For : State_Type := Debug.Breakpoints);
-- Display information about various debugger facilities.

type Information_Type is (Exceptions, Rendezvous, Space);

procedure Information (Info_Type : Information_Type := Debug.Exceptions;
For_Task : Task_Name := "");
-- Display information about the specified task.

procedure Comment (Information : String := "");
-- place a comment in the debugger window.

procedure Set_Task_Name (For_Task : Task_Name := "";
To_Name : String := "");
-- Set a task synonyma for the specified task for use as a Task_Name
-- parameter to commands.

procedure Convert (Number : String := ""; To_Base : Natural := 0);

```

```

-- Hex/decimal conversion.

procedure Reset_Defaults;
-- Reset flags to initial values.
-- Unregister all special types.

procedure Current_Debugger (Target : String := "");
-- Set current debugger to the current window, or Target if
-- specified. Subsequent calls to Dabug will be directed to
-- the specified target or native debugger.

-- Machine-level commands

-- For the following commands, address format is #Segment, #Offset
-- memory format is one of CONTROL, TYP, QUEUE, DATA, IMPORT, CODE, SYSTEM

procedure Memory_Display (Address : String := "";
Count : Natural := 0;
Format : String := "DATA");

procedure Location_To_Address (Location : Path_Name := "<SELECTION>";
Stack_Frame : Integer := 0);
procedure Address_To_Location (Address : String := "");
procedure Exception_To_Name (Implementation_Image : String := "");

pragma Subsystem (Native_Debugger);
pragma Module_Name (4, 3801);

end Debug;

```

```

package Diana_Tree is
  procedure Ada_Edit (Name : String := "<IMAGE>");
  pragma Subsystem (Command);
  pragma Module_Name (4, 2211);
end Diana_Tree;

```

```

package Disk_Space is
  type Acceptable is (Any_Space, Any_Permanent_Space,
    Committed_Permanent_Spaces,
    Undeleted_Committed_Permanent_Spaces);
  type Traversals is (Poly_File_Space, Directory_Space, Eedb_Space,
    Constant_Space, Moribund_Space, Backup_Database_Space);
  type Traversing is array (Traversals) of Boolean;
  All_Traversals : constant Traversing := Traversing'(others => True);
  Directory_Only : constant Traversing :=
    Traversing'(Directory_Space => True, others => False);
  No_Traversals : constant Traversing := Traversing'(others => False);

  -- Possible decodings of a space.
  -- Class (R1000_Native_Code .. R1000_Cross_Code) are instruction spaces.
  -- Class (R1000_Import) is any import space.
  -- Class (Diana_Tree .. Other) are module spaces.
  -- Class (Diana_Tree .. Seg_Heap_Other) are all segmented heaps.
  -- Class (Poly_Text .. Poly_Other) are all Polymorphic Io creations.
  -- Class (Backup_Master .. Backup_Tape) are Backup database spaces.
  -- Class (Garbage) is a garbage collected (by the Disk_Cleaner) space.

  type Class is (R1000_Native_Code, R1000_Cross_Code, R1000_Import,
    Diana_Tree, Text_File, Image, Link_Pack,
    Poly_Text, Poly_Object_Id, Poly_State, Poly_Other,
    Backup_Id, Backup_Backup, Backup_Processor,
    Backup_Disk, Backup_Tape, Backup_Master,
    Configuration, Seg_Heap_Other, Garbage, Other);

  type Classes is array (Class) of Boolean;

  All_Classes : constant Classes := Classes'(others => True);
  Module_Classes : constant Classes := Classes'(R1000_Native_Code => False,
    R1000_Cross_Code => False,
    R1000_Import => False,
    others => True);
  Matching_Class : constant Classes := Classes'(others => False);
  Unknown_Classes : constant Classes := Classes'(Poly_Other => True,
    Seg_Heap_Other => True,
    Other => True,
    others => False);

  type Space_Kind is (Instruction, Import, Module);
  Data : constant Space_Kind := Module;

```

```

-- Examine_Spaces locates all spaces known to the kernel and discards
-- any that are either unacceptable or can be reached through one of
-- the traversals.
-- The Summarize booleans cause listings of the space counts / sizes to
-- be printed. List_Lost causes Space_Information for the unreachable
-- spaces to be printed.

```

```

procedure Examine_Spaces

```

```

Disk_Space, !Commands

```

```

Diana_Tree, !Commands

```

```

(Examine : Traversing := Disk_Space.All_Traversals;
 Filter : Acceptable :=
   Disk_Space.Undeleted_Committed_Permanent_Spaces;
 Permit : Classes := Disk_Space.All_Classes;
 Summarize_All : Boolean := False;
 Summarize_Lost : Boolean := True;
 List_Lost : Boolean := False;
 Verbose : Boolean := True);

-- Attempts to find the name of the object which contains the space
-- specified, and prints that name. If the null space is specified
-- (the default values), then the names of all spaces are printed.
-- If the directory system is being searched, Vol_Hint /= 0 will
-- cause the search to attempt to avoid looking on the wrong volume.
-- Root_Name specifies where the directory system search should begin.
procedure Name_Space (Vp : Natural := 0;
 Kind : Space_Kind := Disk_Space.Instruction;
 Segment : Natural := 0;
 Vol_Hint : Natural := 0;
 Root_Name : String := "|";
 Search : Traversing := Disk_Space.All_Traversals;
 Verbose : Boolean := True);

-- Searches just as with Name_Space, but will search for any space
-- with the same Family_Id as the space specified.
procedure Name_Family (Vp : Natural := 0;
 Kind : Space_Kind := Disk_Space.Instruction;
 Segment : Natural := 0;
 Vol_Hint : Natural := 0;
 Root_Name : String := "|";
 Search : Traversing := Disk_Space.All_Traversals;
 Verbose : Boolean := True);

-- Interpret page 0 of the data segment in various ways.
-- Instruction spaces don't have data segments, so only useful for Modules.
procedure Decode_Space (Vp : Natural := 0;
 Kind : Space_Kind := Disk_Space.Module;
 Segment : Natural := 0;
 Match : Classes := Disk_Space.Matching_Class;
 Verbose : Boolean := True);

-- *****
-- Do not use the following commands unless you know what you are doing.
-- *****

type Mark_Type is new Natural range 0 .. 1023;
type Volume_Number is new Natural range 0 .. 31;
type Block_Number is new Natural range 0 .. 2 ** 24 - 1;

type Usage_Array_Type is array (Mark_Type) of Natural;
type Vol_Bit_Map_Array is array (Block_Number range <>) of Boolean;

```

```

type Vol_Usage_Array is array (Block_Number range <>) of Mark_Type;
type System_Usage_Array is
  array (Volume_Number range <>) of Usage_Array_Type;

Unable_To_Acquire_Backup_Lock : exception;
Garbage_Collection_Is_Running : exception;

function First_Volume return Volume_Number;
function Last_Volume return Volume_Number;

function Find_Storage_Consumed return System_Usage_Array;

procedure Clean_Cache;

function Get_Bit_Map (Volume : Volume_Number) return Vol_Bit_Map_Array;

function Find_Current_Usage (Volume : Volume_Number) return Vol_Usage_Array;

pragma Subsystem (Commands);
pragma Module_Name (4, 3935);
end Disk_Space;

```

```

package Editor is
  procedure Cursor is
    procedure Down (Repeat : Integer := 1);
    procedure Left (Repeat : Integer := 1);
    procedure Right (Repeat : Integer := 1);
    procedure Up (Repeat : Integer := 1);
    -- Quarter-plane motion

    procedure Forward (Repeat : Integer := 1);
    procedure Backward (Repeat : Integer := 1);
    -- Stream motion, end of line N adjacent to beginning of line N+1

    procedure Next (Repeat : Integer := 1;
      Prompt : Boolean := True;
      Underline : Boolean := True);
    procedure Previous (Repeat : Integer := 1;
      Prompt : Boolean := True;
      Underline : Boolean := True);
    -- Position the cursor at the next (previous) closest prompt or
    -- underline. Prompt (Underline) false indicates not to look
    -- for the next Prompt (Underline). Both false does nothing

  end Cursor;

  package Search is
    procedure Previous (Target : String := ""; Wildcard : Boolean := False);
    procedure Next (Target : String := ""; Wildcard : Boolean := False);
    procedure Replace_Previous (Target : String := "";
      Replacement : String := "";
      Repeat : Integer := 1;
      Wildcard : Boolean := False);
    procedure Replace_Next (Target : String := "";
      Replacement : String := "";
      Repeat : Integer := 1;
      Wildcard : Boolean := False);

  end Search;

  package Char is
    procedure Capitalize (Repeat : Integer := 1);
    procedure Delete_Backward (Repeat : Integer := 1);
    procedure Delete_Forward (Repeat : Integer := 1);
    -- Stream deletion end of line N is adjacent to beginning
    -- of line N+1

    procedure Delete_Next (Repeat : Integer := 1);
    procedure Delete_Previous (Repeat : Integer := 1);
    -- Quarter-plane deletion

    procedure Delete_Spaces (Remaining : Natural := 1);
    -- Delete spaces surrounding the cursor, leaving remaining spaces

    procedure Insert_String (Value : String);
    procedure Insert_Character (Repeat : Integer := 1; Value : Character);
    procedure Lower_Case (Repeat : Integer := 1);
    procedure Quote;
    procedure Tab_Backward (Repeat : Integer := 1);
    procedure Tab_Forward (Repeat : Integer := 1);
    procedure Tab_To_Comment;

```

Editor, !Commands

```

-- Tab to the comment column and insert comment marks
  procedure Transpose (Offset : Integer := 1);
  procedure Upper_Case (Repeat : Integer := 1);
end Char;

package Line is
  procedure Beginning_Of (Offset : Natural := 0);
  procedure Capitalize (Repeat : Integer := 1);
  procedure Center (Right_Margin : Natural := 0);
  procedure Copy (Repeat : Integer := 1);
  procedure Delete (Repeat : Integer := 1);
  procedure Delete_Backward (Repeat : Integer := 1);
  procedure Delete_Forward (Repeat : Integer := 1);
  procedure End_Of (Offset : Natural := 0);
  procedure Insert (Repeat : Integer := 1);
  procedure Indent (Repeat : Integer := 1);
  procedure Join (Repeat : Integer := 1);
  procedure Lower_Case (Repeat : Integer := 1);
  procedure Open (Repeat : Integer := 1);
  procedure Transpose (Offset : Integer := 1);
  procedure Upper_Case (Repeat : Integer := 1);
  procedure Next (Repeat : Integer := 1) renames Cursor.Down;
  procedure Previous (Repeat : Integer := 1) renames Cursor.Up;
end Line;

package Word is
  procedure Beginning_Of;
  procedure Breaks (Break_Set : String := "";
    Are_Delimiters : Boolean := True);
  procedure Capitalize (Repeat : Integer := 1);
  procedure End_Of;
  procedure Delete (Repeat : Integer := 1);
  procedure Delete_Backward (Repeat : Integer := 1);
  procedure Delete_Forward (Repeat : Integer := 1);
  procedure Lower_Case (Repeat : Integer := 1);
  procedure Next (Repeat : Integer := 1);
  procedure Previous (Repeat : Integer := 1);
  procedure Transpose (Offset : Integer := 1);
  procedure Upper_Case (Repeat : Integer := 1);
end Word;

package Image is
  -- repeat = 0 scrolls one page
  procedure Up (Repeat : Integer := 0);
  procedure Down (Repeat : Integer := 0);
  procedure Left (Repeat : Integer := 0);
  procedure Right (Repeat : Integer := 0);
  procedure Find (Name : String);
  procedure Beginning_Of (Offset : Natural := 0);
  procedure End_Of (Offset : Natural := 0);
end Image;

-- Many of the following packages implement a "stack" discipline. For
-- these packages, the following operations are supported:
-- Copy_Top Push a copy of the top of stack
-- Delete_Top Delete the top element from the stack
-- Next Use the next value on the stack

```

```

-- Previous      Use the previous value on the stack
-- Push         Put the appropriate item on the stack
-- Rotate       Rotate the stack; top becomes the bottom; value not
--             used
-- Swap         Interchange the top and next to top items; value not
--             used
-- Top          Use the top value on the stack

package Screen is
  procedure Down (Repeat : Integer := 1);
  procedure Left (Repeat : Integer := 1);
  procedure Right (Repeat : Integer := 1);
  procedure Up (Repeat : Integer := 1);
  procedure Dump (To_File : String := ">>NAME<<");
  procedure Redraw;
  procedure Clear;
  -- Screen stack operations

  procedure Copy_Top;
  procedure Delete_Top;
  procedure Next (Repeat : Integer := 1);
  procedure Previous (Repeat : Integer := 1);
  procedure Push (Repeat : Integer := 1);
  procedure Rotate (Repeat : Integer := 1);
  procedure Swap;
  procedure Top;
end Screen;

package Window is
  procedure Beginning_Of (Offset : Natural := 0);
  procedure Child (Repeat : Integer := 1);
  procedure Copy;
  procedure Delete;
  procedure Demote;
  procedure Directory;
  procedure End_Of (Offset : Natural := 0);
  procedure Expand (Lines : Integer := 4);
  procedure Focus;
  procedure Frames (Maximum : Positive);
  procedure Join (Repeat : Integer := 1);
  procedure Next (Repeat : Integer := 1);
  procedure Parent (Repeat : Integer := 1);
  procedure Previous (Repeat : Integer := 1);
  procedure Promote;
  procedure Transpose (Offset : Integer := 1);
end Window;

package Macro is
  procedure Start;
  procedure Finish;
  -- Start/Finish the definition of a keyboard macro

  procedure Execute (Repeat : Integer := 1; Prior : Natural := 0);
  -- Execute the current keyboard macro Repeat times. If Prior /= 0
  -- execute the macro with that number.

  procedure Bind (Key : String := "");
  -- bind the current macro to the key name given, e.g. F1, M.F1.

  procedure Save (Expanded : Boolean := False);

```

```

-- Save the current macro state in the user macro file.
-- Expanded causes the file string to be saved in text form.

procedure Restore;
-- Recreate macro state from the user macro file.

end Macro;

package Hold_Stack is
  procedure Copy_Top;
  procedure Delete_Top;
  procedure Next (Repeat : Integer := 1);
  procedure Previous (Repeat : Integer := 1);
  procedure Push (Repeat : Integer := 1);
  procedure Rotate (Repeat : Integer := 1);
  procedure Swap;
  procedure Top;
end Hold_Stack;

package Mark is
  procedure Copy_Top;
  procedure Delete_Top;
  procedure Next (Repeat : Integer := 1);
  procedure Previous (Repeat : Integer := 1);
  procedure Push (Repeat : Integer := 1);
  procedure Rotate (Repeat : Integer := 1);
  procedure Swap;
  procedure Top;
end Mark;

package Region is
  procedure Beginning_Off;
  procedure Capitalize;
  procedure Comment;
  -- Add comment marks to the beginning of the lines in the region
  procedure Copy;
  procedure Delete;
  procedure End_Off;
  procedure Fill (Column : Natural := 0; Leading : String := "");
  procedure Finish;
  procedure Justify (Column : Natural := 0; Leading : String := "");
  -- 0 argument uses default fill column
  procedure Lower_Case;
  procedure Move;
  procedure Off;
  procedure On;
  procedure Start;
  procedure Uncomment;
  procedure Upper_Case;
end Region;

package Set is
  procedure Insert_Mode (On : Boolean := True);
  procedure Fill_Mode (On : Boolean := True);
  procedure Fill_Column (Column : Positive := 72);
  procedure Designation_Off;
  procedure Input_From (File_Name : String := "<SELECTION>");
  procedure Input_Logging_To (File_Name : String := ">>Name<<");
  procedure Input_Logging_Off;
  procedure Tab_Off (Column : Positive);

```

```

procedure Tab_On (Column : Positive);
procedure Tab_Width (Size : Positive := 4);
-- Only to be bound on keys

procedure Argument_Prefix;
procedure Argument_Digit (Argument : Integer := 1);
procedure Argument_Minus;
end Set;

package Key is
procedure Define (Key_Name : String := ">>KEY NAME, e.g. CM_F1<<";
Command_Name : String := ">>COMMAND NAME<<";
Prompt : Boolean := False);
procedure Name (Key_Code : String := "");
procedure Save;
procedure Prompt (Key_Code : String := "");
end Key;
procedure Quit (Ignore_Changes : Boolean := False);
procedure Alert;
procedure Noop;

pragma Subsystem (Command);
pragma Module_Name (4, 2205);

end Editor;

```

```

package File_Uilities is
subtype Name is String;
Current_Output : constant Name := "";

procedure Difference (File_1 : Name := "<REGION>";
File_2 : Name := "<IMAGE>";
Result : Name := "");
Compressed_Output : Boolean := False;
Subobjects : Boolean := False);
-- Find differences between two versions of an object.
-- If Subobjects is True, subobjects are compared as well.
-- Compressed output omits lines that are the same in both objects.
-- Non-compressed output shows every line from both objects,
-- only showing common lines once.

procedure Merge (Original : Name := "";
File_1 : Name := "";
File_2 : Name := "";
Result : Name := "");
-- merge two variants of the same object into new version with all changes
-- Result defaults to Current_Output = ""

procedure Strip (Source : Name := "<SELECTION>"; Target : Name := "");
-- take the output of Merge or Difference and create a clean file

procedure Compare (File_1 : Name := "<REGION>";
File_2 : Name := "<IMAGE>";
Subobjects : Boolean := False;
Ignore_Case : Boolean := False;
Options : String := "");
-- find the first difference between two objects
-- Subobjects=true causes subunits or units in a library to be compared
-- as well as the named units.
-- Ignore_Case=true causes upper and lower case to be treated as
-- equivalent.
-- Options Include: Ignore_Blank_Lines: causes only on-blank lines
-- to be considered in the compare
-- File_2_Has_Wildcards: Interpret characters in File_2
-- as possible Wildcards. Wildcard
-- characters include:
-- - negate next char
-- ? - match any char
-- % - match any Ada ident char
-- $ - match any Ada delimiter
-- \ - quotes next char
-- { - beginning of line
-- } - end of line
-- [ - start of class
-- ] - end of class
-- * - zero or more of prev item

-- Use of Ignore_Case or Ignore_Blank_Lines slows the compare operation
-- moderately with respect to a straight compare. File_2_Has_Wildcards
-- slows the compare dramatically and should only be used if you have
-- a lot of time to wait. The wildcard compare is conducted on a line-
-- by-line basis.

function Equal (File_1 : Name := "<REGION>";
File_Uilities, !Commands

```

```

File_2 : Name := "<IMAGE>";
Subobjects : Boolean := False;
Ignore_Case : Boolean := False;
Options : String := "" return Boolean;
-- Indicates whether the two files are the same
-- See notes under Compare, above.

procedure Find (Pattern : String := "");
File : Name := "<IMAGE>";
Wildcards : Boolean := False;
Ignore_Case : Boolean := True;
Result : Name := "";
function Found (Pattern : String := "");
File : Name := "<IMAGE>";
Wildcards : Boolean := False;
Ignore_Case : Boolean := True) return Natural;
-- find instances of Pattern in File, optionally using Wildcards

procedure Append (Source : Name := ""; Target : Name := "<SELECTION>");
-- append the contents of one file to another

procedure Dump (File : Name := "<SELECTION>";
Page_Number : Natural := 0;
Word_Number : Natural := 0;
Word_Count : Positive := 64);
-- display a hex dump of the file. A "word" is 16 bytes.
-- Defaults dump the first page of the file.

procedure Sort (File : Name := "<IMAGE>";
Result : Name := "";
Key_1 : String := "";
Key_2 : String := "";
Key_3 : String := "");
-- Sort File using Key_n as sort keys.

-- Key_1 is most significant. Key_2 is ignored if Key_1 not specified, etc.
-- No keys cause ascending Ascii sort on full-line compare.

-- Key_n follow form parameter syntax, parameters are first-character
-- unique, so any prefix of the names is sufficient.

FIELD => number

Field is a field on the line. Fields are non-blank characters
separated by blanks. Field 1 is the first field. Field 1
always includes column 1, even if blank. If no field is given,
the entire line, blanks included is the field.

START_COLUMN => number (default is 1)

The starting column relative to the start of the field.

END_COLUMN => number (default is Integer'Last)

The ending column relative to the start of the field.

REVERSE => true | FALSE

```

```

-- True implies sort descending for this key.
-- NUMERIC => true | FALSE
-- Perform the sort on the numeric value of the field represented
-- as a Long_Integer.

-- Examples:
-- "F=2, S=5, E=7, R, N" will sort the field 2, columns 5 through 7,
-- descending (reversed) using a numeric comparison. Fully specified,
-- "Field => 2, Start_Column => 5, End_Column => 7, Reversed, Numeric"
-- "S=10, E=>15" will sort using Ascii ordering columns 10 through 15
-- of the entire line.

pragma Subsystem (Os_Commands);
pragma Module_Name (4, 3929);

end File_Uilities;

```

```

with Profile;
with Compilation;

package Library is

  subtype Name is String;
  -- Lexically and syntactically an Ada Name.

  subtype Simple_Name is String;
  -- A simple Ada name. Basically, an identifier or operator.

  subtype Context_Name is Name;

  -- Treatment of context. There is a current context that constitutes
  -- the assumed naming context. Names are resolved in this context.

  -- The following characters modify the context:
  -- | specifies the Universe context
  -- # specifies the enclosing library for the current context.
  -- ## specifies the enclosing world for the current context.
  -- - specifies the parent of the current context.
  -- @ matches any single name segment (or part thereof)
  -- ? matches 0 or more name segments, only the last of which may be a
  -- world.
  -- ?? matches 0 or more name segments.

  -- The special strings "<IMAGE>", etc., attempt to get the designated
  -- object from the current selection/image.

  -- Note that many commands are recursive by default (they are
  -- recognizable as such by the presence of a Recursive parameter). When
  -- the Recursive parameter is true, all descendants of the specified
  -- objects partake in the operation. When Recursive is false, just the
  -- specified objects partake.

  -- The effects of the Recursive option can also be obtained using "?"
  -- wildcards, but with more writing. In any case, an object is operated
  -- on only once whether it is introduced by an input parameter or the
  -- recursive option or both.

  Error : exception renames Profile.Error;

  -- Only the single exception Error is raised

  procedure Resolve (Name_Of : Name := "<TEXT>";
                    Target_Name : Name := "";
                    Objects_Only : Boolean := True;
                    Response : String := "<PROFILE>");

  -- Print the Full name for Name_Of. Defaults to the current selection's
  -- text.

  procedure Enclosing_World (Levels : Positive := 1;
                             Response : String := "<PROFILE>");
  -- Enclosing_World is equivalent to Context ("##");

  procedure Context (To_Be : Context_Name := "");

```

```

with Machine;
package Job is

  subtype Id is Machine.Job_Id;
  -- start, stop and terminate a job

  procedure Kill (The_Job : Id; The_Session : String := "");
  procedure Disable (The_Job : Id; The_Session : String := "");
  procedure Enable (The_Job : Id; The_Session : String := "");
  procedure Interrupt;

  procedure Connect (The_Job : Id := 0);
  procedure Disconnect (The_Job : Id := 0);
  procedure Set_Termination_Message (S : String := "");

  pragma Subsystem (Command);
  pragma Module_Name (4, 2206);
end Job;

```



```
Response : String := "<PROFILE>";
```

```
-- Set the job context to To.Be. When To.Be is already the job context,  
-- only printing takes place.
```

```
procedure Copy (From : Name := "<REGION>";  
               To : Name := "<IMAGE>";  
               Recursive : Boolean := True;  
               Response : String := "<PROFILE>";  
               Copy_Links : Boolean := True;  
               Options : String := "");
```

```
-- Copy version From resulting in version To; see table below.
```

```
-- To designates an object that will exist after the copy has  
-- completed. For Ada objects, changing the simple name may require  
-- user intervention before installation.
```

```
-- To is interpreted in the current context or specified full  
-- context and must be unique.
```

```
-- The object designated by To will be the same class as From.
```

```
-- Objects representing devices cannot be copied.
```

```
-- Any situation that would require demoting unrelated declarations  
-- results in an error, suppressing the copy.
```

```
-- Recursive applies to objects that contain other objects and indicates  
-- that these contained objects should be copied.
```

```
-- If Copy_Links is true, then link packs for any worlds copied are  
-- duplicated, and any link which pointed to the source for a copy is  
-- altered to point to the destination. If Copy_Links is false, any  
-- copied worlds will have empty link packs.
```

```
-- If a world and its switch file are copied, then the copied unit will  
-- point to the copy of the switch file. If the switch file is not  
-- copied, then the unit and its original will reference the same switch  
-- file.
```

```
-- Ada units are copied as source.
```

```
-- Copy and Move subsume the functionality of Copy_Into and Move_Into  
-- from previous releases. Whether a Copy/Move is "To" or "Into" is  
-- determined by the type of object specified by the From and To  
-- parameters. The chart below gives the details.
```

```
-- If wildcards/substitution characters are involved in the From and To  
-- parameters, this matrix is applied AFTER these wildcards have been  
-- expanded. If the source is over-specified (e.g., "?") is used with  
-- the recursive switch) a source object is copied only once.
```

```
FROM COPY/MOVE to/into matrix
```

	TO								
FROM	Non-Ada Object	Library Unit	Subunit	World	Drctry	No Object			

Non-Ada Object	TO (1)	Error	Error	INTO	INTO	INTO	TO
Library Unit (2)	Error	TO	TO	INTO	INTO	INTO	TO
Subunit (2)	Error	INTO	TO	INTO	INTO	INTO	TO
World (3)	Error	Error	Error	TO (4)	TO (4)	TO (4)	TO
Drctry (3)	Error	Error	Error	TO (4)	TO (4)	TO (4)	TO

Notes:

1. User can make any "TO" an "INTO" by appending ".name" to To;  
Appending ".#" would yield target with same simple name as From.
2. Any class mismatch is an error.
3. Subunits of unit are involved if Recursive switch is set;  
nesting of subunits is preserved.
4. Subcomponents of library are involved if Recursive switch is set;  
relative nesting of subcomponents is preserved.
5. Contents of source library are merged with contents of  
target library.

```
procedure Move (From : Name := "<REGION>";  
              To : Name := "<IMAGE>";  
              Recursive : Boolean := True;  
              Response : String := "<PROFILE>";  
              Copy_Links : Boolean := True;  
              Options : String := "");
```

```
-- Equivalent to Copy (Existing, ...); Delete (Existing);
```

```
subtype Volume is Natural range 0 .. 31;  
Nil : constant Volume := Volume First;
```

```
type Kind is (World, Directory, Subpackage);
```

```
procedure Create (Name : Library_Name := ">>LIBRARY NAME<<";  
                Kind : Library_Kind := Library.Directory;  
                Vol : Volume := Library.Nil;  
                Model : String := "Model.R1000";  
                Response : String := "<PROFILE>");
```

```
-- Create a library of the specified type. The Nil volume represents
-- the 'best' volume. Vol is ignored for Subpackages, which are not
-- control points, and must be on the same volume as their parent.
-- When creating a World, links are copied from Model (unless it is "").
```

```
procedure Rename (From : Name := "<SELECTION>";
                  To : Simple_Name := ">>NEW SIMPLE NAME<<";
                  Response : String := "<PROFILE>");
```

```
-- Change the name of an existing library unit or managed object.
-- References to library units are not changed -- only the actual
-- name of the unit. Various other restrictions apply.
```

```
procedure Delete (Existing : Name := "<SELECTION>";
                  Limit : Compilation.Change_Limit := "<DIRECTORIES>";
                  Response : String := "<PROFILE>")
renames Compilation.Delete;
```

```
-- Delete versions of objects designated by Existing. Either an object
-- must be selected, or the name of an object supplied.
```

```
-- Results will be reversible with Undelete, unless retention count = 0.
```

```
procedure Destroy (Existing : Name := "<SELECTION>";
                  Threshold : Natural := 1;
                  Limit : Compilation.Change_Limit := "<DIRECTORIES>";
                  Response : String := "<PROFILE>")
renames Compilation.Destroy;
```

```
-- Destroy versions and associated declarations designated by Existing.
-- Destroyed versions are expunged and cannot be undeleted.
```

```
procedure Undelete (Existing : Name := "<CURSOR>";
                   Response : String := "<PROFILE>");
```

```
-- Undelete an Existing version.
```

```
-- Only a fixed number of deleted versions will be retained. Excess
-- versions will be automatically expunged, at which time they can no
-- longer be undeleted.
```

```
Default_Keep_Versions : constant := -1;
```

```
-- Keep the default number of deleted versions.
```

```
procedure Expunge (Existing : Name := "<IMAGE>";
                  Keep_Versions : Integer := 0;
                  Recursive : Boolean := True;
                  Response : String := "<PROFILE>");
```

```
-- Make deletions permanent. Recursive causes subobjects to be
-- expunged. Keep_Versions deleted versions will be retained.
-- Recursive causes subobjects to be touched. Use Recursive => false
-- and "?" wildcard to avoid expunging nested worlds.
```

```
procedure Set_Retention_Count
```

```
(Existing : Name := "<IMAGE>";
  Keep_Versions : Integer := Library.Default_Keep_Versions;
  Recursive : Boolean := True;
  Response : String := "<PROFILE>");
```

```
-- Set the default number of deleted versions of an object which are
-- retained. Default is the same as the object's parent. Recursive
-- causes subobjects to be touched. Use Recursive => false and "?"
-- wildcard to avoid setting retention count for nested worlds.
```

```
procedure Freeze (Existing : Name := "<IMAGE>";
                  Recursive : Boolean := True;
                  Response : String := "<PROFILE>");
```

```
-- Prevent further changes to an object. Recursive causes subobjects to
-- be frozen. Use Recursive => false and "?" wildcard to avoid freezing
-- nested worlds.
```

```
procedure Unfreeze (Existing : Name := "<IMAGE>";
                   Recursive : Boolean := True;
                   Response : String := "<PROFILE>");
```

```
-- Permit changes to an object. Recursive causes subobjects to be
-- unfrozen. Use Recursive => false and "?" wildcard to avoid
-- unfreezing nested worlds.
```

```
procedure Default (Existing : Name := "<SELECTION>";
                  Response : String := "<PROFILE>");
```

```
-- Set the default Version for the existing object and print the result
-- as a message.
```

```
procedure Set_Subclass (Existing : Name := "<SELECTION>";
                       To_Subclass : String := "";
                       Response : String := "<PROFILE>");
```

```
-- Set the subclass of an object. A null string for To_Subclass
-- requests the system to set the subclass to its 'best guess'.
```

```
type Field is (Object,
               Version,
               Class,
               Subclass,
               Update_Time,
               Creator,
               Create_Time,
               Read_Time,
               Size,
               Status,
               Frozen,
               Retain,
               Declaration
              );
-- Ada name.
-- Version name.
-- Directory class name.
-- Subclass of the object.
-- User to last update object.
-- Time of last update.
-- User who created object.
-- Time of creation.
-- User to last read object.
-- Time of last read.
-- Current size of object.
-- Source, Installed, Coded, Elaborated, etc.
-- Is this object frozen.
-- Max. number of deleted versions retained.
-- Ada declaration of object.
```

```
type Fields is array (Field) of Boolean;
```

```

Verbose_Format : constant Fields := Fields'(Object .. Update_Time => True,
Size .. Retain => True,
others => False);

Ada_Format : constant Fields :=
Fields'(Status => True, Declaration => True, others => False);
All_Fields : constant Fields := Fields'(others => True);
Terse_Format : constant Fields := Fields'(Object => True, others => False);

procedure List (Pattern : Name := "<IMAGE>@";
Displaying : Fields := Library.Terse_Format;
Sorted_By : Field := Library.Object;
Descending : Boolean := False;
Response : String := "<PROFILE>";
Options : String := "");

procedure Verbose_List (Pattern : Name := "<IMAGE>{@'V(ALL)}";
Displaying : Fields := Library.Verbose_Format;
Sorted_By : Field := Library.Object;
Descending : Boolean := False;
Response : String := "<PROFILE>";
Options : String := ""); renames List;

procedure File_List (Pattern : Name := "<IMAGE>@'C(FILE)";
Displaying : Fields := Library.Verbose_Format;
Sorted_By : Field := Library.Object;
Descending : Boolean := False;
Response : String := "<PROFILE>";
Options : String := ""); renames List;

procedure Ada_List (Pattern : Name := "<IMAGE>@'C(ADA)";
Displaying : Fields := Library.Ada_Format;
Sorted_By : Field := Library.Declaration;
Descending : Boolean := False;
Response : String := "<PROFILE>";
Options : String := ""); renames List;

procedure Space (For_Object : Name := "<IMAGE>";
Levels : Positive := 2;
Recursive : Boolean := True;
Each_Object : Boolean := False;
Space_Types : Boolean := False;
Response : String := "<PROFILE>";
Options : String := "");

```

```

-- Show the space utilization (in pages) for For_Object. Also
-- display space usage for contained libraries to depth specified
-- by Levels. The space includes subobjects and contained libraries,
-- unless Recursive is false, in which case only the space for the
-- specified object is displayed. Thus, if Recursive is true, the
-- space is cumulatively totalled.
--
-- Each Object causes the individual space the each object to be included
-- in the display in addition to libraries.
--
-- If Space_Types is true, a different display showing space broken down
-- by category (including the object itself, code segment, attribute
-- spaces, and list files) is displayed. In this case, the Each_Version

```

```

-- parameter will show information for each version of each object.
-- Each_Version is used only if Space_Types true. Levels is used only
-- if Space_Types is false.

```

```

procedure Compact_Library (Existing : Name := "<SELECTION>";
Response : String := "<PROFILE>");
-- This procedure may be used to reduce the amount of storage consumed
-- by frequently modified directories which are used to store files.
--
-- Quiet forms similar to those in Library_Object_Editor, but
-- these commands work based on the current context rather than
-- the current image.

procedure Create_World (Name : Library.Name := ">WORLD NAME<<";
Kind : Library.Kind := Library.World;
Vol : Volume := Library.Nil;
Model : String := "INModel.R1000";
Response : String := "<PROFILE>") renames Create;

procedure Create_Directory (Name : Library.Name := ">DIRECTORY NAME<<";
Kind : Library.Kind := Library.Directory;
Vol : Volume := Library.Nil;
Model : String := "";
Response : String := "<PROFILE>")
renames Create;

procedure Create_Unit (Name : Library.Name := ">>ADA NAME<<";
Kind : Library.Kind := Library.Subpackage;
Vol : Volume := Library.Nil;
Model : String := "";
Response : String := "<PROFILE>") renames Create;

procedure Display (Name : Library.Name := "[]");
-- Display the named object in a library window.

procedure Reformat_Image (Existing : Name := "<SELECTION>";
Response : String := "<PROFILE>");
-- Cause the image for a unit to be reconstructed.

pragma Subsystem (Commands);
pragma Module_Name (4, 3921);
end Library;

```

with Links\_Implementation;  
package Links is

subtype World\_Name is String;

-- The string name for any directory object may be given for a world  
-- parameter, to indicate the world that contains the object.

subtype Link\_Name is String;

-- An Ada simple name. When used as an in-parameter, except in Add and  
-- Replace, it may contain wildcard characters. In Add and Replace it  
-- may contain substitution characters.

subtype Source\_Name is String;

-- A directory string name that specifies an existing Ada Library Unit.  
-- (The unit does not have to be installed, but its declaration must be  
-- in a library.) May contain wildcard characters when used as an  
-- in-parameter.

subtype Source\_Pattern is String;

-- A string (containing wildcards) which will be matched against the  
-- full names of the objects denoted by links.

subtype Link\_Kind is Links\_Implementation.Link\_Kind;

Internal : constant Link\_Kind := Links\_Implementation.Internal;  
External : constant Link\_Kind := Links\_Implementation.External;  
Any : constant Link\_Kind := Links\_Implementation.Any;

-- A link is Internal if its source object is in the world of the link  
-- pack; otherwise it is External.

```
procedure Add (Source : Source_Name := ">>SOURCE NAMES<<";  
              Link : Link_Name := "#";  
              World : World_Name := "<IMAGE>";  
              Response : String := "<PROFILE>");
```

-- For each Ada library unit defined by Source, a link is created in the  
-- link pack for World. The Source object is associated with the simple  
-- Ada name given by Link. The operation fails if the specified Link name  
-- already exists in the pack, unless the new link is compatible with the  
-- old link. The new link is defined to be compatible with the old link  
-- iff both links refer to the same object or the object referred to be the  
-- old link has been deleted.

```
procedure Replace (Source : Source_Name := ">>SOURCE NAMES<<";  
                 Link : Link_Name := "#";  
                 World : World_Name := "<IMAGE>";  
                 Response : String := "<PROFILE>");
```

-- For each Ada Library unit defined by Source, a link is created in  
-- the link pack for World. The Source object is associated with the  
-- simple Ada name given by Link. If a link of the same name

Links, !Commands

-- already exists, it is replaced by the new definition.

```
procedure Delete (Link : Link_Name := ">>LINK NAMES<<";  
                Source : Source_Pattern := "?";  
                Kind : Link_Kind := Links.Any;  
                World : World_Name := "<IMAGE>";  
                Response : String := "<PROFILE>");
```

-- The Links that match both the Source and Link wildcards and the  
-- specified kind are deleted from the link pack of the given World.

```
procedure Copy (Source_World : World_Name := ">>WORLD NAME<<";  
               Target_World : World_Name := "<IMAGE>";  
               Link : Link_Name := "@";  
               Source : Source_Pattern := "?";  
               Kind : Link_Kind := Links.Any;  
               Response : String := "<PROFILE>");
```

-- The Links of Source\_World that match the specified Source and Link  
-- names and the given Link\_Kind are copied to Target\_World.

```
procedure Display (World : World_Name := "<IMAGE>";  
                  Link : Link_Name := "@";  
                  Source : Source_Pattern := "?";  
                  Kind : Link_Kind := Links.Any;  
                  Response : String := "<PROFILE>");
```

-- Lists the links that match the given wild cards in the given world

```
procedure Dependents (Link : Link_Name := "@";  
                     Source : Source_Pattern := "?";  
                     Kind : Link_Kind := Links.Any;  
                     World : World_Name := "@@";  
                     Response : String := "<PROFILE>");
```

-- Computes the Library Units of the world that are installed or coded  
-- and reference any of the Link commands specified by the Source and  
-- Link parameters.

```
procedure Edit (World : World_Name := "<IMAGE>");  
procedure Visit (World : World_Name := "<IMAGE>");
```

-- Enters the links object editor. If there is no links window for the  
-- world to be edited, edit will create a new window, and visit will  
-- reuse an existing window of there is one.

```
procedure Insert (Source : Source_Name := ">>SOURCE NAME<<";  
                Source : Source_Name := ">>SOURCE NAME<<");
```

-- Insert and Update perform the same function as Add and Replace, but  
-- they must be run in a command window off a links image.

```

procedure Expunge (World : World_Name := "<IMAGE>";
  Response : String := "<PROFILE>");
pragma Subsystem (Commands);
pragma Module_Name (4, 3938);

```

```

end Links;

```

```

with Io;
with Diana;
with Directory;
with Error_Messages;
with Machine;
with Profile;
with Simple_Status;
package Log is

  subtype Name is String; -- an unambiguous string name

  procedure Set_Log (To_Be : Name := ">>FILE NAME<<";
    Filter : Profile.Log_Filter := Profile.Filter);
  -- Set Current_Output to To_Be, changing the profile to direct log
  -- output to Use_Current_Output. Change the Log_Filter to Filter.
  -- If To_Be cannot be created, Current_Output is not redirected, but
  -- no exception is raised.

  procedure Reset_Log (Filter : Profile.Log_Filter := Profile.Filter);
  -- Equivalent to IO.Reset_..., but changes Log_Filter

  procedure Put_System_Messages
    (Response : Profile.Response_Profile := Profile.Get);
  -- Copy contents of the message log for the current job into Current_Output

  procedure Put_Job_Messages
    (For_Job : Machine.Job_Id;
     Response : Profile.Response_Profile := Profile.Get);
  -- Copy contents of the message log for specified job into Current_Output

  procedure Put_Condition
    (Status : Simple_Status.Condition;
     Response : Profile.Response_Profile := Profile.Get);
  -- Display contents of Status in Current_Output.

  procedure Put_Line (Message : String;
    Kind : Profile.Msg_Kind := Profile.Note_Msg;
    Response : Profile.Response_Profile := Profile.Get);
  -- Appends the Message to the end of the Current_Output as described by
  -- the given response profile. If Profile.Includes (Kind, Response) is
  -- true, then the message is generated as described below; otherwise
  -- the Put_Line call returns immediately.

  -- The Time, Date and Symbol prefixes are printed first, in the order
  -- and format specified by the Profile.Prefixes (Response) array.
  -- If the Profile.Symbols prefix is requested, a unique three-character
  -- string is generated for each possible value of Kind:

  -- -- KIND -- Symbol -- Explanation
  -- Position_Msg >>>
  -- Identifies the location in a file or program
  -- to which subsequent messages refer.
  -- Sharp_Msg ###
  -- Dollar_Msg $$$ + Available for user-defined purposes
  -- At_Msg @@@ /
  -- Debug_Msg ???

```

```

-- Auxiliary_Msg :::
-- Note_Msg ---
-- Supplemental information.
-- Positive_Msg +++
-- Indicates that a major step in the process has
-- completed successfully. e.g. a unit has been
-- compiled, or generation of an output file is
-- complete.
-- Warning_Msg !!!
-- Indicates a minor problem in processing a major
-- step of the process. Warnings generally do not
-- lead to negative messages (see below).
-- Negative_Msg ---
-- Indicates that a major step in the process has
-- completed unsuccessfully. e.g. a unit has failed
-- to compile, or generation of an output file is
-- could not be accomplished.
-- Error_Msg +++
-- Indicates a significant problem within a major
-- step of the process that has been detected by
-- the command. Error messages will
-- frequently be followed by negative messages
-- Exception_Msg XXX
-- Indicates that a command caught an unexpected
-- exception.
--
-- The text of the message follows the prefixes. If the message line
-- exceeds Profile.Width (Response), it is continued on the next line.
-- Each continuation line starts with the same prefixes as the first
-- line, except that the three-character string "... " is used instead
-- of the symbols in the table above. (if no Symbols prefix is
-- requested by the Profile.Prefixes (Response), the symbol string
-- "... " is inserted between the rightmost prefix and the message text.)
--
procedure Copy (Log_File : Name := "<IMAGE>";
                Destination : Name := "";
                Filter : Profile.Log_Filter := Profile.Filter);
--
-- Once a log file has been generated with symbol prefixes, the
-- following procedures may be used to copy the file while filtering
-- out unwanted messages. The default destination is Current_Output

procedure Filter (Log_File : Name := "<IMAGE>";
                 Destination : Name := "";
                 Auxiliaries : Boolean := True;
                 Diagnostics : Boolean := True;
                 Notes : Boolean := True;
                 Positives : Boolean := True;
                 Negatives : Boolean := True;
                 Positions : Boolean := True;
                 Warnings : Boolean := True;
                 Errors : Boolean := True;
                 Exceptions : Boolean := True;
                 Sharps : Boolean := True;
                 Dollars : Boolean := True;
                 Ats : Boolean := True);
--
-- procedure Summarize (Log_File : Name := "<IMAGE>";
--                    Destination : Name := "";
--                    Auxiliaries : Boolean := True;

```

```

Diagnostics : Boolean := True;
Notes : Boolean := False;
Positives : Boolean := True;
Negatives : Boolean := True;
Positions : Boolean := False;
Warnings : Boolean := False;
Errors : Boolean := False;
Exceptions : Boolean := False;
Sharps : Boolean := False;
Dollars : Boolean := False;
Ats : Boolean := False) renames Filter;

procedure Filter_Errors (Log_File : Name := "<IMAGE>";
                        Destination : Name := "";
                        Auxiliaries : Boolean := True;
                        Diagnostics : Boolean := True;
                        Notes : Boolean := False;
                        Positives : Boolean := False;
                        Negatives : Boolean := True;
                        Positions : Boolean := False;
                        Warnings : Boolean := True;
                        Errors : Boolean := True;
                        Exceptions : Boolean := False;
                        Sharps : Boolean := False;
                        Dollars : Boolean := False;
                        Ats : Boolean := False) renames Filter;

procedure Set_Error (To_Bo : Name := ">>FILE NAME<<");
procedure Set_Input (To_Bo : Name := "<REGION>") renames Io.Set_Input;
procedure Set_Output (To_Bo : Name := ">>FILE NAME<<");
-- Set_Output and Set_Error deal with interaction with profiles that
-- direct log output to streams other than Current_Output.

procedure Pop_Error renames Io.Pop_Error;
procedure Pop_Input renames Io.Pop_Input;
procedure Pop_Output renames Io.Pop_Output;

procedure Reset_Error renames Io.Reset_Error;
procedure Reset_Input renames Io.Reset_Input;
procedure Reset_Output renames Io.Reset_Output;

procedure Flush (Response : Profile.Response_Profile := Profile.Get);
-- force any log output into the log file

procedure Save (Response : Profile.Response_Profile := Profile.Get);
-- make the current contents of the log file permanent; calls flush

generic
type Object_Type is private;
with function Full (Object : Object_Type) return String;
with function Is_Nil (Object : Object_Type) return Boolean;
with function Nil return Object_Type;
procedure Put_Line_Generic
(Object1 : Object_Type;
 Message : String := "";
 Object2 : Object_Type := Nil;
 Kind : Profile.Msg_Kind := Profile.Note_Msg;
 Response : Profile.Response_Profile := Profile.Get);

```

```

procedure Put_Line (Object1 : Directory.Object;
  Message : String := "";
  Object2 : Directory.Object := Directory.Nil;
  Kind : Profile.Msg_Kind := Profile.Note_Msg;
  Response : Profile.Response_Profile := Profile.Get);

procedure Put_Line (Object1 : Directory.Version;
  Message : String := "";
  Object2 : Directory.Version := Directory.Nil;
  Kind : Profile.Msg_Kind := Profile.Note_Msg;
  Response : Profile.Response_Profile := Profile.Get);

procedure Put_Line (Object1 : Diana.Tree;
  Message : String := "";
  Object2 : Diana.Tree := Diana.Empty;
  Kind : Profile.Msg_Kind := Profile.Note_Msg;
  Response : Profile.Response_Profile := Profile.Get);

-- Enters a message into the log, if messages of the Kind specified
-- are to be included.

-- If the message does go into the log, the name of the specified
-- object(s) is computed and inserted into the text of the message.
-- The location for the name of the first object is indicated by the
-- symbol "<1>"; if this string is not found in the message, the
-- name of the object is placed at the beginning of the message.
-- The location for the name of the object object is indicated by the
-- symbol "<2>"; if this string is not found in the message, the
-- name of the object, if not nil, is placed at the end of the message.

-- Directory.Naming.Unique_Full_Name is used to generate the name of
-- the object when the symbols given above are used or if no symbols
-- are found. The symbols "<1>>" and "<2>>" cause the value of
-- Directory.Naming.Get_Simple_Name to be used instead.

procedure Put_Errors (Errors : Error_Messages.Errors;
  Response : Profile.Response_Profile := Profile.Get);

-- Enter the Error messages into the log.

function Image (Kind : Profile.Msg_Kind) return String;

-- Returns the three-letter prefix used for the indicated Msg_Kind.

pragma Subsystem (Input_Output);
pragma Module_Name (4, 3218);
end Log;

```

```

package Message is

-- Write message in the message window of other user's sessions.
-- Send selects an individual user; Send_All sends to all logged in users.

procedure Send (Who : String; Message : String);
procedure Send_All (Message : String);

pragma Subsystem (Command);
pragma Module_Name (4, 2208);

end Message;

```

```

-- to not enable privileged mode unless it is really needed so
-- as to avoid accidentally doing something that would normally be
-- stopped by access control. All tasks in the job become
-- privileged when the mode is enabled. No output is produced
-- by any of these procedures. Failure to acquire privileged mode
-- is indicated only by the absence of the privileges. Privileged_Mode
-- returns false in this case.

procedure Enable_Terminal (Physical_Line : Terminal.Port;
  Response : String := "<PROFILE>");
procedure Disable_Terminal (Physical_Line : Terminal.Port;
  Response : String := "<PROFILE>");
-- (Dis)allow login on the specified terminal port
procedure Force_Logoff (Physical_Line : Terminal.Port;
  Commit_Buffers : Boolean := True;
  Response : String := "<PROFILE>");
-- Force a user off of the specified terminal.
-- Try to commit modified buffers if Commit_Buffers is true.
-- Each of these operations requires operator capability.

procedure Set_System_Time (To_Be : String := ">>TIME<<";
  Response : String := "<PROFILE>");
-- Requires operator capability.

procedure Shutdown_Warning (Interval : Duration := 3600.0);
-- Note that Interval is rounded to the nearest minute. Less than
-- 30.0 is rounded to 0.

function Get_Shutdown_Interval return Duration;

procedure Archive_On_Shutdown (On : Boolean := True);
function Get_Archive_On_Shutdown return Boolean;
-- Archive_On_Shutdown causes the next shutdown to store internal
-- state in "archive" form, allowing upgrades and conversion of
-- internal data structures. It typically takes several hours to
-- complete a shutdown or restart with archive conversions.

procedure Show_Shutdown_Settings;
procedure Cancel_Shutdown;

procedure Shutdown (Reason : String :=
  "COPS";
  Explanation : String := "Cause not entered");
-- Shutdown the machine. Enter the cause and explanation in the system
-- log, wait for the Shutdown interval to expire, then log users
-- off and shutdown the machine.
-- Enter Reason = "?" to get list of reasons. The shutdown will not
-- happen unless Reason is a legal value.

procedure Explain_Crash;
-- Reads a shutdown cause and explanation from current input and enters
-- these in the machine's error log. Corresponds to the information
-- entered by shutdown.

procedure Limit_Login (Sessions : Positive := Positive'Last);
procedure Show_Login_Limit;

```

```

with Terminal;
package Operator is
  procedure Disk_Space;
  procedure Create_User (User : String := ">>USER NAME<<";
    Password : String := "";
    Volume : Natural := 0;
    Response : String := "<PROFILE>");
  -- create a user with the given password on volume (0 => Most Available)
  procedure Delete_User (User : String := ">>USER NAME<<";
    Response : String := "<PROFILE>");
  -- delete user; Operator capability is required (or priv mode)
  procedure Change_Password (User : String := ">>USER NAME<<";
    Old_Password : String := "";
    New_Password : String := "";
    Response : String := "<PROFILE>");
  procedure Create_Session (User : String := ">>USER NAME<<";
    Session : String := ">>SESSION NAME<<";
    Response : String := "<PROFILE>");
  procedure Create_Group (Group : String := ">>GROUP NAME<<";
    Response : String := "<PROFILE>");
  -- Create the named group. It must currently not exist. It has
  -- no initial members.
  procedure Delete_Group (Group : String := ">>GROUP NAME<<";
    Response : String := "<PROFILE>");
  -- Delete the named group. This operation cannot be used to delete the
  -- group with the same name as an existent user. Delete_User will
  -- get rid of the group associated with a user. Acl entries
  -- that refer to a deleted group become inoperative and will be
  -- reclaimed during the next access list compaction.
  procedure Add_To_Group (User : String := ">>USER NAME<<";
    Group : String := ">>GROUP NAME<<";
    Response : String := "<PROFILE>");
  -- Add the specified user to the specified group.
  -- Operator privilege is required to execute this operation.
  procedure Remove_From_Group (User : String := ">>USER NAME<<";
    Group : String := ">>GROUP NAME<<";
    Response : String := "<PROFILE>");
  -- Remove the specified user to the specified group.
  -- Operator privilege is required to execute this operation.
  procedure Display_Group (Group : String := ">>GROUP NAME<<";
    Response : String := "<PROFILE>");
  -- Display the names of users in the specified group on Current_Output.
  procedure Enable_Privileges (Enable : Boolean := True);
  function Privileged_Mode return Boolean;
  -- If the caller is a member of the predefined group "privileged",
  -- calling this procedure actually enables or disables the
  -- extra capabilities that such a job can have. General usage is
  Operator, !Commands

```



```

function Get_Login_Limit return Positive;
-- Control over the number of simultaneously active user sessions

procedure Internal_System_Diagnosis;
-- Requires Operator capability

pragma Subsystem (Os_Commands);
pragma Module_Name (4, 3926);

end Operator;

```

```

with Machine;
with Simple_Status;

package Program is

    subtype Job_Id is Machine.Job_Id;
    subtype Condition is Simple_Status.Condition;

    procedure Run (S : String := "<SELECTION>";
                  Context : String := "q";
                  Response : String := "<PROFILE>");
    -- sets root of job_garbage_unit, dangerous to run concurrently in one job

    procedure Run_Job (S : String := "<SELECTION>";
                      Debug : Boolean := False;
                      Context : String := "q";
                      After : Duration := 0.0;
                      Options : String := "";
                      Response : String := "<PROFILE>");

    procedure Create_Job (S : String := "<SELECTION>";
                         Job : out Job_Id;
                         Status : in out Condition;
                         Debug : Boolean := False;
                         Context : String := "q";
                         After : Duration := 0.0;
                         Options : String := "";
                         Response : String := "<PROFILE>");

    -- Run_Job and Create_Job are identical except that Create_Job
    -- returns the job number of the job just started and a status indicating
    -- success or failure.

    -- Debug => True starts the debugger on the newly started job

    -- The following options are defined:

    -- Output          Specifies the name of the new job's output file.
    -- Input           New job's standard input file.
    -- Error           New job's error file.
    -- User            File names given are resolved in the directory
    --                 context of the caller, NOT the Context parameter.
    --                 Causes the new job to run with the identity
    --                 of this user. Password must be valid unless
    --                 running job is privileged. If not specified
    --                 new job runs with same identity as parent.
    -- Password       Password used in conjunction with User.
    -- Session        Session used in conjunction with User.

    function Started_Successfully (Status : Condition) return Boolean;
    -- True => Job has been started successfully

    procedure Wait_For (Job : Job_Id);
    -- Wait until the job specified has terminated.

Program, !Commands

```

```

procedure Change_Identity (To_User : String := "";
                          Password : String := "";
                          Options : String := "";
                          Status : in out Condition);
-- Change the identity of the calling job to the specified
-- user. Password must be supplied and correct unless the
-- caller is privileged. Options specifies additional
-- characteristics to be changed. If To_User is null,
-- the options are processed.
-- Note that only the access control identity is changed.
-- The actual username and session of the job are NOT changed.
-- This operation should never be used to change identity and
-- execute untrusted code. The identity can always be changed
-- back to the original job identity.
-- Options presently defined are:
-- Privileged -- enable privileged mode. The specified user
-- must be a member of group PRIVILEGED
-- Privileged => False -- disable privileged. No effect if caller
-- was not already privileged.
-- Restore_Identity -- Change the identity back to the original
-- identity of the job. Password is not
-- required to do this.
function Current (Subsystem : String := ">>SUBSYSTEM NAME<<";
                 Unit : String := ">>PROCEDURE NAME<<";
                 Parameters : String := "");
  Activity : String := "<ACTIVITY>" return String;
-- Constructs a procedure call suitable for Run or Run_Job that references
-- the appropriate view, has the appropriate quotes, etc. Unit name is
-- the Ada name to be called; it will be found anywhere in the
-- view. If the procedure being called has parameter they may be
-- provided. If the current view of !Subsystem is Rev8_4_0 and package
-- View is in the Commands directory, then:
-- Current ("!Subsystem", "View.Initial", "New_Tool") returns:
-- "!Subsystem.Rev8_4_0.Units.Commands".View.Initial ("New_Tool");
pragma Subsystem (Commands);
pragma Module_Name (4, 3930);
end Program;

```

```

with Directory;
package Queue is
  procedure Print (Name : String := "<IMAGE>";
                  Options : String := "<DEFAULT>";
                  Banner : String := "<DEFAULT>";
                  Header : String := "<DEFAULT>";
                  Footer : String := "<DEFAULT>");
  procedure Print_Version (The_Version : Directory.Version;
                           Options : String := "<DEFAULT>";
                           Banner : String := "<DEFAULT>";
                           Header : String := "<DEFAULT>";
                           Footer : String := "<DEFAULT>");
-- The Print and Print_Version procedures are the provided user interfaces
-- for sending files to a printer. They queue object(s) to be printed and
-- echo request IDs in the message window with corresponding objects.
-- NOTE : if a value is not specified for a parameter (<DEFAULT> is
-- indicated) then the value supplied in the session switch
-- file is used; if a session switch is not defined or
-- unavailable then the default specified here is used.
-- BANNER: String to be used on the banner page
-- (truncated at 11 characters), user's id is the default
-- Specifying the null string ("") will inhibit the generation
-- of a banner page.
-- HEADER: User supplied page header; default is none.
-- FOOTER: User supplied page footer; default is none.
-- (see R1000 documentation for headers or footers
-- containing Line-Feeds or exceeding Width characters)
-- OPTIONS: A form parameter for setting various formatting and
-- spooling options; default is "Format=>(Wrap, System_Header)".
-- The Currently available Options and semantic rules for these options are
-- described at the end of this package and in detail in the documentation.
-- procedure Cancel (Request_Id : Positive);
-- -- cancels a request by ID obtained from Print or Queue
-- -- Extreme measures for wedged spooler
-- procedure Kill_Print_Spooler;
-- procedure Restart_Print_Spooler;
-- -- The remaining procedures do NOT use any session switches.
  subtype Class_Name is String;
  All_Classes : constant Class_Name := "all";
  All_Spooler_Devices : constant String := "all";

```

Queue, !Commands

```

-- The following procedures provide information on the state
-- of the print spooler.

procedure Display (Class : Class_Name := "all");
-- print the current contents of the Queue

procedure Classes (Which : Class_Name := "all";
                  Show_Devices : Boolean := True);
-- Display information about one or all classes

procedure Devices (Which : String := "all";
                  Show_State : Boolean := True;
                  Show_Classes : Boolean := True);
-- Display information about one or all devices

-- The following procedures are used to define queues in the spooler.

procedure Create (Class : Class_Name := "");
procedure Destroy (Class : Class_Name := ""); Reroute : Class_Name := "";
-- Create/Destroy a class.
-- When a class is destroyed any requests in that class are rerouted to
-- the class specified (the default class if none is specified).

procedure Default (Class : Class_Name := "");
-- set Default Class or print current Default (if null string provided)

procedure Add (Device : String := ""; Options : String := "XON_XOFF");
-- Options :
-- XON_XOFF, RTS, DTR indicate what flow control is to be used.
-- Host => name indicates that a telnet connection is to be used.
-- If Host is given, Socket may be specified: Socket => (0, 23).

procedure Remove (Device : String := ""; Immediate : Boolean := False);
-- Associate/Disassociate a device with the print spooler.

procedure Register (Device : String := ""; Class : Class_Name := "");
procedure Unregister (Device : String := ""; Class : Class_Name := "");
-- Associates/disassociates a class and a device.
-- If a class is not associated with a device then items spooled to that
-- class can not be printed.

procedure Enable (Device : String := "all");
procedure Disable (Device : String := ""; Immediate : Boolean := False);
-- Allows/Disallows printing on device(s)

```

---

```

-- Description of the Options available for Print and Print_Version.
-- The following is a list of legal options.
-- BANNER_PAGE_USER_TEXT => text
-- Text appears on the banner page (if one is generated) after the
-- "Banner".
-- CLASS => class name

```

```

-- Class to which printout is to be queued. (default is <DEFAULT>)
-- COPIES => number
-- Number of copies of the printout (default is 1)
-- LENGTH => number
-- Number of printed lines available on a page (default is 60).
-- NOTIFY => Mail | MESSAGE | None
-- Type of notification desired upon completion of the print request.
-- ORIGINAL_RAW => true | FALSE
-- DO NOT make a copy of the file to be printed. Notification is set
-- to Message and each file is spooled separately with a banner page.
-- Class must NOT be Remote.
-- PostScript => ( <PostScript_Options> )
-- Specify to print using PostScript rules. PostScript options and
-- functionality are described below. The null options string, (),
-- invokes the PostScript printer with default parameters.
-- FORMAT => ( <Format_Options> )
-- The printer is to be treated as a conventional Ascii device with
-- the specified options, which are described below. FORMAT with the
-- null options string, (), is the default unless other options are
-- specified.
-- RAW => true | FALSE
-- DO NOT interpret the input. This option can be useful for
-- preformatted text or binary data.
-- SPOOL_EACH_ITEM => true | FALSE
-- Spool each file as a separate job.
-- Exactly one of the Format, Original_Raw, PostScript, or Raw can be supplied
-- for any print request. If any of these are specified in the Options
-- parameter, then the corresponding session switch is ignored.
-- <Format_Options>
-- The following is a list of legal <format_options>. Unless
-- otherwise specified, the Boolean options are assumed to be False.
-- NUMBERING => true | FALSE
-- Provide line numbering.
-- SYSTEM_HEADER => number
-- Produce a system page header on each page.
-- TAB_WIDTH => number
-- Number of spaces to replace a tab character (Ascii.HT) with
-- (default is 8). 0 causes tabs to be sent to the printer.

```

TRUNCATE => true | FALSE

Truncate lines longer than Width.

WIDTH => number

Number of characters to be printed on a line (default is 80).

WRAP => true | FALSE

Wrap lines longer than Width.

<PostScript\_Options>

FORMAT => PostScript | plain\_text | fancy | letter | image | AUTOMATIC

Broadly specifies how the file is to be printed, whether the file to be printed is a PostScript program (such as generated by a text formatter) or plain text that must be prepared for printing.

AUTOMATIC is the default, in which case the file is looked at to determine its type. If the file begins with a % it is processed as a PostScript program, if it begins with Ascii.Nul it is printed as an IMAGE, otherwise it is processed as PLAIN\_TEXT.

LETTER format is similar to PLAIN\_TEXT except that the defaults for TMOUP, BORDER, DATE, FILENAME, WRAP, and NUMBER are all False.

FANCY format is similar to PLAIN\_TEXT, except that Ada reserved words are emboldened and comments are italicized.

The following options apply to both PostScript and Plain\_Text files.

STATS => TRUE | false

Causes statistics on the size of files and their print speed to be included in job messages.

FLOW => true | FALSE

By default (FLOW=false), each file printed starts on a new sheet of paper. When FLOW is true, however, a file will start on the right half of a sheet if not occupied by the previous file. Setting FLOW to true forces TMOUP = true and REVERSED = false.

REVERSED => TRUE | false

If true, the default, the pages are reversed before printing so that the stack of pages in the printer's output tray are in the correct order with the first page on top. If false, the pages will be printed in the order they appear in the file.

CHATTY => TRUE | false

If true, the default, messages will be generated in the message window before accessing each file in the print request when false, PostScript issues a message only when all files have been printed

and under error conditions.

PAGES = <integer> [...<integer>]

Specifies the range of pages to be printed. The first page in the file is numbered 1. The default is to print all pages in the file. If only one integer is given, that one page is printed.

HEADER => true | FALSE

If true, a header page is printed that identifies the file that is being printed and the circumstances of its printing.

TMOUP => TRUE | false

If true, two file pages are printed per sheet of printer paper. The image of each page is 2/3 the size of a full page. The default for this option for plain text files is true; for PostScript files, it is false.

OUTLINES => TRUE | false

If true, a solid box is drawn around the text for each page. BORDER is an alternative name for this option. The default for this option for PLAIN\_TEXT files is true; for PostScript files, it is false.

DATE => true | false

If true, the time and date at the time of queuing is printed in the lower-left corner of each page, outside the outline box if present. The default for this option for plain text files is true; for PostScript files, it is false;

FILENAME => true | false

If true, the full name of the file is printed in the upper-left corner of each page, outside the outline box if present. The default for this option for plain text files is true; for PostScript files, it is false;

The following options apply to PLAIN\_TEXT files only. All combinations are valid.

NUMBER => TRUE | false

If true, a page number is printed in the upper right corner of each page, outside the outline box, if present. The numbering starts again at 1 for each file printed.

WIDE => true | FALSE

If true, each page is printed in landscape orientation, i.e., with the lines of text parallel to the longer side of the page.

RULES => true | FALSE

If true, faint dashed lines are drawn every other line of the output.

SIZE = <integer>  
SPACING = <integer>

Specifies the point-size of the typeface used to generate the output and the vertical spacing of each line measured in points. These point sizes determine the number of lines per page and the number of characters per line according to the following formulae:

For the WIDE format:

Lines/Page = 540 / Spacing  
Characters/Line = 1200 / Size

For the "WIDE (narrow)" format:

Lines/Page = 720 / Spacing  
Character/Line = 900 / Size

The default SPACING is SIZE + 1; The default SIZE is 11 (yielding a SPACING of 12). In "WIDE" format this allows for 60 lines of 81-character lines.

FONT = <font name>

Specifies the typeface to be used in printing the file. Any built-in PostScript font may be specified. The default is /Courier-Bold. If <font-name> begins with a '/', PostScript assumes the font is already resident and uses the <font name> to define the font to use. If <font name> does not begin with '/', PostScript assumes it is the name of a file containing PostScript for a downloadable font. This file is sent to the printer before any files are processed by PostScript. The simple name of the file, capitalized as it appears in the font option, is used to set the font for the plain\_text file.

CHOP => true | FALSE

If false, the default, a line longer than the line length defined by the above formulae is broken at the rightmost blank within the line and the extra text is printed on the next line justified to the right margin. If true, long input lines will be clipped at the boundaries of the imageable area (7.5 x 10.0 inches).

The following options affect the IMAGE format:

X = number  
Y = number

Specifies, in inches, the coordinates of the lower left corner of the first image. The default coordinate is (0.25, 0.25), a point 1/4 inch from the lower left corner of the paper.

DX = number  
DY = number

Specifies the offset from the previous image coordinate to the coordinate for the next image. Dx is added to the X coordinate for each successive image until the resulting coordinate would be outside the bounds of the paper, at which time X is reset to its original value and Dy is added to the Y coordinate. When the Y

coordinate exceeds the bounds of the paper, a new page is started at the original X, Y coordinate.

WIDTH = number  
HEIGHT = number

Specifies the maximum width and height allowed for the image. The default values specify a full page image.

DISTORT => true | FALSE

If true, the image will be magnified so that the image fills exactly the box defined by width and height. If false, the image will be magnified as large as possible while retaining the aspect ratio of the image.

ASPECT => number

Overrides the aspect ratio of the image.

CAPTION => text

Text to be rendered below the printed image.

PROLOG => text

EPILOG => text

PostScript code to be sent before and after each image. The following regards action taken on files when the PostScript option is specified and a list of legal <PostScript\_options>.

The following "commands" will be recognized when embedded in an input file when using a PostScript printer. These commands must begin in the first column of a line and must be capitalized as shown above.

XXINCLUDE naming-expression

Recognized in all formats except Image. Causes the files named in the expression to be opened and processed as if they were part of the input file. XXINCLUDEs can be nested to 10 deep.

XXASCII naming-expression

Recognized in PostScript format only. Causes the named files to be opened and sent to the destination without further interpretation by PostScript (nested commands are ignored).

XXBINARY naming-expression

Recognized in PostScript format only. Causes the named files to be opened and sent to the destination as strings of hexadecimal numbers. The XXBINARY command should be preceded by PostScript code that will prepare the printer to receive hexadecimal data.

pragma Subsystem (Os\_Commands);  
pragma Module\_Name (4, 3922);

end Queue;

```

with Machine;
package Scheduler is
  subtype Job_Id is Machine.Job_Id;
  subtype Cpu_Priority is Natural range 0 .. 6;
  subtype Milliseconds is Long_Integer;

  procedure Disable (Job : Job_Id);
  procedure Enable (Job : Job_Id);
  function Enabled (Job : Job_Id) return Boolean;

  function Get_Cpu_Priority (Job : Job_Id) return Cpu_Priority;
  type Job_Kind is (Co, Os, Attached, Detached, Server, Terminated);
  function Get_Job_Kind (Job : Job_Id) return Job_Kind;

  type Job_State is (Run, Wait, Idle, Disabled, Queued);
  function Get_Job_State (Job : Job_Id) return Job_State;

  -- returns the current state of job.
  RUN:      the job is currently runnable
  WAIT:    the job is runnable but being withheld by the scheduler.
  IDLE:    the job isn't using cpu time and has no unblocked tasks.
  DISABLED: an external agent has disabled the job from running.
  QUEUED:  the job is DETACHED and must wait for another to complete.

  function Get_Cpu_Time_Used (Job : Job_Id) return Milliseconds;

  -- returns the number of milliseconds of cpu time used by the job.
  -- belongs on the previous page, put here for compatibility reasons

  function Disk_Waits (Job : Job_Id) return Long_Integer;

  -- returns the number of disk_waits the job has done since last initialized

  function Working_Set_Size (Job : Job_Id) return Natural;

  -- returns the number of pages in the job's working set.

  subtype Load_Factor is Natural;

  -- for run queues, number of tasks * 100

  procedure Get_Run_Queue_Load (Last_Sample : out Load_Factor;
                               Last_Minute : out Load_Factor;
                               Last_5_Minutes : out Load_Factor;
                               Last_15_Minutes : out Load_Factor);

  -- number of runnable tasks * 100

  procedure Get_Disk_Wait_Load (Last_Sample : out Load_Factor;
                               Last_Minute : out Load_Factor;
                               Last_5_Minutes : out Load_Factor;
                               Last_15_Minutes : out Load_Factor);

  -- number of tasks waiting for a page on the disk wait queue * 100

  procedure Get_Withheld_Task_Load (Last_Sample : out Load_Factor;
                                    Last_Minute : out Load_Factor;

```

Scheduler, !Commands

```

Last_5_Minutes : out Load_Factor;
Last_15_Minutes : out Load_Factor);
-- returns the average number of tasks withheld from running by
-- the scheduler * 100. In this call LAST_SAMPLE is the number of tasks
-- held at the last 100ms scheduling cycle.

procedure State;

-- print scheduler state

procedure Display (Show_Parameters : Boolean := True;
                  Show_Queue : Boolean := True);
-- display current scheduler state and queues

type Job_Descriptor is
  record
    The_Cpu_Priority : Cpu_Priority;
    The_State : Job_State;
    The_Disk_Waits : Long_Integer;
    The_Time_Consumed : Milliseconds;
    The_Working_Set_Size : Natural;
    The_Making_Set_Limit : Natural;
    The_Milliseconds_Per_Second : Natural;
    The_Disk_Waits_Per_Second : Natural;
    The_Maps_To : Job_Id;
    The_Kind : Job_Kind;
    The_Made_Runnable : Long_Integer;
    The_Total_Runnable : Long_Integer;
    The_Made_Idle : Long_Integer;
    The_Made_Wait : Long_Integer;
    The_Wait_Disk_Total : Long_Integer;
    The_Wait_Memory_Total : Long_Integer;
    The_Wait_Cpu_Total : Long_Integer;
    The_Min_Working_Set_Limit : Long_Integer;
    The_Max_Working_Set_Limit : Long_Integer;
  end record;

function Get_Job_Descriptor (Job : Job_Id) return Job_Descriptor;

-- use to get a consistent snapshot of a job's statistics.

generic
  with procedure Put (Descriptor : Job_Descriptor);

procedure Traverse_Job_Descriptors (First, Last : Job_Id);

-- use to get a consistent, efficient snapshot of a range of
-- job's statistics.

procedure Set (Parameter : String := ""; Value : Integer);
function Get (Parameter : String) return Integer;

-- Programmatic versions of set and display
-- Initial parameters Default Units
-- CPU_Scheduling 1 1 or 0 (true or false)
-- Percent_For_Background 10 %
-- Min_Foreground_Budget -250 milliseconds (-5000..0)
-- Max_Foreground_Budget 250 milliseconds (0..5000)
-- Withhold_Run_Load 130 load * 100

```

```

-- Withhold_Multiple_Jobs          0      1 or 0 (true or false)
-- Memory_Scheduling              1      1 or 0 (true or false)
-- Environment_Msl                11000 pages
-- Min_Ce_Msl                    400 pages
-- Max_Ce_Msl                    1000 pages
-- Min_Oe_Msl                    250 pages
-- Max_Oe_Msl                    2000 pages
-- Min_Attached_Msl              50 pages
-- Max_Attached_Msl              2000 pages
-- Min_Detached_Msl              50 pages
-- Max_Detached_Msl              4000 pages
-- Min_Server_Msl                400 pages
-- Max_Server_Msl                1000 pages
-- Daemon_Msl                    200 pages
-- Msl_Decay_Factor              50 pages
-- Msl_Growth_Factor             50 pages
-- Min_Available_Memory          2048 pages
-- Page_Withdrawal_Rate          1      n*640 pages/sec (n in 0..64)

-- Disk_Scheduling                1      1 or 0 (true or false)
-- Max_Disk_Load                 250 Load_Factor
-- Min_Disk_Load                 200 Load_Factor

-- Foreground_Time_Limit         60 seconds
-- Background_Streams            3 minutes
-- Stream_Time_N                 2,5,20 jobs
-- Stream_Jobs_N                 3,0,0 jobs
-- Strict_Stream_Policy          0      1 or 0 (true or false)

procedure Set_Job_Attribute (Job : Job_Id;
Attribute : String := "Kind";
Value : String := "Server");

function Get_Job_Attribute
(Job : Job_Id; Attribute : String := "Kind") return String;

-- These interfaces exist to deal with ongoing changes to scheduler
-- characteristics without requiring new procedures.

-- The default parameters to Set_Job_Attributes make the indicated job
-- a server.

-- See the documentation for other attributes.

procedure Set_Msl_Limits (Job : Job_Id; Min, Max : Natural);
procedure Get_Msl_Limits (Job : Job_Id; Min, Max : out Natural);
procedure Use_Default_Msl_Limits (Job : Job_Id);

-- Each class of job has a default for working set min and max.
-- Set_Parameter lets you change the default value. Set_Msl_Limits lets
-- you override the default for a specific job. Use_Default_Msl_Limits
-- restores the values to the defaults, cancelling any prior Set_Msl_Limits
-- call.
-- Get_Msl_Limits returns the current values for a specific job.
-- Min and Max specify the range (in number of pages) in which the
-- working set limit is set. The scheduler chooses the working set
-- limit based on prevailing conditions on the machine. If Min and
-- Max are the same, the a fixed limit is specified.

```

```

-- Min must be less than or equal to Max and Max less than the memory size.
-- Error messages are sent to an output window in the case of errors.
-- No message of any kind if success.

```

```

pragma Subsystem (Os_Commands);
pragma Module_Name (4, 3923);

end Scheduler;

```

package Search\_List is

```
-- Conceptually a search list is a sequence of component names of
-- libraries. A component name could have wild characters, and would
-- therefore resolve to many libraries. The resolution of a name
-- depends on the resolution of the libraries, order being important.
-- Furthermore, the resolution of a component name or an Ada name
-- depends on the context in which such resolution is done. For
-- instance, the component name "q" meaning enclosing library resolves
-- to different libraries depending on the current context.
```

```
-- A separate image comes up for each Edit with different parameters.
-- Most commands take in defaulted Session and User parameters. The
-- defaults refer to the present user and session.
```

```
procedure Display (Session : String := ""; User : String := "");
```

```
-- Displays the Session Search List Components in a text-io image.
```

```
procedure Display_Libraries;
```

```
-- Displays the resolution of all the Libraries of the Search List
-- in the present context in a text-io image.
```

```
procedure Show_List (Session : String := ""; User : String := "");
```

```
-- Shows the Session Search List
```

```
procedure Show_Item (Component : String := "<CURSOR>");
```

```
-- Displays the library indicated by a Search List component provided it
-- resolves to a unique library. By default, displays the library at
-- the cursor.
```

```
procedure Set_Up (Component : String := ">>>SEARCH LIST<<";
                 Session : String := "";
                 User : String := "");
```

```
-- Initialize Search List. Replaces entire previous contents.
```

```
procedure Reset_To_System_Default
  (Session : String := ""; User : String := "");
```

```
-- Resets to system default search list.
```

```
procedure Add (Component : String := ">>>LIBRARY NAME<<";
              Position : Integer := Integer'Last;
              Session : String := "";
              User : String := "");
```

```
-- Adds Component in the indicated Position in the Search
-- List Components image. If defaulted, and cursor is on the
-- Search List image, then that is the location of the addition
-- Otherwise, addition is at end.
```

```
procedure Replace (New_Component : String := ">>>LIBRARY NAME<<";
                  Old_Component : String := "<SELECTION>";
                  Session : String := "";
                  User : String := "");
```

Search\_List, !Commands

```
-- Replace Old_Component (the component indicated by selection is the
-- default) by New_Component in Search List image.
```

```
procedure Delete (Component : String := "<SELECTION>";
                 Session : String := "";
                 User : String := "");
```

```
-- Remove Component (the component indicated by the current selection is
-- the default) from the Search List image.
```

```
procedure Release;
```

```
-- Removes current image from the screen
```

```
procedure Save (File_Name : String := ">>>FILE NAME<<";
               Session : String := "";
               User : String := "");
```

```
-- Save the search list of the given user's session
```

```
procedure Revert (File_Name : String := "";
                 Session : String := "";
                 User : String := "");
```

```
-- Revert the search list for the given user's session from the named
-- file. If the file name is defaulted, the search list is reverted
-- from the permanent search list maintained for this user's session
```

```
pragma Subsystem (Commands);
pragma Module_Name (4, 3939);
```

```
end Search_List;
```



```

package Switches is
-- This is the command-level interface to the Switch file facility
subtype File_Name is String;
-- An unambiguous Directory string name for a switch file or a
-- Directory or World. In the latter case, the file associated with
-- that Directory or World is implied.
Default_File : constant File_Name := "";
-- The default file is the selected object if it is a switch file,
-- otherwise it is the switch file associated with the current
-- enclosing library.
subtype Composite_Name is String;
-- an expanded Ada name whose prefix is a processor and whose simple
-- name is a switch of that processor. (If the switch name is unique,
-- the processor name can be omitted.)
-- "Semantics.Ignore_Minor_Errors", "Cg.Enable_Environment_Debugger"
subtype Value_Image is String;
-- Processor/Switch dependent. Will follow Ada conventions where
-- possible. E.g. the value images of Boolean valued switches are "true"
-- and "false"
subtype Specification is String;
-- A specification of the settings for selected switches in the form of
-- a sequence of Ada assignment statements. The lefthand side of the
-- assignment is the name of the switch and the righthand side is the
-- image of the value to be assigned to that switch.
-- e.g.,
-- "Ignore_Minor_Errors := true; Cg.Enable_Environment_debugger := false;"
procedure Define (File : File_Name := ">>SWITCH FILE<<";
Response : String := "<PROFILE>");
-- Creates an empty switch file with the given name. (File must not
-- denote an existing object.)
procedure Associate (File : File_Name := "<SELECTION>";
Library : String := "<IMAGE>";
Response : String := "<PROFILE>");
-- The specified File is associated with the given Library.
-- Association is by-reference. Any subsequent changes to the specified
-- File will be reflected immediately in the associated library.
function Associated (Library : String := "<IMAGE>") return File_Name;
-- Returns the name of the switch file associated with the given Library.
-- Returns the null string if no switch file has been associated.

```

```

procedure Set (Spec : Specification := ">>SWITCHES<<";
File : File_Name := "<SWITCH>";
Response : String := "<PROFILE>");
-- In the given switch file, the values of the switches named in the
-- specification are updated to the values in that spec.
procedure Display (Names : Composite_Name := "@@";
File : File_Name := "<SWITCH>";
Response : String := "<PROFILE>");
-- The switches in the given file whose names match the wildcard Names
-- specification are listed to the current output file.
procedure Edit (File : File_Name := "<SWITCH>");
-- Brings up a new Switch Display Window containing the contents of the
-- specified file. This window becomes the current Switch Display Window
procedure Visit (File : File_Name := "<SWITCH>");
-- Changes the current Switch Display Window to display the contents of
-- the specified switch File. The existing contents are committed
-- before the new file is displayed. A new Switch Display Window is
-- created if none have yet been created by the user.
procedure Insert (Spec : Specification := ">>SWITCHES<<");
-- The switch values displayed in the current Switch Display Window are
-- changed as indicated. (Generated in response to Object."1" on a
-- Switch Display Window)
procedure Change (Image : Value_Image := ">>SWITCH VALUE<<");
-- The highlighted switch in the current Switch Display Window is
-- changed to the value of the given image. (Generated in response to
-- Object."2" on a Switch Display Window.)
procedure Write (File : File_Name := ">>SWITCH FILE<<");
-- The contents of the Current Switch Display Window are copied to the
-- specified switch file.
procedure Create (File : File_Name := ">>SWITCH FILE<<";
Category : Character := 'L';
Response : String := "<PROFILE>");
-- Creates an empty switch file of the specified Category with the
-- given name. File should not exist. If it exists and is a File
-- object, a new, empty version will be created of the indicated
-- category.
Of_Session : constant File_Name := "<SESSION>";
-- Switch File_Name used to denote the switches associated with the
-- current session.
Of_Library : constant File_Name := "<SWITCH>";
-- Switch File_Name used to denote the switches associated with the

```

```

-- enclosing library.
procedure Edit_Session_Attributes;
-- Equivalent to Edit (Switches.Of_Session);
procedure Dissociate (Library : String := "<IMAGE>";
Response : String := "<PROFILE>");
-- Sever the association between the specified library and any switch
-- file.
pragma Subsystem (Commands);
pragma Module_Name (4, 3934);
end Switches;

```

```

package System_Backup is
subtype Id is Natural;
type Kind is (Full, Primary, Secondary);
-- Full backup is self-sufficient
-- Primary incremental is a differential from last Full backup
-- Secondary incremental is a differential from last Primary
procedure Backup (Variety : Kind := System_Backup.Full);
-- take a backup of kind Variety.
procedure History (Entry_Count : Positive := 10;
Full_Backups_Only : Boolean := False;
Tape_Information : Boolean := False);
-- print a list of Entry_Count previous backups. Full_Backups_Only
-- implies showing only Full backups. Tape_Information implies a list
-- of tapes involved in each.
pragma Subsystem (Os_Commands);
pragma Module_Name (4, 3924);
end System_Backup;

```

package Tape is

```
procedure Rewind (Drive : Natural := 0);
procedure Unload (Drive : Natural := 0);
procedure Read_Mt (Drive : Natural := 0);
procedure Write_Mt (File : String := "<SELECTION>";
  Indirect : Boolean := True;
  Drive : Natural := 0);
```

```
procedure Read (Volume : String;
  Directory : String := "#";
  Options : String := "R1000 Add_New_Line";
  To_Operator : String := "Thank You";
  Response : String := "<PROFILE>");
```

-- The specified volume is mounted and all files are read into the  
-- given directory.

-- Options are:

-- FORMAT = R1000 | MV | VAX/VMS

-- ADD\_NEW\_LINE

-- Add a line terminator following each record read from tape.  
-- Without this option, bytes are copied from tape without  
-- interpretation or modification.

-- Notes on mapping of tape names to R1000 file names

-- The file name from the tape is processed by replacing strings  
-- of non-alpha-numeric characters with a single '\_'. Then,  
-- if the name ends with an '\_', the character 'B' is appended  
-- to the name. If the name contains no alpha-numeric  
-- characters, a name derived from the user name and time is generated.

```
procedure Write (Files : String := "@@";
  Volume : String := "#";
  Options : String := "R1000 Text_Files";
  To_Operator : String := "Thank You";
  Response : String := "<PROFILE>");
```

-- The specified Volume is mounted and the specified files are  
-- written to the volume.

-- The To\_Operator string is displayed to the operator when the  
-- request to mount the tape is made.

-- Options are:

-- Text\_Files

If Text\_Files is specified, the file is assumed  
to contain only characters, line\_terminators,  
page\_terminators, etc. Each line of the file  
is written to a record on the tape. Lines are  
read according to the same rules as  
Text\_To\_Get\_Line.

-- Label an optional part of the label written

```
-- to the volume header.
-- Target system (no abbreviations):
--   R1000, MV, or VAX/VMS
--   [Default: R1000]
-- Record_Format: Ansi record format:
--   [Default: VARIABLE_LENGTH or SPANNED]
-- Record_Length: A positive integer. [Default: 512]
-- Block_Length: A positive integer. [Default: 2048]
--
-- The file name that goes on the tape is generated as follows:
--
-- First, if the object is an Ada Unit, then "V_" or "B_" are prepended
-- to the name if the unit is an Ada spec or body, respectively.
-- Then, '_' characters in the name are removed. One exception is
-- to this is that if the name ends in ".xyz", that underscore is
-- replaced with '.', yielding a filename that will end in ".xyz".
-- VAX/VMS bound file names are shortened to 9 characters; others
-- are shortened to 17 characters. If, after removing '_' characters,
-- the name is too long, vowels are removed starting at the right end
-- of the name (excluding the suffix). Then, if the name
-- is still too long, it is truncated (again, excluding the ".xyz"
-- suffix, if any).
--
-- Finally, to produce a unique name (with respect to others going on to
-- the tape), 'A' characters are inserted in front of the suffix, if any,
-- (preserving the ".xyz" suffix) and then these characters are
-- incremented alphabetically until the name is unique.
--
-- Thus, "AnInterestingName.Txt" becomes (if not VAX/VMS bound),
-- "AnInterestingNa.Txt"
--
Error : exception;
procedure Examine_Labels (Vol_Id : String := "";
  Vol_Set_Name : String := "";
  To_Operator : String := "Thank you";
  Volume_Labels_Only : Boolean := True);
--
procedure Format_Tape (Drive : Natural := 0; Vol_Id : String := "");
--
procedure Display_Tape (Drive : Natural := 0;
  Marks_To_Skip : Integer := 0;
  Records_To_Skip : Integer := 0;
  Blocks_To_Display : Natural := 10);
--
pragma Subsystem (Input_Output);
pragma Module_Name (4, 3927);
--
end Tape;
```

```

with Default;
with Machine;
with System;
with System_Utilities;

package Terminal is

    subtype Port is Natural range 0 .. 4 * 16 * 16;

    -- valid terminal types
    -- Rational, VT100, Facit
    -- valid terminal rates
    -- 75, 110,
    -- 134.5, 150, 200, 300,
    -- 600, 1200, 1800, 2400,
    -- 4800, 9600, 19200, EXT_REC_CLK

    subtype Stop_Bits_Range is System_Utilities.Stop_Bits_Range;
    subtype Character_Bits_Range is System_Utilities.Character_Bits_Range;
    subtype Parity_Kind is System_Utilities.Parity_Kind;
    -- None, Even, Odd

    function Current (S : Machine.Session_Id := Default.Session) return Port
    renames System_Utilities.Terminal;

    procedure Settings (Line : Port := Terminal.Current);
    -- print summary of current terminal

    procedure Set_Terminal_Type
    (Line : Port := Terminal.Current;
     To_Be : String := System_Utilities.Terminal_Type);

    procedure Set_Input_Rate (Line : Port := Terminal.Current;
                              To_Be : String := System_Utilities.Input_Rate);

    procedure Set_Output_Rate (Line : Port := Terminal.Current;
                               To_Be : String := System_Utilities.Output_Rate);

    procedure Set_Parity (Line : Port := Terminal.Current;
                          To_Be : Parity_Kind := System_Utilities.Parity);

    procedure Set_Stop_Bits (Line : Port := Terminal.Current;
                             To_Be : Stop_Bits_Range :=
                               System_Utilities.Stop_Bits);

    procedure Set_Character_Size (Line : Port := Terminal.Current;
                                  To_Be : Character_Bits_Range :=
                                    System_Utilities.Character_Size);

    procedure Set_Xon_Koff_Characters
    (Line : Port := Terminal.Current;
     Xon_Koff : String := System_Utilities.Xon_Koff_Characters);
    -- takes a 2-element string consisting of Xon followed by Xoff

    procedure Set_Xon_Koff_Bytes (Line : Port := Terminal.Current;
                                   Xon_Koff : System.Byte_String :=
                                     System_Utilities.Xon_Koff_Bytes);

```

```

procedure Set_Flow_Control
  (Line : Port := Terminal.Current;
   To_Be : String := System_Utilities.Flow_Control);

procedure Set_Receive_Xon_Koff_Characters
  (Line : Port := Terminal.Current;
   Xon_Koff : String := System_Utilities.
    Receive_Xon_Koff_Characters);

procedure Set_Receive_Xon_Koff_Bytes
  (Line : Port := Terminal.Current;
   Xon_Koff : System.Byte_String :=
    System_Utilities.Receive_Xon_Koff_Bytes);

procedure Set_Receive_Flow_Control
  (Line : Port := Terminal.Current;
   To_Be : String := System_Utilities.Receive_Flow_Control);

procedure Set_Disconnect_On_Disconnect
  (Line : Port := Terminal.Current;
   Enabled : Boolean := System_Utilities.
    Disconnect_On_Disconnect);

procedure Set_Logoff_On_Disconnect
  (Line : Port := Terminal.Current;
   Enabled : Boolean := System_Utilities.Logoff_On_Disconnect);

procedure Set_Disconnect_On_Logoff
  (Line : Port := Terminal.Current;
   Enabled : Boolean := System_Utilities.Disconnect_On_Logoff);

procedure Set_Disconnect_On_Failed_Login
  (Line : Port := Terminal.Current;
   Enabled : Boolean := System_Utilities.
    Disconnect_On_Failed_Login);

procedure Set_Log_Failed_Logins
  (Line : Port := Terminal.Current;
   Enabled : Boolean := System_Utilities.Log_Failed_Logins);

procedure Set_Login_Disabled
  (Line : Port := Terminal.Current;
   Disabled : Boolean := System_Utilities.Login_Disabled);

procedure Set_Detach_On_Disconnect
  (Line : Port := Terminal.Current;
   Enabled : Boolean := System_Utilities.Detach_On_Disconnect);

pragma Subsystem (Os_Commands);
pragma Module_Name (4, 3925);
end Terminal;

```

package Text is

```
type Image_Kind is (File, Input_Output);
procedure Create (Image_Name : String := ">>IMAGE_NAME<<";
                 Kind : Image_Kind := Text.File);
-- Create a text image.
-- Image_Kind = File a text file with the given name
-- Image_Kind = Input_Output creates an input_output image of that name
-- Commands run from an input_output image will have that image as the
-- default destination for Current_Output

procedure Block (All_Windows : Boolean := False);
procedure Continue (Page_Mode : Boolean := False;
                  All_Windows : Boolean := False);
procedure End_Of_Input;
procedure Redirect (To : String := ">>File Name<<");
-- Redirect the output associated with the current output
-- window to be redirected to the named file.
pragma Subsystem (Command);
pragma Module_Name (4, 2210);
end Text;
```

package What is

```
procedure Does (Name : String := "");
procedure Command (Clue : String := "");
procedure Line;
procedure Tabs;
procedure Message (File : String := "Daily_Message");
procedure Time;
procedure Load (Verbose : Boolean := True);
procedure Version;
procedure Users (All_Users : Boolean := True);
procedure Jobs (Interval : Positive := 10;
               User_Jobs_Only : Boolean := False;
               My_Jobs_Only : Boolean := False;
               Running_Jobs_Only : Boolean := True);
procedure Home_Library;
procedure Object (Name : String := "<IMAGE>");
procedure Locks (Name : String := "<IMAGE>");
pragma Subsystem (Command);
pragma Module_Name (4, 2217);
end What;
```

```

-- The "" Venture_Name argument is interpreted as "<CURSOR>".

procedure Set_Default (To_Work_Order : String := "<CURSOR>";
  For_Venture : String := "<VENTURE>";
  For_User : String := "<CURRENT_USER>";
  Response : String := "<PROFILE>");

function Default (For_Venture : String := "<VENTURE>";
  For_User : String := "<CURRENT_USER>") return String;

procedure Set_Notes (To_Value : String := ">>>New Notes<<";
  Order_Name : String := "<ORDER>";
  Response : String := "<PROFILE>");

-- The "" Order_Name argument is interpreted as "<CURSOR>".

function Notes (Order_Name : String := "<ORDER>") return String;

-- The "" Order_Name argument is interpreted as "<CURSOR>".

procedure Close (Order_Name : String := "<ORDER>";
  Response : String := "<PROFILE>");

-- The "" Order_Name argument is interpreted as "<CURSOR>".

procedure Display (Order_Name : String := "<ORDER>";
  Options : String := "";
  Response : String := "<PROFILE>");

-- Display the object by formatting and printing it. The "" argument
-- is interpreted as "<CURSOR>".
-- Valid Options are all of the session switches, plus "<DEFAULT>"
-- (which is the current session switch values), "<TERSE>" (the default),
-- and "<VERBOSE>".

procedure Edit (Order_Name : String := "<ORDER>");

-- Invoke the appropriate object_editor. The "" Argument is
-- interpreted as "<CURSOR>".

procedure Create (Order_Name : String := ">>>OBJECT NAME<<";
  Notes : String := "";
  On_List : String := "<WORK_LIST>";
  On_Venture : String := "<VENTURE>";
  Make_Default_Work_Order : Boolean := True;
  Response : String := "<PROFILE>");

-- Command line interface
-- "" for list is interpreted as Nil (Added to no list)

procedure Create_Field (Field_Name : String := ">>>FIELD NAME<<";
  Field_Type : String := ">>>BOOLEAN|STRING|INTEGER<<";
  Is_Vector : Boolean := False;
  Is_Controlled : Boolean := False;
  Default : String := "");

```

```

package Work_Order is

procedure Set_Default_Venture (To_Venture : String := "<CURSOR>";
  For_User : String := "<CURRENT_USER>";
  Response : String := "<PROFILE>");

function Default_Venture
  (For_User : String := "<CURRENT_USER>") return String;

procedure Set_Notes_Venture (To_Value : String := ">>>New Notes<<";
  Venture_Name : String := "<VENTURE>";
  Response : String := "<PROFILE>");

-- The "" Venture_Name is interpreted as "<CURSOR>".
-- "<VENTURE>" are interpreted as the default venture.

function Notes_Venture (Venture_Name : String := "<VENTURE>") return String;

-- The "" Venture_Name is interpreted as "<CURSOR>".

procedure Display_Venture (Venture_Name : String := "<VENTURE>";
  Options : String := "";
  Response : String := "<PROFILE>");

-- Display the object by formatting and printing it. The "" argument
-- is interpreted as "<CURSOR>".
-- Valid Options are all of the session switches, plus "<DEFAULT>"
-- (which is the current session switch values), "<TERSE>" (the default),
-- and "<VERBOSE>".

procedure Edit_Venture (Venture_Name : String := "<VENTURE>");

-- Invoke the appropriate object_editor. The "" Argument is
-- interpreted as "<CURSOR>".

procedure Create_Venture (Venture_Name : String := ">>>OBJECT NAME<<";
  Notes : String := "";
  Make_Default_Venture : Boolean := True;
  Response : String := "<PROFILE>");

-- Intended to be called from the command line

type Venture_Policy_Switch is
  (Require_Current_Work_Order, Require_Comments_At_Check_In,
  Require_Comment_Lines, Journal_Comment_Lines,
  Allow_Edit_Of_Work_Orders);

procedure Set_Venture_Policy (The_Switch : Venture_Policy_Switch;
  To_Value : Boolean;
  Venture_Name : String := "<VENTURE>";
  Effort_Only : Boolean := False;
  Response : String := "<PROFILE>");

-- Change a venture's policy switches.

```

```

Display_Position : Natural := 1;
On_Venture : String := "<VENTURE>";
Propagate : Boolean := True;
Response : String := "<PROFILE>";

-- Create a new user-defined field in a Venture.
-- Field_Name is the name given to the field.
-- Field_Type can be "Boolean", "String", or "Integer".
-- If Is_Vector is true, the field is declared equivalent to
-- Field_Name : array (Positive) of Field_Type.
-- If Is_Controlled is true, whether or not the field is modifiable
-- using the object editor is controlled by a policy switch.
-- Display_Position specifies the relative position of this field
-- in the object editor display as compared to all of the other
-- user defined fields. 0 means don't display.
-- Default is the image of the default value (all elements of
-- a vector have the same default). If no default is supplied,
-- False, "", or 0 will be assumed.
-- If Propagate is True, all existing work orders will be updated.

procedure Add_To_List (Order_Names : String := "IMAGE";
List_Name : String := "<WORK_LIST>";
Response : String := "<PROFILE>");

procedure Remove_From_List (Order_Names : String := "IMAGE";
List_Name : String := "<WORK_LIST>";
Response : String := "<PROFILE>");

procedure Set_Default_List (To_List : String := "<CURSOR>";
For_Venture : String := "<VENTURE>";
For_User : String := "<CURRENT_USER>";
Response : String := "<PROFILE>");

function Default_List (For_Venture : String := "<VENTURE>";
For_User : String := "<CURRENT_USER>") return String;

procedure Set_Notes_List (To_Value : String := ">>>New Notes<<";
List_Name : String := "<WORK_LIST>";
Response : String := "<PROFILE>");

-- The "" List_Name argument is interpreted as "<CURSOR>".

function Notes_List (List_Name : String := "<WORK_LIST>") return String;

-- The "" List_Name argument is interpreted as "<CURSOR>".

procedure Display_List (List_Name : String := "<WORK_LIST>";
Options : String := "";
Response : String := "<PROFILE>");

-- Display the object by formatting and printing it. The "" argument
-- is interpreted as "<CURSOR>".
-- "<WORK_LIST>" is the default list for the current user.
-- Valid Options are all of the session switches, plus "<DEFAULT>"
-- (which is the current session switch values), "<TERSE>" (the default),
-- and "<VERBOSE>".

```

```

procedure Edit_List (List_Name : String := "<WORK_LIST>");
-- Invoke the appropriate object_editor. The "" Argument is interpreted
-- as "<CURSOR>";

procedure Create_List (List_Name : String := ">>>OBJECT NAME<<";
Notes : String := "";
On_Venture : String := "<VENTURE>";
Make_Default_List : Boolean := True;
Response : String := "<PROFILE>");

package Venture_Editor is
procedure Set_Notes (Notes : String := ">>>New Notes<<");
procedure Set_Policy (To_Value : Boolean := False;
The_Switch : Venture_Policy_Switch);
procedure Spread_Fields (Interval : Natural := 10);
procedure Set_Field_Info (Is_Controlled : Boolean := False;
Display_Position : Natural := 1;
The_Field : String := ">>>Field Name<<");
procedure Set_Default_Order (New_Default : String := "<SELECTION>";
For_User : String := "<CURRENT_USER>");
procedure Set_Default_List (New_Default : String := "<SELECTION>";
For_User : String := "<CURRENT_USER>");
-- Command line procedures for modifying a Venture.
end Venture_Editor;

package Editor is
procedure Set_Notes (Notes : String := ">>>New Notes<<");
-- A command line procedure to change the Notes.

procedure Set_Field (To_Value : String := ">>>Field Value<<";
The_Index : Natural := 0;
The_Field : String := ">>>Field Name<<");

procedure Set_Field (To_Value : Integer := 0;
The_Index : Natural := 0;
The_Field : String := ">>>Field Name<<");

procedure Set_Field (To_Value : Boolean := False;
The_Index : Natural := 0;
The_Field : String := ">>>Field Name<<");

-- A command line procedure for changing a field in a Work_Order.
-- The_Index is ignored for scalar fields.

procedure Add_User (The_User : String := "<CURRENT_USER>");
procedure Add_Version (The_Configuration : String :=
">>>Configuration Name<<";
The_Element : String := ">>>Element Name<<";
The_Generation : Natural := 0);

procedure Add_Configuration
(The_Configuration : String := ">>>Configuration Name<<");
procedure Add_Comment (The_Comment : String := ">>>Comment<<";

```

```
The_Element : String := ">>Element Name<<";
The_User : String := "<CURRENT_USER>";

-- Command line procedures for augmenting a Work_Order.
end Editor;

package List_Editor is
  procedure Set_Notes (Notes : String := ">>New Notes<<");
  -- A command line procedure to change the Notes.
  procedure Add (Work_Orders : String := ">>Work Order Names<<");
  -- A command line procedure for adding to a Work_Order_List.
end List_Editor;

pragma Subsystem (Cmvc);
pragma Module_Name (4, 3781);

end Work_Order;
```



```

with System;
with Action;
with Directory;
with Io_Exceptions;

package Device_Independent_Io is

-- Device_Independent_Io is designed to provide a uniform method of
-- accessing sequential devices, including files, terminals, windows,
-- printers, and tape drives. Its clients are expected to be Text_IO and
-- Sequential_IO, though there may be others.

-- The assumption is made that devices deal in bytes or characters rather
-- than elemental types.

pragma Subsystem (Input_Output);
pragma Module_Name (4, 3208);

type File_Type is private;
type File_Mode is (In_File, Out_File);

subtype Class is Directory.Class;
subtype Subclass is Directory.Subclass;
subtype Version is Directory.Version;
subtype Byte is System.Byte;
subtype Byte_String is System.Byte_String;

procedure Open (File : in out File_Type;
               Mode : File_Mode;
               Name : String;
               Form : String := "");
with Class : Directory.Class := Directory.Nil;
Action_Id : Action.Id := Action.Null_Id;

procedure Open (File : in out File_Type;
               Mode : File_Mode;
               Object : Version;
               Form : String := "");
with Class : Directory.Class := Directory.Nil;
Action_Id : Action.Id := Action.Null_Id;

procedure Append (File : in out File_Type;
                 Name : String;
                 Form : String := "");
with Class : Directory.Class := Directory.Nil;
Action_Id : Action.Id := Action.Null_Id;

procedure Append (File : in out File_Type;
                 Object : Version;
                 Form : String := "");
with Class : Directory.Class := Directory.Nil;
Action_Id : Action.Id := Action.Null_Id;

-- open the object for output and position at end of file

procedure Create (File : in out File_Type;
                 Mode : File_Mode := Out_File;
                 Name : String := "";
                 Form : String := "");
with Class : Class := Directory.Nil;
with Subclass : Subclass := Directory.Nil;

```

```

Action_Id : Action.Id := Action.Null_Id;
-- creates the named object, if it currently does not exist
-- declaration of a new version is dependent on the class of the object
-- if object does not exist, and class is nil, file is assumed.

procedure Close (File : in out File_Type);
procedure Delete (File : in out File_Type);
procedure Reset (File : in out File_Type; Mode : File_Mode);
procedure Reset (File : in out File_Type);

procedure Save (File : File_Type; Immediate_Effect : Boolean := True);
-- Save the current contents of the file, but leave it open
-- Immediate_Effect => don't wait until the action is committed

function Mode (File : File_Type) return File_Mode;
function Name (File : File_Type) return File_Mode;
function Form (File : File_Type) return String;
function Get_Class (File : File_Type) return Class;
function Get_Subclass (File : File_Type) return Subclass;
function Get_Version (File : File_Type) return Version;
function Get_Action (File : File_Type) return Action.Id;

function Is_Open (File : File_Type) return Boolean;
function End_Of_File (File : File_Type) return Boolean;

procedure Read (File : File_Type;
               Item : out Byte_String;
               Count : out Natural);
procedure Read (File : File_Type; Item : out Byte);
procedure Read (File : File_Type; Item : out String; Count : out Natural);
procedure Read (File : File_Type; Item : out Character);

procedure Write (File : File_Type; Item : Byte_String);
procedure Write (File : File_Type; Item : Byte);
procedure Write (File : File_Type; Item : String);
procedure Write (File : File_Type; Item : Character);

function Is_Interactive (File : File_Type) return Boolean;
-- The user-visible function that determines whether or not a file
-- is interactive.

function Is_Empty (File : File_Type) return Boolean;
-- Determine if the file has any contents

generic
type Derived_File_Type is limited private;
-- Only works for types derived from Device_Independent_IO.File_Type
pragma Must_Be_Constrained (Derived_File_Type);
package File_Type_Conversions is
function From_Standard (File : File_Type) return Derived_File_Type;
function To_Standard (File : Derived_File_Type) return File_Type;
end File_Type_Conversions;

```

```

-- Ability to convert between Device_Independent_IO File_Type and those
-- used by its clients.
-- Provided principally to allow setting of options for device_specific

```

```
-- packages that may be used in conjunction with Text_IO, etc.
-- Specific clients may buffer data in ways not visible to
-- Device_Independent_IO, so this is a generally dangerous operation for
-- Input/Output operations.
```

```
generic
  type Element_Type is private;
package Type_Specific_Operations is
  function Read (File : File_Type) return Element_Type;
  procedure Read (File : File_Type; Item : out Element_Type);
  procedure Write (File : File_Type; Item : Element_Type);
end Type_Specific_Operations;
-- Type_Specific_Operations make it possible to implement the equivalent
-- of Sequential_IO or Polymorphic_Sequential_IO.
```

```
-- Exceptions
```

```
Status_Error : exception renames Io_Exceptions.Status_Error;
Mode_Error : exception renames Io_Exceptions.Mode_Error;
Name_Error : exception renames Io_Exceptions.Name_Error;
Use_Error : exception renames Io_Exceptions.Use_Error;
Device_Error : exception renames Io_Exceptions.Device_Error;
End_Error : exception renames Io_Exceptions.End_Error;
Data_Error : exception renames Io_Exceptions.Data_Error;
```

```
end Device_Independent_IO;
```

```
with Io_Exceptions;
with Device_Independent_IO;

generic
  type Element_Type is private;
  pragma Must_Be_Constrained (Element_Type);
package Direct_IO is
```

```
  pragma Subsystem (Input_Output);
  pragma Module_Name (4, 3203);
```

```
  type File_Type is limited private;
```

```
  type File_Mode is (In_File, Inout_File, Out_File);
```

```
  type Count is new Integer range 0 .. Integer'Last / Element_Type'Size;
  subtype Positive_Count is Count range 1 .. Count'Last;
```

```
-- File management
```

```
procedure Create (File : in out File_Type;
  Mode : File_Mode := Inout_File;
  Name : String := "";
  Form : String := "");
```

```
procedure Open (File : in out File_Type;
  Mode : File_Mode;
  Name : String;
  Form : String := "");
```

```
procedure Close (File : in out File_Type);
procedure Delete (File : in out File_Type);
procedure Reset (File : in out File_Type; Mode : File_Mode);
procedure Reset (File : in out File_Type);
```

```
function Mode (File : File_Type) return File_Mode;
function Name (File : File_Type) return String;
function Form (File : File_Type) return String;
```

```
function Is_Open (File : File_Type) return Boolean;
```

```
-- Input and output operations
```

```
procedure Read (File : File_Type;
  Item : out Element_Type;
  From : Positive_Count);
```

```
procedure Read (File : File_Type; Item : out Element_Type);
```

```
procedure Write (File : File_Type;
  Item : Element_Type;
  To : Positive_Count);
```

```
procedure Write (File : File_Type; Item : Element_Type);
```

```
procedure Set_Index (File : File_Type; To : Positive_Count);
```

```
function Index (File : File_Type) return Positive_Count;
```

```
Direct_IO, !Io
```

```

function Size (File : File_Type) return Count;
function End_Of_File (File : File_Type) return Boolean;
-- Exceptions
Status_Error : exception renames Io_Exceptions.Status_Error;
Mode_Error : exception renames Io_Exceptions.Mode_Error;
Name_Error : exception renames Io_Exceptions.Name_Error;
Use_Error : exception renames Io_Exceptions.Use_Error;
Device_Error : exception renames Io_Exceptions.Device_Error;
End_Error : exception renames Io_Exceptions.End_Error;
Data_Error : exception renames Io_Exceptions.Data_Error;
private
-- implementation dependent
type File_Descriptor;
type File_Type is access File_Descriptor;
pragma Segmented_Heap (File_Type);
end Direct_Io;

```

```

with Action;
with Device_Independent_Io;
with Directory;
with Io_Exceptions;
with Text_Io;
package Io is
pragma Subsystem (Input_Output, Private_Part => Closed);
pragma Module_Name (4, 3506);
type File_Type is private;
subtype File_Mode is Text_Io.File_Mode;
In_File : constant File_Mode := Text_Io.In_File;
Out_File : constant File_Mode := Text_Io.Out_File;
subtype Count is Text_Io.Count;
subtype Positive_Count is Count range 1 .. Count'Last;
Unbounded : constant Count := Text_Io.Unbounded;
subtype Field is Integer range 0 .. Integer'Last;
subtype Number_Base is Integer range 2 .. 16;
subtype Type_Set is Text_Io.Type_Set;
Upper_Case : constant Type_Set := Text_Io.Upper_Case;
Lower_Case : constant Type_Set := Text_Io.Lower_Case;
-- File Management
procedure Create (File : in out File_Type;
Mode : File_Mode := Out_File;
Name : String := "";
Form : String := "");
procedure Open (File : in out File_Type;
Mode : File_Mode := Out_File;
Name : String;
Form : String := "");
procedure Open (File : in out File_Type;
Mode : File_Mode;
Object : Directory.Version;
Form : String := "");
-- Open a particular directory version; not Text_IO
procedure Append
(File : in out File_Type; Name : String; Form : String := "");
procedure Append (File : in out File_Type;
Object : Directory.Version;
Form : String := "");
-- Open existing file for output, positioned at end of file; not Text_IO
-- Output starting after an Append is on a new line, but on the same page
-- as the previous end of the file
procedure Flush (File : File_Type);
-- Force any buffer characters out to file
procedure Save (File : File_Type);

```

```

-- Save the current contents of the file, but leave it open; calls Flush
procedure Close (File : in out File_Type);
procedure Delete (File : in out File_Type);
procedure Reset (File : in out File_Type; Mode : File_Mode);
procedure Reset (File : in out File_Type);

function Mode (File : File_Type) return File_Mode;
function Name (File : File_Type) return String;
function Form (File : File_Type) return String;

function Is_Open (File : File_Type) return Boolean;

-- Control of default input, output and error files; error not Text_IO
procedure Set_Input (File : File_Type);
procedure Set_Output (File : File_Type);
procedure Set_Error (File : File_Type);

-- Equivalent of an Open/Creates followed by above; not in Text_IO
procedure Set_Input (Name : String := "<SELECTION>");
procedure Set_Output (Name : String := ">FILE NAME<<");
procedure Set_Error (Name : String := ">>FILE NAME<<");

function Standard_Input return File_Type;
function Standard_Output return File_Type;
function Standard_Error return File_Type;
function Standard_Error return Text_IO.File_Type;

function Current_Input return File_Type;
function Current_Output return File_Type;
function Current_Error return File_Type;
function Current_Error return Text_IO.File_Type;

-- For each default files, f, Set_f pushes that File_Type entry on a stack
-- for the job. Pop_f removes the top of the stack. Reset is equivalent
-- to a Close and a Pop.
-- All open files in the default file stack at job termination are closed.

procedure Reset_Error;
procedure Reset_Input;
procedure Reset_Output;

procedure Pop_Error;
procedure Pop_Input;
procedure Pop_Output;

-- Specification of line and page lengths
procedure Set_Line_Length (File : File_Type; To : Count);
procedure Set_Line_Length (To : Count);

procedure Set_Page_Length (File : File_Type; To : Count);
procedure Set_Page_Length (To : Count);

function Line_Length (File : File_Type) return Count;
function Line_Length return Count;

```

```

function Page_Length (File : File_Type) return Count;
function Page_Length return Count;

-- Column, Line and Page Control
procedure New_Line (File : File_Type; Spacing : Positive_Count := 1);
procedure New_Line (Spacing : Positive_Count := 1);

procedure Skip_Line (File : File_Type; Spacing : Positive_Count := 1);
procedure Skip_Line (Spacing : Positive_Count := 1);

function End_Of_Line (File : File_Type) return Boolean;
function End_Of_Line return Boolean;

procedure New_Page (File : File_Type);
procedure New_Page;

procedure Skip_Page (File : File_Type);
procedure Skip_Page;

function End_Of_Page (File : File_Type) return Boolean;
function End_Of_Page return Boolean;

function End_Of_File (File : File_Type) return Boolean;
function End_Of_File return Boolean;

procedure Set_Col (File : File_Type; To : Positive_Count);
procedure Set_Col (To : Positive_Count);

procedure Set_Line (File : File_Type; To : Positive_Count);
procedure Set_Line (To : Positive_Count);

function Col (File : File_Type) return Positive_Count;
function Col return Positive_Count;

function Line (File : File_Type) return Positive_Count;
function Line return Positive_Count;

function Page (File : File_Type) return Positive_Count;
function Page return Positive_Count;

-- Character Input-Output
procedure Get (File : File_Type; Item : out Character);
procedure Get (Item : out Character);

procedure Put (File : File_Type; Item : Character);
procedure Put (Item : Character);
procedure Echo (Item : Character);

-- String Input-Output
procedure Get (File : File_Type; Item : out String);
procedure Get (Item : out String);

```

```

procedure Put (File : File_Type; Item : String);
procedure Put (Item : String);
procedure Echo (Item : String := "");

procedure Get_Line
  (File : File_Type; Item : out String; Last : out Natural);
procedure Get_Line (Item : out String; Last : out Natural);

procedure Put_Line (File : File_Type; Item : String);
procedure Put_Line (Item : String);
procedure Echo_Line (Item : String := "");

-- String Input-Output not in Text_IO

function Get_Line (File : File_Type) return String;
function Get_Line return String;

procedure Get (File : File_Type;
  Item : out String;
  Last : out Natural;
  End_Of_Line : out Boolean;
  End_Of_Page : out Boolean;
  End_Of_File : out Boolean);

-- Get all or part of a line.
-- End_Of_Line iff Item contains the end of line (possibly null)
-- End_Of_Page iff End_Of_Line and this is the last line of the page
-- End_Of_File iff End_Of_Page and this is the last page of the file
--
-- The intent is for each call to return as many characters from the
-- line as fit, but Last /= Item'Last doesn't imply End_Of_Line.

-- equivalents for instantiations of the type-specific generics

-- Integer Input-Output; equivalent to an instantiation of Integer_IO
procedure Get (File : File_Type; Item : out Integer; Width : Field := 0);
procedure Get (Item : out Integer; Width : Field := 0);

procedure Put (File : File_Type;
  Item : Integer;
  Width : Field := 0;
  Base : Number_Base := 10);

procedure Put (Item : Integer;
  Width : Field := 0;
  Base : Number_Base := 10);

procedure Echo (Item : Integer;
  Width : Field := 0;
  Base : Number_Base := 10);

-- Float Input-Output; equivalent to an instantiation of Float_IO
procedure Get (File : File_Type; Item : out Float; Width : Field := 0);
procedure Get (Item : out Float; Width : Field := 0);

procedure Put (File : File_Type;

```

```

  Item : Float;
  Fore : Field := 2;
  Aft : Field := 14;
  Exp : Field := 3);

procedure Put (Item : Float;
  Fore : Field := 2;
  Aft : Field := 14;
  Exp : Field := 3);

procedure Echo (Item : Float;
  Fore : Field := 2;
  Aft : Field := 14;
  Exp : Field := 3);

-- Boolean Input-Output; equivalent to an instantiation of Enumeration_IO

procedure Get (File : File_Type; Item : out Boolean);
procedure Get (Item : out Boolean);

procedure Put (File : File_Type; Item : Boolean; Width : Field := 0);
procedure Put (Item : Boolean; Width : Field := 0);

procedure Echo (Item : Boolean; Width : Field := 0);

-- Generic package for Input-Output of Integer Types
generic
  type Num is range <>;
  package Integer_IO is

    Default_Width : Field := Num'Width;
    Default_Base : Number_Base := 10;

    procedure Get (File : File_Type; Item : out Num; Width : Field := 0);
    procedure Get (Item : out Num; Width : Field := 0);

    procedure Put (File : File_Type;
      Item : Num;
      Width : Field := Default_Width;
      Base : Number_Base := Default_Base);

    procedure Put (Item : Num;
      Width : Field := Default_Width;
      Base : Number_Base := Default_Base);

    procedure Get (From : String; Item : out Num; Last : out Positive);
    procedure Put (To : out String;
      Item : Num;
      Base : Number_Base := Default_Base);

  end Integer_IO;

-- Generic package for Input-Output of Floating Point Types
generic
  type Num is digits <>;

```

```
package Float_Jo is
```

```
  Default_Fore : Field := 2;  
  Default_Aft : Field := Num'Digits - 1;  
  Default_Exp : Field := 3;
```

```
  procedure Get (File : File_Type; Item : out Num; Width : Field := 0);  
  procedure Get (Item : out Num; Width : Field := 0);
```

```
  procedure Put (File : File_Type;
```

```
    Item : Num;  
    Fore : Field := Default_Fore;  
    Aft : Field := Default_Aft;  
    Exp : Field := Default_Exp);
```

```
  procedure Put (Item : Num;  
    Fore : Field := Default_Fore;  
    Aft : Field := Default_Aft;  
    Exp : Field := Default_Exp);
```

```
  procedure Get (From : String; Item : out Num; Last : out Positive);
```

```
  procedure Put (To : out String;
```

```
    Item : Num;  
    Aft : Field := Default_Aft;  
    Exp : Field := Default_Exp);
```

```
end Float_Jo;
```

```
-- Generic package for Input-Output of Fixed Point Types
```

```
generic  
  type Num is delta <>;  
package Fixed_Jo is
```

```
  Default_Fore : Field := Num'Fore;  
  Default_Aft : Field := Num'Aft;  
  Default_Exp : Field := 0;
```

```
  procedure Get (File : File_Type; Item : out Num; Width : Field := 0);
```

```
  procedure Get (Item : out Num; Width : Field := 0);
```

```
  procedure Put (File : File_Type;
```

```
    Item : Num;  
    Fore : Field := Default_Fore;  
    Aft : Field := Default_Aft;  
    Exp : Field := Default_Exp);
```

```
  procedure Put (Item : Num;  
    Fore : Field := Default_Fore;  
    Aft : Field := Default_Aft;  
    Exp : Field := Default_Exp);
```

```
  procedure Get (From : String; Item : out Num; Last : out Positive);
```

```
  procedure Put (To : out String;
```

```
    Item : Num;  
    Aft : Field := Default_Aft;
```

```
end Fixed_Jo;
```

```
-- Generic package for Input-Output of Enumeration Types
```

```
generic  
  type Enum is (<>);  
package Enumeration_Jo is
```

```
  Default_Width : Field := 0;  
  Default_Setting : Type_Set := Upper_Case;
```

```
  procedure Get (File : File_Type; Item : out Enum);
```

```
  procedure Get (Item : out Enum);
```

```
  procedure Put (File : File_Type;
```

```
    Item : Enum;  
    Width : Field := Default_Width;  
    Set : Type_Set := Default_Setting);
```

```
  procedure Put (Item : Enum;
```

```
    Width : Field := Default_Width;  
    Set : Type_Set := Default_Setting);
```

```
  procedure Get (From : String; Item : out Enum; Last : out Positive);
```

```
  procedure Put (To : out String;
```

```
    Item : Enum;  
    Set : Type_Set := Default_Setting);
```

```
end Enumeration_Jo;
```

```
-- Interchange with other system file_types
```

```
-- Compatibility with Device_Independent_IO is solely for the purpose  
-- of allowing access to device-specific options at open.
```

```
-- Interchange of Get/Put and Read/Write operations between IO and  
-- Device_Independent_IO is undefined due to internal buffering in IO,  
-- though Flush can be used on output to clear the buffer.
```

```
-- Free interchange of operations with Text_IO is supported.
```

```
function Convert (File : File_Type) return Text_Io.File_Type;  
function Convert (File : Text_Io.File_Type) return File_Type;
```

```
function Convert (File : File_Type) return Device_Independent_Io.File_Type;  
function Convert (File : Device_Independent_Io.File_Type) return File_Type;
```

```
function "=" (L, R : File_Mode) return Boolean renames Text_Io."=";
```

```
function "=" (L, R : Type_Set) return Boolean renames Text_Io."=";
```

```
function "<" (L, R : Count) return Boolean renames Text_Io."<";
```

```
function "=" (L, R : Count) return Boolean renames Text_Io."=";
```

```
function ">" (L, R : Count) return Boolean renames Text_Io.">";
```

```
-- Operate on multiple input files matching a wildcard
```

```
generic
```

```
  with procedure Process (File : in out File_Type) is <>;
```

```

with procedure Note_Error (Message : String) is Io.Put_Line;
procedure Wildcard_Iterator (Names : String);
-- Calls Process once with an open File corresponding to each of Names
-- Name errors and unhandled exceptions are reported through Note_Error.
-- Closes File after each call to Process, if not already closed.

procedure Convert (From : Device_Independent_Io.File_Type;
                  To : in out Text_Io.File_Type);
-- Conversion form that allows changing limited private file_type

-- File management logical overloads. Each procedure duplicates one
-- above, but with direct control over the Action.ID used.
procedure Create (File : in out File_Type;
                 Mode : File_Mode := Out_File;
                 Name : String := "";
                 Form : String := "";
                 Action_Id : Action.Id);

procedure Open (File : in out File_Type;
               Mode : File_Mode := Out_File;
               Name : String;
               Form : String := "";
               Action_Id : Action.Id);

procedure Open (File : in out File_Type;
               Mode : File_Mode;
               Object : Directory.Version;
               Form : String := "";
               Action_Id : Action.Id);

procedure Append (File : in out File_Type;
                 Name : String;
                 Form : String := "";
                 Action_Id : Action.Id);

procedure Append (File : in out File_Type;
                 Object : Directory.Version;
                 Form : String := "";
                 Action_Id : Action.Id);

function Get_Action (File : File_Type) return Action.Id;

-- Exceptions

Status_Error : exception renames Io_Exceptions.Status_Error;
Mode_Error : exception renames Io_Exceptions.Mode_Error;
Name_Error : exception renames Io_Exceptions.Name_Error;
Use_Error : exception renames Io_Exceptions.Use_Error;
Device_Error : exception renames Io_Exceptions.Device_Error;
End_Error : exception renames Io_Exceptions.End_Error;
Data_Error : exception renames Io_Exceptions.Data_Error;
Layout_Error : exception renames Io_Exceptions.Layout_Error;

end Io;

```

```

package Io_Exceptions is

Status_Error : exception;
Mode_Error : exception;
Name_Error : exception;
Use_Error : exception;
Device_Error : exception;
End_Error : exception;
Data_Error : exception;
Layout_Error : exception;

pragma Exception_Name (Status_Error, 271);
pragma Exception_Name (Status_Error, 256); -- 256..271

pragma Exception_Name (Mode_Error, 287);
pragma Exception_Name (Mode_Error, 272); -- 272..287

pragma Exception_Name (Name_Error, 303);
pragma Exception_Name (Name_Error, 288); -- 288..303

pragma Exception_Name (Use_Error, 319);
pragma Exception_Name (Use_Error, 304); -- 304..319

pragma Exception_Name (Device_Error, 335);
pragma Exception_Name (Device_Error, 320); -- 320..335

pragma Exception_Name (End_Error, 351);
pragma Exception_Name (End_Error, 336); -- 336..351

pragma Exception_Name (Data_Error, 367);
pragma Exception_Name (Data_Error, 352); -- 352..367

pragma Exception_Name (Layout_Error, 383);
pragma Exception_Name (Layout_Error, 368); -- 368..383

pragma Subsystem (Miscellaneous);
pragma Module_Name (4, 804);

end Io_Exceptions;

```

```

with Action;
with Directory;
with Polymorphic_Io;

package Object_Set is

pragma Subsystem (Directory, Closed);
pragma Module_Name (4, 1721);

function Is_Set (Object : Directory.Object) return Boolean;

type Set is private;

procedure Create (Set_Name : String;
                 Set_Id : out Directory.Object;
                 Status : out Directory.Error_Status;
                 Action_Id : Action.Id := Action.Null_Id);

procedure Open (Set_Id : Directory.Object;
               The_Set : out Set;
               Status : out Directory.Error_Status;
               Action_Id : Action.Id := Action.Null_Id;
               For_Update : Boolean := False;
               Prevent_Create : Boolean := False);

procedure Close (The_Set : Set; Status : out Directory.Error_Status);

function Is_Empty (The_Set : Set) return Boolean;
procedure Make_Empty (The_Set : in out Set);

-- The Set must be open for update. (all sets are initially empty).

function Is_Member (The_Set : Set; Id : Directory.Object) return Boolean;

procedure Add (The_Set : in out Set; Id : Directory.Object);

procedure Remove (The_Set : in out Set; Id : Directory.Object);

type Iterator is limited private;

procedure Init (Iter : out Iterator; The_Set : Set);
procedure Next (Iter : in out Iterator);
function Value (Iter : Iterator) return Directory.Object;
function Done (Iter : Iterator) return Boolean;

function Handle_Of (H : Set) return Polymorphic_Io.Handle;

-- returns the Polymorphic Io handle on the open set.

end Object_Set;

```

Object\_Set, !Io

```

with Action;
with Directory;
with Io_Exceptions;
with System;

package Pipe is

subtype Action_Id is Action.Id;
Null_Action_Id : constant Action_Id := Action.Null_Id;
subtype Byte is System.Byte;
subtype Byte_String is System.Byte_String;
subtype Object_Id is Directory.Version;

subtype Operate_Status is Integer;

Status_Error : exception renames Io_Exceptions.Status_Error;
Mode_Error : exception renames Io_Exceptions.Mode_Error;
Name_Error : exception renames Io_Exceptions.Name_Error;
Use_Error : exception renames Io_Exceptions.Use_Error;
Device_Error : exception renames Io_Exceptions.Device_Error;
End_Error : exception renames Io_Exceptions.End_Error;
Data_Error : exception renames Io_Exceptions.Data_Error;
Layout_Error : exception renames Io_Exceptions.Layout_Error;

-- Exceptions are raised iff the the comments following the operation
-- indicate that the exception is possible. All other exception
-- propagation is considered a bug in the underlying implementation.

-- A pipe is an object which contains a queue of messages, possibly empty.
-- By opening the object for write, one can do "write" operations which
-- append messages to the end of the queue. By opening the object for
-- read, one can do "read" operations which consume messages from the
-- beginning of the queue. Each message is read by exactly one Read
-- operation; thus, in the face of concurrent Reads, each client may see
-- just a subset of the messages that were written to the pipe.

-- It is ok to make concurrent calls to this package. BUT, this does NOT
-- include calls which supply the same Handle; this is considered
-- erroneous. The implementation does not prevent such erroneous behavior;
-- this behavior might cause your Handle to be left inconsistent, but the
-- internal representation of the pipe itself is protected and will remain
-- consistent; thus, other Handle's (including those in other jobs) should
-- still work properly.

function Pipe_Class return Directory.Class;

type Handle is private;
Null_Handle : constant Handle;

-- Contains control information which is pertinent to a particular "open"
-- of a particular pipe. Other control information (about pipes) is kept
-- internally. Logically, a Handle is limited private. Use of multiple
-- copies is considered erroneous. The implementation does not prevent
-- such erroneous behavior. At worst, an erroneous program will be able
-- to Read/Write a pipe which is still open elsewhere, even though other
-- copies of the Handle have been closed; the internal representation of
-- the pipe itself is protected from such erroneous behavior and will
-- remain consistent.

```

Pipe, !Io



```

function Max_Buffer_Size return Positive;
-- Measured in bytes. Currently about half the maximum size of a heap.

function Default_Buffer_Size return Positive;
-- Measured in bytes. Currently about 20K bytes.

function Message_Overhead return Natural;
-- Measured in bytes. Currently about 8 bytes. Clients can compute the
-- value of  $B = n * (c + M)$ , where "c" is the result of this function, "M"
-- is the fixed size of messages supplied to the Write operation, "n" is
-- the desired capacity of the buffer (in messages), and "B" is the
-- resulting requirement for buffer size, in bytes. This function may
-- change between releases of the system.

type Pipe_Open_Mode is (Exclusive_Read, Shared_Read,
                        Exclusive_Write, Shared_Write, Exclusive);

-- The read modes allow one to use Read operations to consume messages from
-- the beginning of the queue. The write modes allow one to use Write
-- operations to append messages to the end of the queue. The same client
-- can use both Read and Write by opening the pipe multiple times. The
-- compatibility matrix is as follows:

-- Other action compatibility matrix:
-- Current Mode
-- (other actions)
-- ER SR EW SW E
-----
ER      X X
SR      X X X
EW      X X
SW      X X X
E
-----

-- Absence of an "X" indicates that the desired access will not be granted
-- if any OTHER action (not including requesting action) has the indicated
-- current access. Via Max_Wait, queuing is available when access is
-- denied for this reason.

-- Assuming access is not denied by the above rules, the following matrix
-- indicates the upgrade compatibility rules:

Upgrade matrix:
-- Current Access
-- (by same action)
-- ER SR EW SW E
-----
ER      ER ER E
SR      ER SR E
-- Desired
-----

```

```

-- Access
-- EW | EW E
-- SW | EW SW E
-- E | E E E E
-----
-- Absence of a mode indicates that the desired access will not be granted
-- if the requesting action already has the indicated current access.
-- Queuing is not available in this case. Presence of a mode indicates
-- that the access will be granted, and indicates the new lock mode in
-- which the action holds object.

-- Note that the upgrade rules imply that a single action cannot be used to
-- both read and write the same pipe. Of course, a single task can read
-- and write the same pipe by using 2 actions.

-- RESTRICTION: In Delta, the Create operation only supports Exclusive,
-- and the Open operation only supports Shared_Read, Shared_Write, and
-- Exclusive.

procedure Create (Pipe : in out Handle;
                 Mode : Pipe_Open_Mode;
                 Name : String;
                 Action : Action_Id := Null_Action_Id;
                 Max_Wait : Duration := Directory.Default_Wait;
                 Permanent_Contents : Boolean := False;
                 Buffer_Size : Positive := Default_Buffer_Size;
                 Reader_Buffer_Size : Natural := 0);

-- Since it's an object, a pipe lives in the directory system, as specified
-- by the Name parameter. Naming of pipes is the same as for vanilla
-- files. Multiple versions of a pipe are allowed.

-- With respect to disk space, the system reserves the right to allocate
-- disk space for the entire buffer, at any time, including the first
-- open. Thus, one should not use excessively large buffer sizes. With
-- respect to working set, under certain circumstances the buffer is used
-- in a cyclic fashion. Thus, a large buffer size may cause a large
-- working set. All in all, it's a good idea to use reasonable buffer
-- sizes. One rule of thumb is to use a buffer of size  $2 * N * E$ , where
-- "N" is the number of servers (readers), and "E" is the expected message
-- size; this tends to leave just enough room for writers to be "double
-- buffered". (Of course, the buffer must be large enough to hold the
-- biggest message.)

-- Given variable length messages, it is not possible for clients to
-- accurately predict the number of messages that can be stored by a buffer
-- of a particular size.

-- A pipe is made empty when it is last closed, or first opened if the
-- system crashes while the pipe is open. The only difference from the
-- user's point of view is disk space consumption.

-- The create operation leaves the pipe "open" in the specified mode.

-- Rules for Action are the same as for Open.

-- Abandoning the action may cause the object to disappear from the
-- directory system.

```

```
-- Abandoning/Committing the action causes all open handles (using the
-- action) to become closed.
--
-- If Null_Action_Id is supplied, a new action is created. If the Create
-- is successful, the new id is stored in the Handle, and committed when
-- the Handle is closed. If the Create fails, the new action is
-- abandoned.
```

```
-- Reader_Buffer_Size controls the operation of the Read function. If 0,
-- then it defaults to the result of calling Max_Buffer_Size.
```

```
-- KNOWN BUGS IN DELTA: (1) Abandoning the action which created the pipe
-- does NOT cause all open handles to become closed immediately. (2) The
-- implementation has a window in which concurrent opens may acquire the
-- object; this will cause the Create to return any of the exceptions that
-- can be returned by Open.
```

```
-- EXCEPTIONS:
```

```
-- Status_Error: The given Handle is already open
-- Mode_Error: Mode must be Exclusive
-- Name_Error: Directory wont create the object
-- Use_Error: Illegal buffer size, or lock error, or
-- access control violation
-- Device_Error: Obj Mgr can't set/get the buffer size;
-- and other internal errors
```

```
procedure Open (Pipe : in out Handle;
```

```
Mode : Pipe_Open_Mode;
Name : String;
Reader_Buffer_Size : Natural := 0;
Action : Action_Id := Null_Action_Id;
Max_Wait : Duration := Directory.Default_Wait);
```

```
-- Open an already existing pipe.
```

```
-- If the action is abandoned, the Handle may become implicitly closed.
```

```
-- Committing the action has no effect on the state of the pipe buffer.
```

```
-- If Null_Action_Id is supplied, a new action is created, its id stored
-- in the Handle, and the action is committed when the Handle is closed.
```

```
-- Reader_Buffer_Size controls the operation of the Read function. If 0,
-- then it defaults to the result of calling Max_Buffer_Size.
```

```
-- KNOWN BUG IN DELTA: Exclusive access shows up in the action_manager's
-- lock information as "Update"; all other access modes show up as "Read".
```

```
-- EXCEPTIONS:
```

```
-- Status_Error : The given Handle is already open.
-- Mode_Error: Mode must be Shared_Read, Shared_Write, or Exclusive
-- Name_Error: Directory can't find the object
-- Use_Error: Lock error, or access control violation
-- Device_Error: Obj mgr can't get the buffer size;
-- and other internal errors
```

```
procedure Open (Pipe : in out Handle;
```

```
Mode : Pipe_Open_Mode;
Object : Object_Id;
Reader_Buffer_Size : Natural := 0;
Action : Action_Id := Null_Action_Id;
Max_Wait : Duration := Directory.Default_Wait);
```

```
-- Same as above, but assumes that the caller has already resolved the
-- string name into an object id.
```

```
procedure Close (Pipe : in out Handle;
```

```
Max_Wait : Duration := Directory.Default_Wait);
```

```
-- If the pipe is open for writing, causes an implicit call to
-- Write_End_Of_File (throwing away a Use_Error caused by Max_Wait
-- induced timeout);
```

```
-- If the corresponding Create/Open supplied Null_Action_Id, then the
-- implicit action is either committed (when the Close is successful) or
-- abandoned (when the Close is unsuccessful).
```

```
-- The handle becomes closed.
```

```
-- EXCEPTIONS:
```

```
-- Status_Error: The given Handle is not open.
-- Device_Error: internal errors
```

```
procedure Delete (Pipe : in out Handle;
```

```
Max_Wait : Duration := Directory.Default_Wait);
```

```
-- Like all objects, causes it to be deleted. Must have the object open
-- for Exclusive access. Assuming a reasonable value for retention count,
-- the object can be "undeleted" using other environment operations.
```

```
-- If the corresponding Create/Open supplied Null_Action_Id, then the
-- implicit action is either committed (when the Delete is successful) or
-- abandoned (when the Delete is unsuccessful).
```

```
-- The handle becomes closed.
```

```
-- EXCEPTIONS:
```

```
-- Status_Error: The given Handle is not open
-- Name_Error: Directory returned an error other than Lock_Error or
-- access control error
-- Use_Error: Directory returned Lock_Error, which probably means
-- that Handle was not open for Exclusive access;
-- or could be an access control violation
-- Device_Error: internal errors
```

```
Dont_Wait : constant Duration := 0.0;
```

```
Forever : constant Duration := Duration'Last;
```

```
procedure Write (Pipe : in out Handle;
Message : Byte_String;
Max_Wait : Duration := Forever);
```

```
procedure Read (Pipe : in out Handle;
```

```

Message : out Byte_String;
Length : out Integer;
Max_Wait : Duration := Forever);

function Read (Pipe : Handle; Max_Wait : Duration := Forever)
return Byte_String;

-- These operations are "record (message) oriented". That is, the write
-- operation puts one record into the pipe which remembers the record and
-- its length. When successful, the read operation reads exactly one
-- record (when unsuccessful, it reads 0 records), the Length out parameter
-- indicates the actual length of the record (as given by the corresponding
-- Write operation).

-- This is in contrast with the Device_Independent_Io (Dio) Byte_String
-- operations which are "stream oriented". That is, the read operation
-- returns exactly the number of bytes that are requested, unless
-- end-of-file is reached, in which case fewer bytes are returned, as
-- indicated by the Length out parameter.

-- Given that pipes are record oriented, it is possible to write a program
-- which reads messages from a pipe, and copies them or sends them
-- somewhere else, without regard for the actual type of the data, and
-- preserving message boundaries.

-- The Read function is the same as the Read procedure except that it
-- internally allocates a Byte_String (of the length specified by the
-- Reader_Buffer_Size parameter of Create/Open) in which to read the
-- message, and then returns the first Length bytes. For variable length
-- messages, this frees the client (of this package) from needing to know
-- the maximum message size. In the current implementation, this
-- convenience is not free: the function makes an additional copy of the
-- message (as compared to the procedure), and it allocates
-- Reader_Buffer_Size - Length extra bytes in its stack frame. Of course,
-- if the function call site simply assigns the result into some variable,
-- there is an additional copy (as compared to the procedure).

-- Read and Write operations are atomic with respect to each other. BUT,
-- This DOES NOT include multiple tasks reading/writing with the same
-- Handle.

-- Messages are passed by value. That is, once Write completes, the entire
-- message is stored within the pipe. Termination of the client (which
-- performed the Write) does not effect the state of the pipe.

-- Recall that a pipe has finite internal buffer capability. A Write
-- operation which would exceed the maximum buffer size (defined at pipe
-- creation time) always raises Use_Error (and extended status
-- Item_Too_Big). A Write operation which would exceed the remaining
-- buffer capacity is handled as follows: If Max_Wait time expires before
-- sufficient room becomes available in the buffer (this is immediately
-- true if Max_Wait = Dont_Wait), then raises Use_Error (and extended
-- status No_Room_In_Buffer). When Use_Error is raised, the pipe is left
-- unmodified (except for overrun notification, as discussed below). The
-- client can distinguish between these flavors of Use_Error via the
-- Get_Extended_Status operation, below.

-- In the event that there are multiple clients waiting to do Write, they
-- are typically serviced FIFO in order to avoid starvation.

```

```

-- Similarly, a Read operation specifies the maximum amount of time to
-- wait for the buffer to become non-empty. A time of 0 indicates that the
-- client does not want to wait at all. If the wait time expires before a
-- message is received by the client, then the client gets Use_Error, and
-- the pipe is left unchanged.

-- The Read operation returns a single message. The Length parameter
-- indicates the actual number of bytes written into the Message parameter.
-- In the event that the actual message (supplied by the corresponding
-- Write operation) was longer than the Message parameter supplied to Read,
-- the client will receive Data_Error (and extended status of
-- Item_Too_Big), and the contents of the pipe are left unchanged. In
-- future implementations, negative values of Length may be defined.

-- Recall that each message is read by exactly one Read operation; thus, in
-- the face of concurrent Reads, each client may see just a subset of the
-- messages that were written to the pipe.

-- In the event that there are multiple clients waiting to do Read, they
-- are typically serviced LIFO. We assume that the application considers
-- all readers to be equivalent. In this context, LIFO is better than FIFO
-- because it minimizes the working set of the readers. (LIFO causes the
-- reader which most recently finished working to be the next to receive a
-- message). This simplifies applications which need to choose the number
-- of readers; they can simply pick the maximum number of readers which can
-- operate in parallel.

-- The implementation of Read and Write waiting can handle aborts of
-- clients.

-- Specifying infinite wait times allows one to use the finite buffer
-- capacity as a flow control mechanism.

-- "end of file" (eof) messages are written into a pipe via the
-- Write_EndOfFile operation, and implicitly via Close (which itself may
-- be implicit via action abandon, which itself may be implicit ...). When
-- a Read operation encounters an eof message, it is consumed, and
-- End_Error is raised. Unlike other sequential media, one can read an
-- eof only once.

-- "Overrun" refers to a situation in which the writer does not wait
-- forever for buffer space to become available and drops the unspent
-- messages on the floor. Pipes include the following mechanism for
-- detecting overrun:

-- In addition to messages of type data and eof, there are messages of type
-- overrun. A Write operation which raises Use_Error (because there is
-- insufficient room in the buffer) appends a message of type overrun.
-- Adjacent overrun messages are coalesced into a single overrun message.
-- The Read operation consumes the overrun message (when encountered) and
-- raises Use_Error. Like eof, an overrun message can only be read once.

-- Death of a client that has the pipe open for update may sometimes cause
-- an overrun to be placed in the pipe.

-- Observations: (1) The writer should probably not "poll" the pipe by
-- using a short Max_Wait, since each unsuccessful attempt will append an
-- overrun message, causing the reader to get a Use_Error. (2) The reader
-- can distinguish between timeout and overrun (both raise Use_Error) by
-- using the Extended_Status function, below.

```

```

-- EXCEPTIONS:
-- Status_Error : The given Handle is not open
-- Mode_Error   : Write: Handle was Open'd for Exclusive_Read
--               or Shared_Read
--               Read: Handle was Open'd for Exclusive_Write
--               or Shared_Write
-- Use_Error    : Write: Max_Wait expired,
--               or Message'length is larger than buffer size;
--               Read: Max_Wait expired,
--               or just consumed an overrun message
-- Data_Error   : Read: Message'length is shorter than next message
--               Read/Write: touching Message caused
--               Nonexistent_Page_Error
--               : Read: storing into Message caused
--               Write_To_Read_Only_Page
-- End_Error    : Read: just consumed an end-of-file message
-- Device_Error : internal errors
-- pragma Consume_Offset (4);

generic
type Element_Type is private;
package Type_Specific_Operations is
  procedure Write (Pipe : in out Handle;
                  Message : Element_Type;
                  Max_Wait : Duration := Forever);
  procedure Read (Pipe : in out Handle;
                  Message : out Element_Type;
                  Max_Wait : Duration := Forever);
  function Read (Pipe : Handle; Max_Wait : Duration := Forever)
  return Element_Type;
  pragma Consume_Offset;
end Type_Specific_Operations;
-- The usual "legal type for IO" rules apply to Element_Type. In
-- particular, Element_Type cannot be (or contain) pointers or tasks.
-- Both ends of the pipe should instantiate this package with the same
-- type, else one will get implicit unchecked conversions, and might
-- get Data_Error.
-- The generic Write operation first normalizes the Message, converts the
-- bits (of the Message) into a Byte_String (adding up to 7 bits of
-- padding, as necessary), and then calls the non-generic Write.
-- By normalize, we mean the following. For record types, if the object is
-- not constrained, allocate a constrained instance of the object and copy
-- the Message into the constrained copy. Note that this is expensive,
-- since it involves declaring collections and doing copies. For array
-- types, if the "bounds with object"ness of the Message is not the same as
-- that of the Element_Type (argument to the generic), then a copy is made
-- to convert the Message to the same boundedness as the Element_Type.
-- The generic Read procedure calls the non-generic Read procedure to fetch
-- the padded Byte_String, does an implicit unchecked conversion to
-- Element_Type, and assigns it to the out parameter.
-- The conversion may cause Data_Error to be raised when Element_Type is
-- not "compatible" with the actual bits in the message; this might happen
-- if the Write generic was instantiated with a different type than the
-- Read generic, for example. Some conditions that may cause
-- incompatibility: The 'size of the result of the unchecked conversion
-- (rounded to a byte) is not the same as the actual byte length of the
-- received message. The Element_Type is unconstrained and the message is
-- garbage (when interpreted according to Element_Type).
-- The assignment follows Ada semantics, and may therefore fail for a
-- variety of reasons, causing Data_Error. Some conditions that may cause
-- the assignment to fail: Element_Type is an unconstrained array type
-- (such as String), and the 'length of the string value in the buffer is
-- not the same as the 'length of the Message out parameter. The
-- Element_Type is unconstrained and the message is garbage (when
-- interpreted according to Element_Type).
-- The generic Read function calls the non-generic Read function, does an
-- implicit unchecked conversion to Element_Type, and returns the result.
-- Data_Error may be raised when Element_Type is not "compatible" with the
-- actual bits in the message, as for the Read procedure.
-- EXCEPTIONS (in addition to those raised by the non-generic forms):
-- Data_Error : Read: bits in the actual message are not
--               "compatible" with Element_Type, or := failed.
--               Pkg instantiation: raised when Element_Type has
--               task or access/heap_access components.
function End_Of_File (Pipe : Handle) return Boolean;
-- Returns true iff a read operation would have caused End_Error to be
-- raised.
EXCEPTIONS:
-- Status_Error : The given Handle is not open
-- Device_Error : Internal errors
procedure Write_End_Of_File (Pipe : in out Handle;
                             Max_Wait : Duration := Forever);
-- Puts an end-of-file message into the pipe. Note that Close (of a pipe
-- open for writing) may implicitly call this procedure. Abandoning
-- an action (of a writer) may implicitly call this procedure. With
-- respect to overruns, this call follows rules given for Write.
EXCEPTIONS:
-- Status_Error : The given Handle is not open
-- Use_Error    : Max_Wait expired,
-- Device_Error : Internal errors
function Current_Message_Count (Pipe : Handle) return Natural;

```

```

-- Can be used to "poll" a pipe to see how many messages are queued up,
-- waiting to be read.
-- EXCEPTIONS:
-- Status_Error : The given Handle is not open
function Max_Buffer_Size (Pipe : Handle) return Positive;
-- Return the buffer size of an open pipe.
-- EXCEPTIONS:
-- Status_Error : The given Handle is not open
function Open_Action (Pipe : Handle) return Action_Id;
-- Returns the action by which the Handle has the pipe open.
-- EXCEPTIONS:
-- Status_Error : The given Handle is not open
pragma Consume_Offset (3);
type Full_Status_Kinds is
(Pipe_Status, Directory_Error_Status,
Directory_Name_Status, Manager_Status, U4, U5, U6, U7);
function Get_Full_Status_Kind (Pipe : Handle) return Full_Status_Kinds;
-- Defined iff the Handle is currently open and the last PROCEDURE call on
-- the Handle raised an exception and the following table indicates that
-- additional status is available.
-- Status_Error no additional status
-- Mode_Error no additional status
-- Name_Error more status available
-- Use_Error more status available
-- Device_Error more status available
-- End_Error more status available
-- Data_Error more status available
-- Layout_Error no additional status In this case, indicates which
-- kind of additional status information is available about the exception.
type Extended_Status is (Internal_Pipe_Error, Item_Too_Big,
No_Room_In_Buffer, Buffer_Is_Empty,
Behind_Other_Readers, Read_An_Eof, Read_An_Overrun,
Missing_Page, Read_Only_Page, U09,
U10, U11, U12, U13, U14, U15, U16);
function Get_Extended_Status (Pipe : Handle) return Extended_Status;
function Get_Directory_Error_Status (Pipe : Handle) return Integer;
function Get_Directory_Name_Status (Pipe : Handle) return Integer;
function Get_Manager_Status (Pipe : Handle) return Operate_Status;
-- The above are defined iff Get_Full_Status_Kind is defined and returns
-- the corresponding value of Full_Status_Kinds. Rational reserves the
-- right to add additional Extended_Status values. Otherwise, it's ok
-- to program against Extended_Status values. The Integer values returned
-- by the last 3 functions are for debugging only, and may change between

```

```

-- between releases of this software.
function Status_Explanation (Pipe : Handle) return String;
-- Returns, in string form, the best explanation of the status that is
-- currently available. This explanation may include additional internal
-- state information. The returned string may differ between releases of
-- this software.
generic
with procedure Put_Line (S : String);
procedure Put_Pipe_Internal_State (Pipe : Handle;
Depth : Natural := 25;
Get_Locks : Boolean := False);
generic
with procedure Put_Line (S : String);
procedure Put_Internal_State (Depth : Natural := 25;
Get_Locks : Boolean := False);
-- These operations are primarily intended for use as debugging
-- aids by Rational personnel. However, it is also possible for customers
-- to use this information to debug their applications. The format of the
-- of the information fed through Put_Line may change in future releases.
-- The first operation gives you more information if the Handle is for an
-- open pipe! Depth is used to keep various algorithms from going into an
-- infinite loop when the internal data structures for the pipe are
-- inconsistent. Get_Locks indicates whether or not the internal data
-- structures should be viewed from within the appropriate critical
-- regions; in the current implementation, only the default is supported.
pragma Subsystem (Input_Output, Private_Part => Closed);
pragma Module_Name (4, 3223);
end Pipe;

```

```

with Action;
with Default;
with Directory;
with Io_Exceptions;
with System;

package Polymorphic_Io is

  pragma Subsystem (Directory);
  pragma Module_Name (4, 1706);

  -- Provides the basic file abstraction on top of the package directory
  -- and file object manager abstractions. Understanding actions is not
  -- necessary to use this level; parameters are always defaulted to be
  -- single, queued actions. Intended users are the Ada LRM Chapter 14
  -- IO packages, as well as sophisticated users that require more facilities
  -- than those provided in Chapter 14.

  subtype File is Directory.Object;

  subtype Version is Directory.Version;

  function Get_Class return Directory.Class;

  subtype Error_Status is Directory.Error_Status;

  type Handle is limited private;
  -- Handle that is needed to do anything to a file.

  function Nil return Handle;

  function Is_Nil (File : Handle) return Boolean;

  type File_Mode is (Read_Only, Write_Only, Read_Write, None);

  type File_Position is private;
  -- Logical file pointer that is needed to input and output operations.

  function Nil return File_Position;
  function First return File_Position;
  function Is_Nil (Position : File_Position) return Boolean;
  function Is_First (Position : File_Position) return Boolean;

  function "<" (Left, Right : File_Position) return Boolean;
  function "<=" (Left, Right : File_Position) return Boolean;
  function ">" (Left, Right : File_Position) return Boolean;
  function ">=" (Left, Right : File_Position) return Boolean;

  package Naming renames Directory.Naming;

  subtype Context is Directory.Naming.Context;

  function Default_Context
    (For_Job : Default.Process_Id := Default.Process) return Context
  renames Directory.Naming.Default_Context;

  procedure Open (The_Handle : in out Handle;
    Mode : File_Mode;
    File_Name : Naming.Name;

with Action;
with Default;
with Directory;
with Io_Exceptions;
with System;

package Polymorphic_Io is

  pragma Subsystem (Directory);
  pragma Module_Name (4, 1706);

  -- Provides the basic file abstraction on top of the package directory
  -- and file object manager abstractions. Understanding actions is not
  -- necessary to use this level; parameters are always defaulted to be
  -- single, queued actions. Intended users are the Ada LRM Chapter 14
  -- IO packages, as well as sophisticated users that require more facilities
  -- than those provided in Chapter 14.

  subtype File is Directory.Object;

  subtype Version is Directory.Version;

  function Get_Class return Directory.Class;

  subtype Error_Status is Directory.Error_Status;

  type Handle is limited private;
  -- Handle that is needed to do anything to a file.

  function Nil return Handle;

  function Is_Nil (File : Handle) return Boolean;

  type File_Mode is (Read_Only, Write_Only, Read_Write, None);

  type File_Position is private;
  -- Logical file pointer that is needed to input and output operations.

  function Nil return File_Position;
  function First return File_Position;
  function Is_Nil (Position : File_Position) return Boolean;
  function Is_First (Position : File_Position) return Boolean;

  function "<" (Left, Right : File_Position) return Boolean;
  function "<=" (Left, Right : File_Position) return Boolean;
  function ">" (Left, Right : File_Position) return Boolean;
  function ">=" (Left, Right : File_Position) return Boolean;

  package Naming renames Directory.Naming;

  subtype Context is Directory.Naming.Context;

  function Default_Context
    (For_Job : Default.Process_Id := Default.Process) return Context
  renames Directory.Naming.Default_Context;

  procedure Open (The_Handle : in out Handle;
    Mode : File_Mode;
    File_Name : Naming.Name;

```

Polymorphic\_Io, !Io

```

Status : out Error_Status;
The_Version : Directory.Version_Name :=
  Directory.Default_Version;
The_Context : Context := Polymorphic_Io.Default_Context;
Action_Id : Action.Id := Action.Null_Id;
Max_Wait : Duration := Directory.Default_Wait;
Prevent_Backup : Boolean := False);

procedure Open (The_Handle : in out Handle;
  Mode : File_Mode;
  The_Object : File;
  Status : out Error_Status;
  The_Version : Directory.Version_Name :=
  Directory.Default_Version;
  Action_Id : Action.Id := Action.Null_Id;
  Max_Wait : Duration := Directory.Default_Wait;
  Prevent_Backup : Boolean := False);

procedure Open
  (The_Handle : in out Handle;
  Mode : File_Mode;
  The_Version : in out Version;
  Status : out Error_Status;
  Action_Id : Action.Id := Action.Null_Id;
  Max_Wait : Duration := Directory.Default_Wait;
  Prevent_Backup : Boolean := False);

procedure Close (File : in out Handle; Status : out Error_Status);
-- Close a previously opened File.

procedure Delete (File : in out Handle; Status : out Error_Status);
-- Delete a previously opened File. File cannot have been opened for Read.
-- Commit any action opened on behalf of the user

function Is_Open (File : Handle) return Boolean;
function Mode (File : Handle) return File_Mode;
function Name (File : Handle) return Naming.Simple_Name;
function Full_Name (File : Handle) return Naming.Name;
-- Extract information about an open File.

function End_Of_File
  (File : Handle; Position : File_Position) return Boolean;
-- TRUE => Position is past end_of_File

function First_Free_Position (File : Handle) return File_Position;
-- Determine the first free (ie. non-existent) position within File.

function Size (File : Handle) return Long_Integer;
-- size of file in bits

generic
  type Element is private;
  -- Element must be constrained and "safe".
package Direct_Operations is

  function Compute (In_File : Handle;
    Index : Positive;
    Base : File_Position := Polymorphic_Io.First)
  return File_Position;

```

I-14

```

-- Determine the File_Position of the Index'th Element past Base.
function Read (From_File : Handle; At_Position : File_Position)
    return Element;
-- Yield the Element at the specified position in From_File.
-- If At_Position >= Free (From_File), END_ERROR is raised.
-- If the system can detect that no element has ever been
-- written At_Position, HOLE_ERROR is raised.

procedure Write (To_File : Handle;
                At_Position : File_Position;
                Value : Element);
-- Store the Value at the specified position in To_File.
-- If At_Position + Value'Size >= Free (To_File), To_File is
-- extended so that Free (To_File) = At_Position + Value'Size.

end Direct_Operations;

generic
type Element is private;
-- must be "safe"
package Sequential_Operations is

function Next (In_File : Handle; After : File_Position)
    return File_Position;
-- Move to the next Element in the specified file beyond After.
-- If After >= Free (In_File), END_ERROR is raised.

function Read (From_File : Handle; At_Position : File_Position)
    return Element;
-- Yield the Element at the specified position in From_File.
-- If At_Position >= Free (From_File), END_ERROR is raised.
-- If the system can detect that no element has ever been
-- written At_Position, HOLE_ERROR is raised.

procedure Write (To_File : Handle;
                At_Position : File_Position;
                Value : Element);
-- Store the Value at the specified position in To_File.
-- If At_Position + Value'Size >= Free (To_File), To_File is
-- extended so that Free (To_File) = At_Position + Value'Size.

end Sequential_Operations;

generic
type Element is private;
type Element_Pointer is access Element;
pragma Segment_Heap (Element_Pointer);
package Access_Operations is

function Reference (From_File : Handle; At_Position : File_Position)
    return Element_Pointer;
-- return a reference to the element "at_position"

function Position (From_File : Handle; Pointer : Element_Pointer)
    return File_Position;
-- return position of element referenced by Pointer.

end Access_Operations;

```

```

Status_Error : exception renames Io_Exceptions.Status_Error;
Mode_Error : exception renames Io_Exceptions.Mode_Error;
End_Error : exception renames Io_Exceptions.End_Error;
Data_Error : exception renames Io_Exceptions.Data_Error;

-----
-- CONVERSION OPERATIONS for FILE_POSITIONS --
-----
function Convert (Pos : File_Position) return Long_Integer;
function Convert (Pos : Long_Integer) return File_Position;

procedure Save (File : in out Handle;
               Status : out Error_Status;
               Immediate_Effect : Boolean := False);

function Get_Action (File : Handle) return Action.Id;

package String_Operations is
    subtype Byte is System.Byte;
    subtype Byte_String is System.Byte_String;

    procedure Read (File : Handle;
                  Pos : in out File_Position;
                  Item : out Byte_String;
                  Count : out Natural);

    procedure Read (File : Handle;
                  Pos : in out File_Position;
                  Item : out Byte);

    procedure Read (File : Handle;
                  Pos : in out File_Position;
                  Item : out String;
                  Count : out Natural);

    procedure Read (File : Handle;
                  Pos : in out File_Position;
                  Item : out Character);

    procedure Write (File : Handle;
                   Pos : in out File_Position;
                   Item : Byte_String);

    procedure Write
        (File : Handle; Pos : in out File_Position; Item : Byte);

    procedure Write
        (File : Handle; Pos : in out File_Position; Item : String);

    procedure Write
        (File : Handle; Pos : in out File_Position;
         Item : Character);

end String_Operations;

procedure Truncate (File : Handle;
                   Pos : File_Position := Polymorphic_Io.First);
-- Shortens the file so that Pos is the first position outside the file.
-- Will not make the file bigger if Pos is larger than the current size of
-- the file.

end Polymorphic_Io;

```

```

with Io_Exceptions;
with Device_Independent_Io;

package Polymorphic_Sequential_Io is
pragma Subsystem (Input_Output);
pragma Module_Name (4, 3210);

type File_Type is limited private;
type File_Mode is (In_File, Out_File);

-- File management

procedure Create (File : in out File_Type;
                 Mode : File_Mode := Out_File;
                 Name : String := "";
                 Form : String := "");

procedure Open (File : in out File_Type;
               Mode : File_Mode;
               Name : String;
               Form : String := "");

procedure Close (File : in out File_Type);
procedure Delete (File : in out File_Type);
procedure Reset (File : in out File_Type; Mode : File_Mode);
procedure Reset (File : in out File_Type);

function Mode (File : File_Type) return File_Mode;
function Name (File : File_Type) return String;
function Form (File : File_Type) return String;
function Is_Open (File : File_Type) return Boolean;

-- Input and output operations

generic
type Element_Type is private;
package Operations is
procedure Read (File : File_Type; Item : out Element_Type);
procedure Write (File : File_Type; Item : Element_Type);
end Operations;

function End_Of_File (File : File_Type) return Boolean;

procedure Append
  (File : in out File_Type; Name : String; Form : String := "");

-- Exceptions

Status_Error : exception renames Io_Exceptions.Status_Error;
Mode_Error : exception renames Io_Exceptions.Mode_Error;
Name_Error : exception renames Io_Exceptions.Name_Error;
Use_Error : exception renames Io_Exceptions.Use_Error;
Device_Error : exception renames Io_Exceptions.Device_Error;
End_Error : exception renames Io_Exceptions.End_Error;
Data_Error : exception renames Io_Exceptions.Data_Error;

end Polymorphic_Sequential_Io;

```

Polymorphic\_Sequential\_Io, !Io

```

with Io_Exceptions;
with Device_Independent_Io;

generic
type Element_Type is private;
package Sequential_Io is
pragma Subsystem (Input_Output);
pragma Module_Name (4, 3204);

type File_Type is limited private;
type File_Mode is (In_File, Out_File);

-- File management

procedure Create (File : in out File_Type;
                 Mode : File_Mode := Out_File;
                 Name : String := "";
                 Form : String := "");

procedure Open (File : in out File_Type;
               Mode : File_Mode;
               Name : String;
               Form : String := "");

procedure Close (File : in out File_Type);
procedure Delete (File : in out File_Type);
procedure Reset (File : in out File_Type; Mode : File_Mode);
procedure Reset (File : in out File_Type);

function Mode (File : File_Type) return File_Mode;
function Name (File : File_Type) return String;
function Form (File : File_Type) return String;
function Is_Open (File : File_Type) return Boolean;

-- Input and output operations

procedure Read (File : File_Type; Item : out Element_Type);
procedure Write (File : File_Type; Item : Element_Type);

function End_Of_File (File : File_Type) return Boolean;

-- Exceptions

Status_Error : exception renames Io_Exceptions.Status_Error;
Mode_Error : exception renames Io_Exceptions.Mode_Error;
Name_Error : exception renames Io_Exceptions.Name_Error;
Use_Error : exception renames Io_Exceptions.Use_Error;
Device_Error : exception renames Io_Exceptions.Device_Error;
End_Error : exception renames Io_Exceptions.End_Error;
Data_Error : exception renames Io_Exceptions.Data_Error;

private
type File_Type is new Device_Independent_Io.File_Type;
end Sequential_Io;

```

Sequential\_Io, !Io



```

with Device_Independent_!o;
with System;

package Tape_Specific is

  subtype File_Type is Device_Independent_!o.File_Type;
  subtype Byte_Range is Natural range 0 .. 4096;
  subtype Pipe_Range is Natural range 0 .. 8;
  subtype Byte_String is System.Byte_String;

  type On_Off is (On, Off);

  procedure Set_Block_Size (File : File_Type; Size : Byte_Range);
  -- default is Recommended_Max_Block_Length

  procedure Set_Streaming_Mode (File : File_Type; Mode : On_Off);
  -- on = true turns streaming mode on
  -- on = false turns streaming mode off
  -- default is off

  procedure Set_Pipeline_Size (File : File_Type; Size : Pipe_Range);
  -- pipeline size to use if in streaming mode
  -- default is Recommended_Pipeline_Size

  procedure Unload (File : File_Type);
  -- the "file" is closed
  -- the tape drive unloads the tape

  procedure Rewind (File : File_Type);
  -- the tape is put at beginning of tape

  type Skip_Records_Obstacles is
    (None,
     Tape_Mark,
     Bot
    );
  -- No obstacle encountered
  -- Tape mark encountered,
  -- Beginning of tape was encountered while
  -- while skipping backwards

  type Skip_Marks_Obstacles is
    (None,
     Double_Tape_Mark,
     Bot
    );
  -- No obstacle encountered
  -- 2 consecutive tape marks were encountered
  -- while skipping forward
  -- Beginning of tape was encountered while
  -- while skipping backwards

  type Error_Status is
    (Success,
     Record_Length_Long,
     Not_On_Line,
     Retry_Count_Exhausted,
     Unexpected_Tape_Error,
     Unit_Is_Bad
    );
  -- No error encountered
  -- Record on tape was longer than parameter
  -- Drive was offline
  -- Record/tape mark can't be read/written
  -- Tape position lost, rewind or unload it
  -- Call Field Service

  -- The following procedures that do not return an error status will have
  -- the exception DATA_ERROR raised if RECORD_LENGTH_LONG would have been
  -- returned. DEVICE_ERROR is raised for all other non-SUCCESS statuses.

  procedure Unload (File : File_Type; Status : out Error_Status);
  -- the "file" is closed
  -- the tape drive unloads the tape

  procedure Rewind (File : File_Type; Status : out Error_Status);
  -- the tape is put at beginning of tape

  -- The following should NOT be intermingled with the Read and Write
  -- procedures in Device_Independent_!o for the same file.

  -- The READ procedures return the next record of data on the tape.
  -- COUNT returns the actual size of the physical tape record in bytes.
  -- Only the first COUNT elements of RECD are valid.
  -- If RECORD_LENGTH_LONG is returned as the error status, then RECD
  -- contains the first RECD_LENGTH bytes of the physical tape record and
  -- COUNT = RECD_LENGTH. If a tape mark was read, then COUNT = 0.

  procedure Read (File : File_Type;
                 Recd : out Byte_String;
                 Count : out Natural);

  procedure Read (File : File_Type;
                 Recd : out Byte_String;
                 Count : out Natural;
                 Status : out Error_Status);

  procedure Read (File : File_Type; Recd : out String; Count : out Natural);

  procedure Read (File : File_Type;
                 Recd : out String;
                 Count : out Natural;
                 Status : out Error_Status);

  -- The two IS_MARK subprograms return whether the next tape record to be
  -- read is a tape mark. These subprograms should only be used while in
  -- streaming mode otherwise they will raise USE_ERROR. They will raise
  -- MODE_ERROR if the file is not open for input.

  function Is_Mark (File : File_Type) return Boolean;

  procedure Is_Mark (File : File_Type;
                   Result : out Boolean;
                   Status : out Error_Status);

  -- The WRITE procedures write the contents of RECD on the tape as a
  -- physical tape record. RECD_LENGTH must be greater than or equal to 18
  -- and less than or equal to the ABSOLUTE_MAX_BLOCK_LENGTH (currently
  -- 4096) otherwise USE_ERROR will be raised.

  -- PAST_EOT_MARKER indicates that the area beyond the reflective EOT marker
  -- on the tape is now being written. Users are cautioned that tape
  -- standards specify that there is at least 25 ft. of tape from the marker
  -- to the end of the reel, but only the first 10 ft. are useable. It
  -- is OK to write in this 10 ft. area but writing beyond that runs the
  -- risk of running the tape off its reel.

  procedure Write (File : File_Type;

```

```

with Device_Independent_!o;
with System;

package Tape_Specific is

  subtype File_Type is Device_Independent_!o.File_Type;
  subtype Byte_Range is Natural range 0 .. 4096;
  subtype Pipe_Range is Natural range 0 .. 8;
  subtype Byte_String is System.Byte_String;

  type On_Off is (On, Off);

  procedure Set_Block_Size (File : File_Type; Size : Byte_Range);
  -- default is Recommended_Max_Block_Length

  procedure Set_Streaming_Mode (File : File_Type; Mode : On_Off);
  -- on = true turns streaming mode on
  -- on = false turns streaming mode off
  -- default is off

  procedure Set_Pipeline_Size (File : File_Type; Size : Pipe_Range);
  -- pipeline size to use if in streaming mode
  -- default is Recommended_Pipeline_Size

  procedure Unload (File : File_Type);
  -- the "file" is closed
  -- the tape drive unloads the tape

  procedure Rewind (File : File_Type);
  -- the tape is put at beginning of tape

  type Skip_Records_Obstacles is
    (None,
     Tape_Mark,
     Bot
    );
  -- No obstacle encountered
  -- Tape mark encountered,
  -- Beginning of tape was encountered while
  -- while skipping backwards

  type Skip_Marks_Obstacles is
    (None,
     Double_Tape_Mark,
     Bot
    );
  -- No obstacle encountered
  -- 2 consecutive tape marks were encountered
  -- while skipping forward
  -- Beginning of tape was encountered while
  -- while skipping backwards

  type Error_Status is
    (Success,
     Record_Length_Long,
     Not_On_Line,
     Retry_Count_Exhausted,
     Unexpected_Tape_Error,
     Unit_Is_Bad
    );
  -- No error encountered
  -- Record on tape was longer than parameter
  -- Drive was offline
  -- Record/tape mark can't be read/written
  -- Tape position lost, rewind or unload it
  -- Call Field Service

  -- The following procedures that do not return an error status will have
  -- the exception DATA_ERROR raised if RECORD_LENGTH_LONG would have been
  -- returned. DEVICE_ERROR is raised for all other non-SUCCESS statuses.

  procedure Unload (File : File_Type; Status : out Error_Status);
  -- the "file" is closed
  -- the tape drive unloads the tape

  procedure Rewind (File : File_Type; Status : out Error_Status);
  -- the tape is put at beginning of tape

  type Skip_Records_Obstacles is
    (None,
     Tape_Mark,
     Bot
    );
  -- No obstacle encountered
  -- Tape mark encountered,
  -- Beginning of tape was encountered while
  -- while skipping backwards

  type Skip_Marks_Obstacles is
    (None,
     Double_Tape_Mark,
     Bot
    );
  -- No obstacle encountered
  -- 2 consecutive tape marks were encountered
  -- while skipping forward
  -- Beginning of tape was encountered while
  -- while skipping backwards

  type Error_Status is
    (Success,
     Record_Length_Long,
     Not_On_Line,
     Retry_Count_Exhausted,
     Unexpected_Tape_Error,
     Unit_Is_Bad
    );
  -- No error encountered
  -- Record on tape was longer than parameter
  -- Drive was offline
  -- Record/tape mark can't be read/written
  -- Tape position lost, rewind or unload it
  -- Call Field Service

  -- The following procedures that do not return an error status will have
  -- the exception DATA_ERROR raised if RECORD_LENGTH_LONG would have been
  -- returned. DEVICE_ERROR is raised for all other non-SUCCESS statuses.

  procedure Write (File : File_Type;

```

```

Recrd : Byte_String;
Past_Eot_Marker : out Boolean);

procedure Write (File : File_Type;
  Recrd : Byte_String;
  Past_Eot_Marker : out Boolean;
  Status : out Error_Status);

procedure Write (File : File_Type;
  Recrd : String;
  Past_Eot_Marker : out Boolean);

procedure Write (File : File_Type;
  Recrd : String;
  Past_Eot_Marker : out Boolean;
  Status : out Error_Status);

-- The WRITE_MARK procedures cause a tape mark to be written to the tape.
procedure Write_Mark (File : File_Type; Past_Eot_Marker : out Boolean);

procedure Write_Mark (File : File_Type;
  Past_Eot_Marker : out Boolean;
  Status : out Error_Status);

-- The SKIP_RECORDS procedures position the tape either forward
-- or backward until ABS (NUM_RECORDS_TO_SKIP) have been skipped or
-- an obstacle has been encountered. A positive NUM_RECORDS_TO_SKIP
-- implies skipping forward; negative implies skipping backward; zero
-- implies no movement. If a tape mark is encountered as an obstacle,
-- the position of the tape is on the "other side" of the tape mark; i.e.,
-- when skipping backward, the next item read would be that same tape mark
-- or when skipping forward, the next item read would be the record or
-- tape mark beyond the obstacle tape mark. The RECORDS_SKIPPED does
-- not include the tape mark. If the Beginning-Of-tape reflective marker
-- is encountered while skipping backward, the position of the tape will
-- be at the beginning of the tape, ready to read the first record.
-- MODE_ERROR is raised if the file is not open for reading. USE_ERROR
-- is raised if the file is in streaming mode.

procedure Skip_Records (File : File_Type;
  Num_Records_To_Skip : Integer;
  Obstacle : out Skip_Records_Obstacles;
  Records_Skipped : out Natural);

procedure Skip_Records (File : File_Type;
  Num_Records_To_Skip : Integer;
  Obstacle : out Skip_Records_Obstacles;
  Records_Skipped : out Natural;
  Status : out Error_Status);

-- The SKIP_TAPE_MARKS procedures position the tape either forward or
-- backward until ABS (NUM_MARKS_TO_SKIP) have been skipped or
-- an obstacle has been encountered. A positive NUM_MARKS_TO_SKIP
-- implies skipping forward; negative implies skipping backward; zero
-- implies no movement. Two consecutive tape marks (a double tape mark)
-- is only an obstacle while skipping forward; in which case neither of

```

```

-- the tape marks is counted in MARKS_SKIPPED. If two consecutive tape
-- marks are encountered while skipping backward, it is not an obstacle
-- and they are treated as individual tape marks in MARKS_SKIPPED. If
-- the Beginning-Of-tape reflective marker is encountered while skipping
-- backward, the position of the tape will be at the beginning of the
-- tape, ready to read the first record. If no obstacle was encountered,
-- the position of the tape is on the "other side" of the last tape mark.
-- MODE_ERROR is raised if the file is not open for reading. USE_ERROR
-- is raised if the file is in streaming mode.

```

```

procedure Skip_Tape_Marks (File : File_Type;
  Num_Marks_To_Skip : Integer;
  Obstacle : out Skip_Marks_Obstacles;
  Marks_Skipped : out Natural);

```

```

procedure Skip_Tape_Marks (File : File_Type;
  Num_Marks_To_Skip : Integer;
  Obstacle : out Skip_Marks_Obstacles;
  Marks_Skipped : out Natural;
  Status : out Error_Status);

```

```

pragma Subsystem (Input_Output);
pragma Module_Name (4, 3214);

```

```

end Tape_Specific;

```

```

with Device_Independent_Io;
with System;
package Terminal_Specific is

-- This package supports operations that are specific to
-- "terminals". For this purpose, a terminal is an object of
-- type Terminal. These objects are in IMachine.Devices.
--
-- Normal Text_IO-style IO to the terminal is done through:
--   USERS.user.session Standard_Output
--   IMACHINE.USERS.user Standard_Error
--
-- Window_IO provides quarter-plane, addressable display and key input
--
-- Access to the terminal for Standard_Output, Standard_Error and
-- Window_IO is handled by the job controlling the session, so there may
-- be multiple, simultaneously-active windows.
--
-- Opening a terminal directly provides the application complete control
-- of the terminal. In this case, the terminal is controlled by the job
-- that opens it, NOT the session job.
--
-- More than one job can have a terminal open at the same time, but
-- only one of them will actually receive input or transmit output.
-- Any others will be blocked on both input and output. A job that
-- references the terminal directly and simultaneously uses one of the
-- session-controlled forms of terminal interaction will not work well and
-- may deadlock.
--
-- Attempts to open an enabled terminal other than the one for
-- current session will fail. Control over enabled/disabled status
-- of terminals is available in the Operator package.
--
-- The determination of which of the various jobs dealing with the
-- terminal actually have the right to transmit/receive is done on
-- the basis of which job is "connected". There is at most one
-- connected job at any time. If a job that has the terminal open
-- is currently connected, it has the terminal. If it disconnects
-- or is terminated, control of the terminal reverts to the session.
-- The user can return control of the terminal to the application
-- by doing a Job.Connect with the appropriate job number.
--
-- Transfers of terminal ownership are detectable as part of the
-- status of the Read and Write operations. This allows
-- applications that support disconnect to detect when to redraw
-- their version of the screen.
--
-- The following device-specific Form options are supported:
--
-- Option      Explanation      Default
-- Echo        whether to echo input      True
-- Edit        Line editing or None    Line
-- CRLF        map LF to CRLF          True
--
-- Note: the CRLF option is ignored by the Write procedures in this
-- package to reduce confusion over whether the CR was transferred for a
-- particular count. CRLF is honored by device_independent write/put

```

Terminal\_Specific, IIO

```

-- operations.
--
subtype File_Type is Device_Independent_Io.File_Type;
subtype Byte_String is Device_Independent_Io.Byte_String;

package Status is
type Code is new Integer;
function Image (Value : Code) return String;

-- Status.Code      Standard reaction
-- Normal           none
-- Break            raise End_Error
-- Disconnect       raise End_Error
-- Timed_Out        0 data bytes transferred
-- Data_Error       raise Data_Error
-- Data_Overrun     raise Device_Error
-- Lost_Ownership   ignored
-- Gained_Ownership ignored
-- Too_Many_Clients raise Device_Error
--
-- Standard Read/Write routines in Device_Independent_IO use:
-- Duration'Last for Wait parameters
-- "Standard Reaction" for Result parameters
--
Normal : constant Code := 0;
Break : constant Code := 1;
Disconnect : constant Code := 2;
Not_Open : constant Code := 3;
Timed_Out : constant Code := 4;
Data_Error : constant Code := 5;
Data_Overrun : constant Code := 6;
Lost_Ownership : constant Code := 7;
Gained_Ownership : constant Code := 8;
Too_Many_Clients : constant Code := 9;
end Status;

package Output is
procedure Map_Lf_To_CrLf (File : File_Type; Value : Boolean := True);
-- Equivalent of CRLF Form option; default True

procedure Transmit_Break (File : File_Type);
procedure Transmit_Break (File : File_Type;
Wait : Duration;
Result : out Status.Code);

procedure Disconnect (File : File_Type);
procedure Disconnect (File : File_Type;
Wait : Duration;
Result : out Status.Code);

procedure Wait_For_Transmission (File : File_Type);
-- Wait for all previously written data to be transmitted.

```

```

procedure Set_Rts (File : File_Type; On : Boolean);
-- Set the current state of the RTS (pin 4) RS-232
-- modem control output. True => ON, False => OFF.

procedure Set_Dtr (File : File_Type; On : Boolean);
-- Set the current state of the DTR (pin 20) RS-232
-- modem control output. True => ON, False => OFF.
end Output;

package Input is
    procedure Flush (File : File_Type);

    procedure Set_Echo (File : File_Type; Echo : Boolean := True);
    -- Equivalent to Echo Form option; default True

    function Get_Echo (File : File_Type) return Boolean;

    procedure Set_Editing (File : File_Type; Mode : String := "Line");
    -- Equivalent to the Edit Form option; default Edit => Line
    -- Disabled with value None.

    function Get_Editing (File : File_Type) return String;
end Input;

procedure Read (File : File_Type;
    Item : out Byte_String;
    Count : out Natural;
    Wait : Duration);

procedure Read (File : File_Type;
    Item : out String;
    Count : out Natural;
    Wait : Duration);

procedure Read (File : File_Type;
    Item : out Byte_String;
    Count : out Natural;
    Wait : Duration;
    Result : out Status.Code);

procedure Write (File : File_Type;
    Item : Byte_String;
    Count : out Natural;
    Wait : Duration);

procedure Write (File : File_Type;
    Item : String;
    Count : out Natural;
    Wait : Duration);

procedure Write (File : File_Type;
    Item : File_Type;
    Count : out Natural;
    Wait : Duration;
    Result : out Status.Code);

```

```

pragma Subsystem (Input_Output);
pragma Module_Name (4, 3215);
end Terminal_Specific;

```

```

with Io_Exceptions;
package Text_Io is
  pragma Subsystem (Input_Output, Private_Part => Closed);
  pragma Module_Name (4, 3201);
  type File_Type is limited private;
  type File_Mode is (In_File, Out_File);
  type Count is range 0 .. 1_000_000_000;
  subtype Positive_Count is Count range 1 .. Count'Last;
  Unbounded : constant Count := 0; -- line and page length
  subtype Field is Integer range 0 .. Integer'Last;
  subtype Number_Base is Integer range 2 .. 16;
  type Type_Set is (Lower_Case, Upper_Case);

  -- File Management
  procedure Create (File : in out File_Type;
                  Mode : File_Mode := Out_File;
                  Name : String := "";
                  Form : String := "");
  procedure Open (File : in out File_Type;
                 Mode : File_Mode := Out_File;
                 Name : String;
                 Form : String := "");
  procedure Close (File : in out File_Type);
  procedure Delete (File : in out File_Type);
  procedure Reset (File : in out File_Type; Mode : File_Mode);
  procedure Reset (File : in out File_Type);
  function Mode (File : File_Type) return File_Mode;
  function Name (File : File_Type) return String;
  function Form (File : File_Type) return String;
  function Is_Open (File : File_Type) return Boolean;

  -- Control of default input and output files
  procedure Set_Input (File : File_Type);
  procedure Set_Output (File : File_Type);
  function Standard_Input return File_Type;
  function Standard_Output return File_Type;
  function Current_Input return File_Type;
  function Current_Output return File_Type;

  -- Specification of line and page lengths

```

```

  procedure Set_Line_Length (File : File_Type; To : Count);
  procedure Set_Page_Length (File : File_Type; To : Count);
  procedure Set_Page_Length (File : File_Type; To : Count);
  function Line_Length (File : File_Type) return Count;
  function Page_Length (File : File_Type) return Count;
  function Page_Length (File : File_Type) return Count;
  -- Column, Line and Page Control
  procedure New_Line (File : File_Type; Spacing : Positive_Count := 1);
  procedure New_Line (Spacing : Positive_Count := 1);
  procedure Skip_Line (File : File_Type; Spacing : Positive_Count := 1);
  procedure Skip_Line (Spacing : Positive_Count := 1);
  function End_Of_Line (File : File_Type) return Boolean;
  function End_Of_Line return Boolean;
  procedure New_Page (File : File_Type);
  procedure New_Page;
  procedure Skip_Page (File : File_Type);
  procedure Skip_Page;
  function End_Of_Page (File : File_Type) return Boolean;
  function End_Of_Page return Boolean;
  function End_Of_File (File : File_Type) return Boolean;
  function End_Of_File return Boolean;
  procedure Set_Col (File : File_Type; To : Positive_Count);
  procedure Set_Col (To : Positive_Count);
  procedure Set_Line (File : File_Type; To : Positive_Count);
  procedure Set_Line (To : Positive_Count);
  function Col (File : File_Type) return Positive_Count;
  function Col return Positive_Count;
  function Line (File : File_Type) return Positive_Count;
  function Line return Positive_Count;
  function Page (File : File_Type) return Positive_Count;
  function Page return Positive_Count;
  -- Character Input-Output
  procedure Get (File : File_Type; Item : out Character);
  procedure Get (Item : out Character);
  procedure Put (File : File_Type; Item : Character);

```

```

procedure Put (Item : Character);
-- String Input-Output

procedure Get (File : File_Type; Item : out String);
procedure Get (Item : out String);
procedure Put (File : File_Type; Item : String);
procedure Put (Item : String);

procedure Get_Line
  (File : File_Type; Item : out String; Last : out Natural);
procedure Get_Line (Item : out String; Last : out Natural);

procedure Put_Line (File : File_Type; Item : String);
procedure Put_Line (Item : String);

-- Generic package for Input-Output of Integer Types
generic
  type Num is range <>;
package Integer_Io is
  Default_Width : Field := Num'Width;
  Default_Base : Number_Base := 10;

  procedure Get (File : File_Type; Item : out Num; Width : Field := 0);
  procedure Get (Item : out Num; Width : Field := 0);

  procedure Put (File : File_Type;
    Item : Num;
    Width : Field := Default_Width;
    Base : Number_Base := Default_Base);

  procedure Put (Item : Num;
    Width : Field := Default_Width;
    Base : Number_Base := Default_Base);

  procedure Get (From : String; Item : out Num; Last : out Positive);

  procedure Put (To : out String;
    Item : Num;
    Base : Number_Base := Default_Base);
end Integer_Io;

-- Generic package for Input-Output of Floating Point Types
generic
  type Num is digits <>;
package Float_Io is
  Default_Fore : Field := 2;
  Default_Aft : Field := Num'Digits - 1;
  Default_Exp : Field := 3;

  procedure Get (File : File_Type; Item : out Num; Width : Field := 0);

```

```

  procedure Get (Item : out Num; Width : Field := 0);

  procedure Put (File : File_Type;
    Item : Num;
    Fore : Field := Default_Fore;
    Aft : Field := Default_Aft;
    Exp : Field := Default_Exp);

  procedure Put (Item : Num;
    Fore : Field := Default_Fore;
    Aft : Field := Default_Aft;
    Exp : Field := Default_Exp);

  procedure Get (From : String; Item : out Num; Last : out Positive);

  procedure Put (To : out String;
    Item : Num;
    Aft : Field := Default_Aft;
    Exp : Field := Default_Exp);
end Float_Io;

-- Generic package for Input-Output of Fixed Point Types
generic
  type Num is delta <>;
package Fixed_Io is
  Default_Fore : Field := Num'Fore;
  Default_Aft : Field := Num'Aft;
  Default_Exp : Field := 0;

  procedure Get (File : File_Type; Item : out Num; Width : Field := 0);
  procedure Get (Item : out Num; Width : Field := 0);

  procedure Put (File : File_Type;
    Item : Num;
    Fore : Field := Default_Fore;
    Aft : Field := Default_Aft;
    Exp : Field := Default_Exp);

  procedure Put (Item : Num;
    Fore : Field := Default_Fore;
    Aft : Field := Default_Aft;
    Exp : Field := Default_Exp);

  procedure Get (From : String; Item : out Num; Last : out Positive);

  procedure Put (To : out String;
    Item : Num;
    Aft : Field := Default_Aft;
    Exp : Field := Default_Exp);
end Fixed_Io;

-- Generic package for Input-Output of Enumeration Types
generic
  type Enum is (<>);

```

```

package Enumeration_1o is
    Default_Width : Field := 0;
    Default_Setting : Type_Set := Upper_Case;

    procedure Get (File : File_Type; Item : out Enum);
    procedure Get (Item : out Enum);

    procedure Put (File : File_Type;
                  Item : Enum;
                  Width : Field := Default_Width;
                  Set : Type_Set := Default_Setting);

    procedure Put (Item : Enum;
                  Width : Field := Default_Width;
                  Set : Type_Set := Default_Setting);

    procedure Get (From : String; Item : out Enum; Last : out Positive);

    procedure Put (To : out String;
                  Item : Enum;
                  Set : Type_Set := Default_Setting);
end Enumeration_1o;

-- Exceptions
Status_Error : exception renames Io_Exceptions.Status_Error;
Mode_Error : exception renames Io_Exceptions.Mode_Error;
Name_Error : exception renames Io_Exceptions.Name_Error;
Use_Error : exception renames Io_Exceptions.Use_Error;
Device_Error : exception renames Io_Exceptions.Device_Error;
End_Error : exception renames Io_Exceptions.End_Error;
Data_Error : exception renames Io_Exceptions.Data_Error;
Layout_Error : exception renames Io_Exceptions.Layout_Error;

private
type File_Type is new Device_Independent_1o.File_Type;
end Text_1o;

```

```

with Io_Exceptions;
package Window_1o is
    pragma Subsystem (Object_Editor, Closed);
    pragma Module_Name (4, 2219);

    -- package for providing raw IO facilities to an image
    type File_Type is private;

    type File_Mode is (In_File, Out_File);
    -- the mode of the handle. Each image can be opened twice - once
    -- for input and once for output.

    -- Create an image for IO.
    -- Normally, a new empty image is created on this call.
    -- If an image is already open for this job with the given name,
    -- and the mode given /= the mode the image is open for, that
    -- image will be opened for the new mode.
    procedure Create (File : in out File_Type;
                    Mode : File_Mode := Out_File;
                    Name : String;
                    Form : String := "");

    -- Open a previously closed image. The same rules apply for create
    -- in the case one job opens the same image twice; once for input and
    -- once for output
    procedure Open (File : in out File_Type;
                  Mode : File_Mode := Out_File;
                  Name : String;
                  Form : String := "");

    -- Terminate operations on this image.
    procedure Close (File : in out File_Type);

    -- Delete the image. Any other handles on this image are implicitly
    -- closed.
    procedure Delete (File : in out File_Type);

    function Mode (File : File_Type) return File_Mode;
    function Name (File : File_Type) return String;
    function Form (File : File_Type) return String;

    function Is_Open (File : File_Type) return Boolean;

package Raw is
    -- gain access to the keyboard for "raw" input.
    -- one channel may be opened per job.
    -- no echoing or local editing is performed.

    type Stream_Type is private;

    procedure Open (Stream : in out Stream_Type);
    procedure Close (Stream : in out Stream_Type;
                    Flush_Pending_Input : Boolean := False);

```

```

procedure Disconnect (Stream : in out Stream_Type);
-- free users keyboard

type Key is new Natural range 0 .. 1023;
type Key_String is array (Positive range <>) of Key;
-- a key is the basic bit of input.

subtype Simple_Key is Key range 0 .. 127;
-- a simple key represents the ascii characters

procedure Get (Stream : Stream_Type; Item : out Key);
procedure Get (Stream : Stream_Type; Item : out Key_String);

-- converting keys to characters

-- the ascii characters map directly to the first 128 keys
function Convert (C : Character) return Simple_Key;
function Convert (K : Simple_Key) return Character;

-- keys are mapped to logical names
-- these names correspond to the 'image' attribute of the
-- enumerations in machine.editor_data.visible_key_names

subtype Terminal is String;
-- supported terminal types are Cit500R, Vt100, Rational

function Image (For_Key : Key; On_Terminal : Terminal) return String;
-- image is "", if For_Key is not defined for this terminal type

procedure Value (For_Key_Name : String;
                 On_Terminal : Terminal;
                 Result : out Key;
                 Found : out Boolean);
-- Found is false => For_Key_Name does not name a key on this terminal

function Value
  (For_Key_Name : String; On_Terminal : Terminal) return Key;

Unknown_Key : exception;
-- raised by functional form of value

end Raw;

subtype Column_Number is Positive;
subtype Line_Number is Positive;
-- a file_type is initialized to column 1, line 1

subtype Count is Natural;
subtype Positive_Count is Count range 1 .. Count'Last;

-- output
-- characters are displayed at the current cursor position
-- control characters are displayed in reverse-video

type Designation is (Text, Prompt, Protected);
type Attribute is
  record

```

```

  Bold : Boolean;
  Faint : Boolean;
  Underscore : Boolean;
  Inverse : Boolean;
  Slow_Blink : Boolean;
  Rapid_Blink : Boolean;
  Unused_0 : Boolean;
  Unused_1 : Boolean;
end record;

Vanilla : constant Attribute := (others => False);

type Character_Set is new Natural range 0 .. 15;

Plain : constant Character_Set := 0;
Graphics : constant Character_Set := 1;

type Font is
  record
    Kind : Character_Set;
    Look : Attribute;
end record;

Normal : constant Font := Font'(Plain, Vanilla);

function Default_Font (For_Type : Designation) return Font;
-- the fonts normally used by the environment for these designations
-- are returned

procedure Position_Cursor (File : File_Type;
                          Line : Line_Number := Line_Number'First;
                          Column : Column_Number := Column_Number'First;
                          Offset : Natural := 0);

-- Position the cursor on the image.
-- Offset is used to position the cursor relative to the top of the window.
-- With an offset of 0, the cursor is made visible in the window using
-- the normal editor defaults.
-- With a positive offset, the image is scrolled in the window so the
-- cursor is the offset line in the window.

procedure Move_Cursor (File : File_Type;
                     Delta_Lines : Integer;
                     Delta_Columns : Integer;
                     Offset : Natural := 0);

procedure Report_Cursor (File : File_Type;
                        Line : out Line_Number;
                        Column : out Column_Number);

procedure Overwrite (File : File_Type;
                    Item : Character;
                    Image : Font := Normal;
                    Kind : Designation := Text);
-- writes ITEM at the current cursor position and advances column by 1

procedure Overwrite (File : File_Type;
                    Item : String;
                    Image : Font := Normal;
                    Kind : Designation := Text);
-- writes ITEM at the current cursor position and advances column by

```



```

-- ITEM'LENGTH

procedure Insert (File : File_Type;
Item : Character;
Image : Font := Normal;
Kind : Designation := Text);
-- writes ITEM at the current cursor position and advances column by 1

procedure Insert (File : File_Type;
Item : String;
Image : Font := Normal;
Kind : Designation := Text);
-- writes ITEM at the current cursor position and advances column by
-- ITEM'LENGTH

procedure New_Line (File : File_Type; Lines : Count := 1);
-- insert lines after the current line
-- advances line by Lines, and sets column to 1.

procedure Delete (File : File_Type; Characters : Count);
-- deletes Count characters at current position. Position is unchanged

procedure Delete_Lines (File : File_Type; Lines : Count := 1);
-- deletes Lines including the current line. Position is unchanged

-- input with editing
-- an input prompt with contents PROMPT will be displayed at the current
-- cursor position. Control of the keyboard will be returned to the
-- core editor for user input at the prompt.

procedure Get (File : File_Type;
Prompt : String := "[input]";
Item : out Character);
procedure Get (File : File_Type;
Prompt : String := "[input]";
Item : out String);
procedure Get_Line (File : File_Type;
Prompt : String := "[input]";
Item : out String;
Last : out Natural);

function Get_Line
(File : File_Type; Prompt : String := "[input]") return String;
-- banner operations

-- The value will be displayed in the banner for this image
-- fields are defined from left to right. The first few fields
-- are reserved for the editor. Users may specify field_names
-- of the form "FIELD_0" .. "FIELD_9". Currently 0 .. 2 are used
-- for job_number, start_time and blocked indication, but may be
-- reused by the user.
-- Calling set_banner with other values will be a noop.

procedure Set_Banner
(File : File_Type; Field_Name : String; Value : String);

function Read_Banner (File : File_Type; Field_Name : String) return String;
-- predefined field_names, may be passed to Set_Banner
function Job_Number return String;

```

```

function Job_Time return String;

-- sound the terminal bell
procedure Bell (File : File_Type);

-- information about the current image

function End_Of_Line (File : File_Type) return Boolean;
function End_Of_File (File : File_Type) return Boolean;
function Line_Length (File : File_Type) return Count;
function Line_Image (File : File_Type) return String;
function Char_At (File : File_Type) return Character;
function Font_At (File : File_Type) return Font;
function Last_Line (File : File_Type) return Line_Number;

-- information about the current window

-- the origin is the line and column number of the point of the image
-- located in the upper right corner of the window
procedure Report-Origin (File : File_Type;
Line : out Line_Number;
Column : out Column_Number);

-- the size of the window in characters
procedure Report_Size (File : File_Type;
Lines : out Positive_Count;
Columns : out Positive_Count);

-- the location of the window on the screen
-- the upper right corner of the screen is line 1, column 1
procedure Report_Location (File : File_Type;
Line : out Line_Number;
Column : out Column_Number);

end Window_1o;

```



```

package Calendar is
pragma Subsystem (Kernel, Private_Part => Closed);
pragma Module_Name (4, 406);

type Time is private;

subtype Year_Number is Integer range 1901 .. 2099;
subtype Month_Number is Integer range 1 .. 12;
subtype Day_Number is Integer range 1 .. 31;
subtype Day_Duration is Duration range 0.0 .. 86_400.0;

function Clock return Time;

function Year (Date : Time) return Year_Number;
function Month (Date : Time) return Month_Number;
function Day (Date : Time) return Day_Number;
function Seconds (Date : Time) return Day_Duration;

procedure Split (Date : Time;
  Year : out Year_Number;
  Month : out Month_Number;
  Day : out Day_Number;
  Seconds : out Day_Duration);

function Time_Of (Year : Year_Number;
  Month : Month_Number;
  Day : Day_Number;
  Seconds : Day_Duration := 0.0) return Time;

function "+" (Left : Time; Right : Duration) return Time;
function "-" (Left : Duration; Right : Time) return Time;
function "--" (Left : Time; Right : Duration) return Time;
function "---" (Left : Time; Right : Time) return Duration;

function "<" (Left, Right : Time) return Boolean;
function "<=" (Left, Right : Time) return Boolean;
function ">" (Left, Right : Time) return Boolean;
function ">=" (Left, Right : Time) return Boolean;

Time_Error : exception; -- can be raised by TIME_OF, "+", and "--"

end Calendar;

```

```

package Standard is
type Boolean is (False, True);
for Boolean'Size use 1;

type Integer is range -2**31-1 .. 2**31-1;

type Long_Integer is range (-2**62 - 2**62) .. (2**62 - 1 + 2**62);
-- -2**63 .. 2**63-1

type Float is digits 15 range (2.0**1023) - (2.0**97) + (2.0**1023) ..
-- ((2.0**1023) - (2.0**97) + (2.0**1023));
-- -1.7977E308 .. 1.7977E308;

type Character is (Nul, ..., Del);
for Character use (0, ..., 127);
for Character'Size use 8;

package Ascii is ... end Ascii;

subtype Natural is Integer range 0 .. Integer'Last;
subtype Positive is Integer range 1 .. Integer'Last;

type String is array (Positive range <>) of Character;

type Duration is delta 2.0**(-15)
-- -3.051757812500E-05
range - (2.0**32) .. (2.0**32) - (2.0**(-15));
-- -4.294967296000E+09 .. 4.294967296000E+09

Constraint_Error : exception;
Numeric_Error : exception;
Program_Error : exception;
Storage_Error : exception;
Tasking_Error : exception;

end Standard;

```

package System is

```
pragma Read_Only;
pragma Open_Private_Part;
pragma Subsystem (Ada_Base);
pragma Module_Name (4, 66);
```

```
type Address is private;
```

```
Null_Address : constant Address;
```

```
type Name is (R1000);
```

```
System_Name : constant Name := R1000;
```

```
Bit : constant := 1;
```

```
Storage_Unit : constant := 1 * Bit;
```

```
Word_Size : constant := 128 * Bit;
```

```
Byte_Size : constant := 8 * Bit;
```

```
Megabyte : constant := (2 ** 20) * Byte_Size;
```

```
Memory_Size : constant := 32 * Megabyte;
```

```
-- System-Dependent Named Numbers
```

```
Min_Int : constant := Long_Integer'Pos (Long_Integer'First);
```

```
Max_Int : constant := Long_Integer'Pos (Long_Integer'Last);
```

```
Max_Digits : constant := 15;
```

```
Max_Mantissa : constant := 63;
```

```
Fine_Delta : constant := 1.0 / (2.0 ** 63);
```

```
Tick : constant := 200.0E-9;
```

```
subtype Priority is Integer range 0 .. 5;
```

```
type Byte is new Natural range 0 .. 255;
```

```
type Byte_String is array (Natural range <>) of Byte;
```

```
-- Basic units of transmission/reception to/from IO devices.
```

```
type Virtual_Processor_Number is new Long_Integer range 0 .. 2 ** 10 - 1;
```

```
type Module_Number is new Long_Integer range 0 .. 2 ** 22 - 1;
```

```
type Module_Name is new Long_Integer range 0 .. 2 ** 32 - 1;
```

```
subtype Code_Segment_Name is Module_Name range 0 .. 2 ** 24 - 1;
```

```
type Bit_Offset is new Long_Integer range 0 .. 2 ** 32 - 1;
```

```
Null_Module : constant Module_Name := 0;
```

```
function Convert (The_Address : Address) return Long_Integer;
```

```
pragma Suppress (Elaboration_Check, Convert);
```

```
function Extract_Vp (From_Address : Address)
```

```
return Virtual_Processor_Number;
```

```
pragma Suppress (Elaboration_Check, Extract_Vp);
```

```
function Extract_Number (From_Address : Address) return Module_Number;
```

```
pragma Suppress (Elaboration_Check, Extract_Number);
```

System, !Lrm

```
function Extract_Name (From_Address : Address) return Module_Name;
pragma Suppress (Elaboration_Check, Extract_Name);

function Extract_Offset (From_Address : Address) return Bit_Offset;
pragma Suppress (Elaboration_Check, Extract_Offset);

function Get_Vp (From_Name : Module_Name) return Virtual_Processor_Number;
pragma Suppress (Elaboration_Check, Get_Vp);

function Get_Number (From_Name : Module_Name) return Module_Number;
pragma Suppress (Elaboration_Check, Get_Number);

function Compose_Name (With_Vp : Virtual_Processor_Number;
With_Number : Module_Number) return Module_Name;
pragma Suppress (Elaboration_Check, Compose_Name);

function Current_Name return Module_Name;
pragma Suppress (Elaboration_Check, Current_Name);

function Current_Vp return Virtual_Processor_Number;
pragma Suppress (Elaboration_Check, Current_Vp);

function Current_Number return Module_Number;
pragma Suppress (Elaboration_Check, Current_Number);

type Segment is private;
Null_Segment : constant Segment;

type Package_Type is private;
pragma Enable_Runtime_Privacy (Package_Type);

Null_Package : constant Package_Type;

Invalid_Package_Value : exception;

type Exception_Number is new Long_Integer range 0 .. 2 ** 48 - 1;

Operand_Class_Error : exception;
pragma Exception_Name (Operand_Class_Error, 96);

Type_Error : exception;
pragma Exception_Name (Type_Error, 97);

Visibility_Error : exception;
pragma Exception_Name (Visibility_Error, 98);

Capability_Error : exception;
pragma Exception_Name (Capability_Error, 99);

Machine_Restriction : exception;
pragma Exception_Name (Machine_Restriction, 100);

Illegal_Instruction : exception;
pragma Exception_Name (Illegal_Instruction, 101);
```

```

Illegal_Reference : exception;
pragma Exception_Name (Illegal_Reference, 102);

Illegal_Frame_Exit : exception;
pragma Exception_Name (Illegal_Frame_Exit, 103);

Record_Field_Error : exception;
pragma Exception_Name (Record_Field_Error, 104);

Utility_Error : exception;
pragma Exception_Name (Utility_Error, 105);

Unsupported_Feature : exception;
pragma Exception_Name (Unsupported_Feature, 106);

Illegal_Heap_Access : exception;
pragma Exception_Name (Illegal_Heap_Access, 107);

Select_Use_Error : exception;
pragma Exception_Name (Select_Use_Error, 108);

Frame_Establish_Error : exception;
pragma Exception_Name (Frame_Establish_Error, 129);

Nonexistent_Space_Error : exception;
pragma Exception_Name (Nonexistent_Space_Error, 131);

Nonexistent_Page_Error : exception;
pragma Exception_Name (Nonexistent_Page_Error, 132);

Write_To_Read_Only_Page : exception;
pragma Exception_Name (Write_To_Read_Only_Page, 133);

Heap_Pointer_Copy_Error : exception;
pragma Exception_Name (Heap_Pointer_Copy_Error, 134);

Assertion_Error : exception;
pragma Exception_Name (Assertion_Error, 135);

Microcode_Assist_Error : exception;
pragma Exception_Name (Microcode_Assist_Error, 136);

private
type Address is new Long_Integer;
Null_Address : constant Address := 0;

type Segment is access Boolean;
pragma Segmented_Heap (Segment);

Null_Segment : constant Segment := null;

type Package_Type is new Long_Integer;
Null_Package : constant Package_Type := 0;

end System;

```

```

generic
type Source is limited private;
type Target is limited private;
function Unchecked_Conversion (S : Source) return Target;

pragma Subsystem (Miscellaneous);
pragma Module_Name (4, 824);

```

```
generic
type Object is limited private;
type Name is access Object;
procedure Unchecked_Deallocation (X : in out Name);
pragma Subsystem (Miscellaneous);
pragma Module_Name (4. 825);
```

Unchecked\_Deallocation, !Lrm

```

with Simple_Status;
with Bounded_String;
with Directory;

with Machine;

```

```

package Access_List_Tools is

```

```

    subtype Name is String; -- an object name

    subtype Access_Class is String; -- of only the following characters:
    Read : constant Character := 'R'; -- objects and worlds
    Write : constant Character := 'W'; -- objects only
    Delete : constant Character := 'D'; -- worlds only; same bit as W
    Create : constant Character := 'C'; -- worlds only
    Owner : constant Character := 'O'; -- worlds only

    -- An object string name is as defined by the directory
    -- package. No wildcards are accepted; each operation in this
    -- package operates on one object.

```

```

    subtype Acl is String;
    Max_Acl_Length : constant := 512; -- max length for access list string
    -- The max size will not be exceeded when an Acl is returned.

```

```

    -- String representations of access lists have the following syntax:

```

```

    Acl ::= Acl_Entry [ ',' Acl_Entry ] *
    Acl_Entry ::= Group '>' Access
    Group ::= Identifier
    Access ::= Acc_Type +
    Acc_Type ::= 'R' | 'W' | 'D' | 'C' | 'O' |
               'r' | 'w' | 'd' | 'c' | 'o'
    -- Examples: "Phil => R, IRM => rw", "Public=>RCOD"

```

```

    Access_Tools_Error : exception; -- Raised by functions

```

```

function Get (For_Object : Name) return Acl;
function Get (For_Object : Directory.Version) return Acl;
procedure Get (For_Object : Name;
              List : out Bounded_String.Variable_String;
              Status : in out Simple_Status.Condition);
procedure Get (For_Object : Directory.Version;
              List : out Bounded_String.Variable_String;
              Status : in out Simple_Status.Condition);

procedure Set (For_Object : Name;
              To_List : Acl;
              Status : in out Simple_Status.Condition);
procedure Set (For_Object : Directory.Version;
              To_List : Acl;
              Status : in out Simple_Status.Condition);

```

```

-- Get or Set the access list for the specified object.
-- Setting the access list requires "Owner" access.
-- function Get raises Access_Tools_Error if an error occurs.
-- The procedure version should be called in that case to get the

```

```

Access_List_Tools, !Tools

```

```

-- actual error information.
-- ACL for world must be contain only R, C, O, or D access. Others
-- must be only R or W access.

```

```

function Check (User_Name : String := "";
               Object_Id : Directory.Version;
               Desired : Access_Class) return Boolean;
function Check (User_Name : String := "";
               Object_Name : String;
               Desired : Access_Class) return Boolean;
function Check (User_Id : Directory.Version;
               Object_Id : Directory.Version;
               Desired : Access_Class) return Boolean;
function Check (Job : Machine.Job_Id;
               Object_Id : Directory.Version;
               Desired : Access_Class) return Boolean;

```

```

-- Check if the specified user has the indicated access to the
-- specified object. Only meaningful for Ada objects, Files, and Worlds.
-- The null string for the User_Name parameter means the identity of
-- the calling job. If a user name is specified, the access control
-- identity of that user (its member groups) is used for the test.
-- If an error is detected during the test, the value false is returned.
-- The most common errors are illegal values for Desired and references
-- to objects that do not exist. If an object that does not have an
-- access list is referenced, the value true is returned.

```

```

function Get_Default (For_World : Name) return Acl;

```

```

procedure Get_Default (For_World : Name;
                     List : out Bounded_String.Variable_String;
                     Status : in out Simple_Status.Condition);

```

```

procedure Set_Default (For_World : Name;
                     To_List : Acl;
                     Status : in out Simple_Status.Condition);

```

```

-- Get or set the default ACL for new objects created in the specified
-- world. The function raises the exception Access_Tools_Error if
-- an error is detected. The procedure version returns a status
-- that indicates the cause of the error.

```

```

procedure Check_Validity (For_List : Acl;
                        Status : in out Simple_Status.Condition);
-- Check the validity of the specified access list. Return status
-- indicating that it is okay, or the error, if any.

```

```

function Has_Operator_Capability return Boolean;
-- Return true if the calling job has operator capability. This is
-- true if the job has an identity that includes the group
-- "operator", is on the access list for "machine.operator_capability",
-- or is privileged.

```

```

function Normalize (Initial_Acl : Acl) return Acl;
-- Scan the acl and eliminate any entries for groups that do
-- not currently exist. Return the revised acl. If the
-- acl is otherwise illegal, raise Access_Tools_Error.

```

```

function Amend (Initial_Acl : Acl; New_Group : Name; Desired : Access_Class)
return Acl;

```

```

-- Amend Initial_Acl so that New_Group is granted Desired access. If
-- necessary, the right-most acl entry is removed to do this.

```

```
-- Raise Access_Fools_Error if any parameter is illegal.
```

```
pragma Subsystem (Os_Commands);  
pragma Module_Name (4, 3508);  
end Access_List_Fools;
```

```
with Directory;  
with Diana;  
package Ada_Object_Editor is  
  
  Lock_Error, Undefined : exception;  
  -- Lock_Error will be raised if the designated tree is open for  
  -- update by the editor.  
  -- Undefined exception will be raised if the designated object  
  -- does not exist.  
  
  function Current_Image return Diana.Tree;  
  function Current_Selection return Diana.Tree;  
  -- Both functions return the appropriate tree with access mode none.  
  -- Lock_Error and Undefined may be raised.  
  
  procedure Display (Tree : Diana.Tree);  
  -- Create a window displaying the given tree. If this tree is already  
  -- displayed on an existing window this window will become the  
  -- current focus. If a new window is created it will be designated  
  -- read-only.  
  
  type Window_State is (Normal, Promoted, Demoted);  
  type Selection_Request is (Must_Select, Dont_Select, Try_Select);  
  type Display_Status is (Successful, Locked_Out, Illegal_Access,  
                          Cannot_Select, Nonexistent_Tree, Unknown_Error);  
  
  procedure Display (Tree : Diana.Tree;  
                    Status : out Display_Status;  
                    Selection_Command : Selection_Request := Try_Select;  
                    State_Of_Window : Window_State := Normal);  
  -- Displays the tree. State_Of_Window indicates if the window should  
  -- appear in its current state, or be promoted, or be demoted. The  
  -- Selection_Command parameter controls as follows:  
  -- Must_Select: Fails if selection is not possible  
  -- Dont_Select: Does no selection  
  -- Try_Select : Tries to select; on failure brings up window anyway  
  
  function Image_Name return String;  
  function Selection_Name  
    (From_Current_Image_Only : Boolean := True) return String;  
  -- Functions to return a directory name for the image or the selection.  
  -- Both functions return "##Unknown##" if unable to get a name from  
  -- from the directory. Selection_Name returns "##No_Selection##"  
  -- if no selection is found on the current or any image (as designated  
  -- by the parameter). No exceptions come out of these functions.  
  
  pragma Subsystem (Ada_Oe);  
  pragma Module_Name (4, 2200);  
end Ada_Object_Editor;
```



```

with Dians;
with Directory;
with Directory_Tools;

package Ada_Text is
pragma Subsystem (Tools);
pragma Module_Name (4, 3548);

-- Ada objects have two components: a Diana tree and a textual image.
-- This package exports a mapping between the tree and the image, and
-- provides read access to the image.

type Handle is private;
Nil_Handle : constant Handle;

procedure Open (Ada_Object : Directory_Tools.Object.Handle;
Unit : out Handle;
Status : out Directory_Tools.Object.Error_Code);

procedure Open (Ada_Object : Directory.Version;
Unit : out Handle;
Status : out Directory.Error_Status);

-- Open acquires a read lock on both the tree and the image, returning
-- a handle that can be used to make further queries. Nil_Handle is
-- returned if the Open fails.

procedure Close (Unit : in out Handle;
Status : out Directory_Tools.Object.Error_Code);

procedure Close (Unit : in out Handle; Status : out Directory.Error_Status);

-- Close releases the locks acquired by Open and sets Unit to
-- Nil_Handle. Using a handle or iterators obtained from it after the
-- handle has been closed will have unpredictable erroneous results.

function Root (Unit : Handle) return Diana.Tree;
-- Returns the root of the unit; returns Diana.Empty if the
-- opened Ada unit is in archive state.

type Area is
record
First_Line : Positive;
First_Column : Positive;
Last_Line : Natural;
Last_Column : Natural;
end record;

Nil_Area : constant Area := (1, 1, 0, 0);

-- An area indicates a stream of contiguous characters on the screen.
-- First_Line and First_Column are the coordinates of the first
Ada_Text, iTools

```

```

-- character of the stream; Last_Line and Last_Column are the
-- coordinates of the last character in the stream. Lines and
-- columns are numbered beginning from 1.

function Is_Empty (Where : Area) return Boolean;
-- An area A is considered empty iff A.Last_Line < A.First_Line or
-- A.First_Line = A.Last_Line and then A.Last_Column < A.First_Column.

function Entire (Unit : Handle) return Area;
-- Returns the area corresponding to the entire image.

function Has_Partial_Lines (Subtree : Diana.Tree) return Boolean;
-- Indicates whether the image of a subtree uses an integral number
-- of lines (such as a statement), or whether it may start or end
-- in the middle of a line (as an expression). If this function
-- returns True for some node, it will also return True for all its
-- children (even for a child whose image happens to occupy an integral
-- number of lines.)

function Where_Is (Subtree : Diana.Tree;
Unit : Handle;
And_Post_Comment : Boolean := True) return Area;

-- Returns the area in the image that corresponds to some subtree.
-- If Has_Partial_Lines (Subtree) is true, then the area returned will
-- not include any leading blanks. If Has_Partial_Lines (Subtree) is
-- False, then the area returned will begin in column 1. The area
-- returned will include a trailing Same_Line comment (see below) if
-- one is present and the And_Post_Comment parameter is true.

function What_Line (Subtree : Diana.Tree; Unit : Handle) return Natural;
-- Returns 0 for Diana.Empty, and the line number the subtree begins
-- on for nonempty trees.

function What_Is (Where_Is : Area; Unit : Handle) return Diana.Tree;
-- Returns the smallest subtree whose image contains the area.

function What_Is (On_Line : Positive; Unit : Handle) return Diana.Tree;
-- Returns the smallest subtree T such that Has_Partial_Lines (T) is
-- False and the image of T contains line On_Line.

function What_Statement
(On_Line : Positive; Unit : Handle) return Diana.Tree;
-- Similar to What_Is, but has a slightly coarser granularity.
-- What_Statement will always return a tree of class DECL or STM, or
-- an ancestor of such a tree.

type Comment_Kind is (Same_Line, Own_Line, Both);

-- A Same_Line comment begins on the same line as an Ada token.
-- Subsequent comment lines are considered to be a continuation of the

```

```

-- Same_Line comment, as long as their double-dash delimiter is in the
-- same column as the initial comment. There may be intervening empty
-- lines as long as they are followed by another properly aligned comment
-- line.
--
-- An Own_Line comment consists of lines that contain only comments (no
-- Ada tokens).
--
-- The collection of comments and white space between two Ada tokens
-- are considered by this package to be either a Same_Line comment, an
-- Own_Line comment, or a Same_Line comment followed by an Own_Line
-- comment.
--
-- The Comment_Kind Both refers to either kind of comment if only one
-- is present, or their concatenation if both are present.
--
-- Examples:
--
A := 0; -- A single-line Same_Line comment
B := 0; -- This Same_Line comment
-- contains two lines.
C := 0;
-- These lines make up
-- an own-line comment.
D := 0;
--
-- The blanks lines before, after, and between these lines
-- are all part of the own-line comment
E := 0;
--
F := 0; -- The blank line before this line is considered
-- to be an Own-line comment.
G := 0; -- A same-line comment
-- Followed by an Own-line comment
H := 0; -- Same-line comments may have blank lines
-- Embedded within them,
-- but the blank line that precedes this one is part of
-- the Own_Line comment.
I := 0;

function Pre_Comment (Tree : Diana.Tree; Kind : Comment_Kind; Unit : Handle)
return Area;

function Post_Comment
(Tree : Diana.Tree; Kind : Comment_Kind; Unit : Handle)
return Area;

-- These functions examine the comment text before the first token
-- or after the last token of program text corresponding to the given
-- Diana tree. If there is a comment there that matches the Kind
-- parameter, then the Area for that comment is returned; otherwise
-- Nil_Area is returned.
--
-- In general, the same comment can be returned for more than one tree.

```

```

-- For example, in:
--
A.B := 1;
-- comment
C.D := 2;
--
-- the comment can be returned as a post-comment of the first Dn_Assign
-- node, or as a pre-comment on the second Dn_Assign node, the
-- Dn_Selected node for C.D or the Dn_Used_Name_Id node for C.
--
-- If a piece of a comment that matches the Kind parameter, just the
-- piece will be returned. For example, in:
--
P; -- comment 1
-- comment 2
Q;
--
-- the first comment will be returned as the Same_Line post-comment of
-- P and the Same_Line pre-comment of Q. Likewise, the second comment
-- be returned as the Own_Line post-comment of P and the Own_Line
-- pre-comment of Q. The value of Same_Line pre-comments is dubious,
-- but they have been provided for the sake of completeness.
--
type Iterator is private;
Nil_Iterator : constant Iterator;

procedure Initialize (Unit : Handle; Where : Area; Iter : out Iterator);
function Done (Iter : Iterator) return Boolean;
procedure Next (Iter : in out Iterator);
--
-- Iterators can be used to retrieve the text that is in some area
-- of an image. An iterator will return a sequence of strings;
-- one string for each line in the area. For an area A,
-- if A.Last_Line < A.First_Line, then no strings will be returned.
-- Otherwise, A.Last_Line - A.First_Line + 1 strings will be returned.
-- The first string returned will be truncated so that characters
-- before (A.First_Line, A.Last_Line) will not be returned. The
-- last string returned will be truncated so that characters after
-- (A.Last_Line, A.Last_Column) will not be returned.
--
function Value (Iter : Iterator) return String;
function Leading_Blanks (Iter : Iterator) return Natural;
function Nonblank_Value (Iter : Iterator) return String;
--
-- Most of the strings returned by the iterator will begin with
-- some leading blanks. The Value function returns the string
-- with its leading blanks. Alternatively, the Leading_Blanks and
-- Nonblank_Value can be used to get these values separately. These
-- functions obey the identity:
--
Value (I) = String'(1..Leading_Blanks (I) => ' ') &
Nonblank_Value(I)
--
-- Warning: the string values returned will be a slice of some internal
-- buffer, so that the numeric values of the lower and upper bounds
-- will not have any meaningful value.

```

```

-- If the iterator is not convenient, the following functions may
-- be used examine the image.

function Number_Of_Lines (Unit : Handle) return Natural;

function Get_Line (Line : Positive; Unit : Handle) return String;
-- Returns the null string if Line is out of bounds.
-- The lower bound of the returned string will be 1.

function Line_Length (Line : Positive; Unit : Handle) return Natural;
-- Returns zero if Line is out of bounds.

function Get_Character (Line : Positive; Column : Positive; Unit : Handle)
return Character;
-- Returns space if Line or Column is out of bounds.

private
type Open_State;
type Handle is access Open_State;
pragma Segmented_Heap (Handle);
Nil_Handle : constant Handle := null;

type Iterator_State;
type Iterator is access Iterator_State;
pragma Segmented_Heap (Iterator);
Nil_Iterator : constant Iterator := null;
end Ada_Text;

```

```

generic
type Object is limited private;
type Name is access Object;
function Allows_Deallocation return Boolean;
pragma Subsystem (Miscellaneous);
pragma Module_Name (4, 827);

```

```

package Bit_Operations is
-- Operations on Integer and Long_Integer as an array of bits.
-- Bit numbering is left to right, 0..31 and 0..63.
-- Arguments specified outside bit range raise Constraint_Error

-- Extract the bits from Start .. Start + Length - 1. Equivalent to
-- Logical_And with 1 in bits of the extraction range and 0 elsewhere.
function Extract
  (M : Integer; Start : Natural; Length : Natural) return Integer;
function Extract (M : Long_Integer; Start : Natural; Length : Natural)
  return Long_Integer;

-- Extract the single bit at B and return as Boolean
-- Equivalent to Extract (M, B, 1) /= 0
function Test_Bit (M : Integer; B : Natural) return Boolean;
function Test_Bit (M : Long_Integer; B : Natural) return Boolean;

-- Replace the specified bits of Into with the rightmost Length bits of M.
function Insert (M, Into : Integer; Start : Natural; Length : Natural)
  return Integer;
function Insert (M, Into : Long_Integer; Start : Natural; Length : Natural)
  return Long_Integer;

-- Set the specified Bit to One or Zero
procedure Set_Bit_To_One (M : in out Integer; B : Natural);
procedure Set_Bit_To_One (M : in out Long_Integer; B : Natural);
procedure Set_Bit_To_Zero (M : in out Integer; B : Natural);
procedure Set_Bit_To_Zero (M : in out Long_Integer; B : Natural);

-- Logical operations on two operands.
-- For shift operations, positive arguments shift Left and negative
-- arguments shift Right.
-- Intermediate results are stored as Long_Integer, so shifting
-- ones into positions outside of Integer will raise Numeric_Error.
function Logical_And (X, Y : Integer) return Integer;
function Logical_And (X, Y : Long_Integer) return Long_Integer;
function Logical_Or (X, Y : Integer) return Integer;
function Logical_Or (X, Y : Long_Integer) return Long_Integer;
function Logical_Xor (X, Y : Integer) return Integer;
function Logical_Xor (X, Y : Long_Integer) return Long_Integer;
function Logical_Not (X : Integer) return Integer;
function Logical_Not (X : Long_Integer) return Long_Integer;
function Logical_Shift (X : Integer; Amount : Integer) return Integer;
function Logical_Shift (X : Long_Integer; Amount : Integer) return Long_Integer;

--/R1000 pragma Module_Name (4, 828);
--/R1000 pragma Subsystem (Miscellaneous);
end Bit_Operations;

```

Bit\_Operations, !Tools

```

package Bounded_String is
pragma Subsystem (Tools);
pragma Module_Name (4, 3976);
subtype String_Length is Natural;
type Variable_String (Maximum_Length : String_Length) is private;
-- initialized to have a length of 0
procedure Copy (Target : in out Variable_String; Source : Variable_String);
procedure Copy (Target : in out Variable_String; Source : String);
procedure Copy (Target : in out Variable_String; Source : Character);
procedure Move (Target : in out Variable_String;
  Source : in out Variable_String);
function Image (V : Variable_String) return String;
-- Value function with maximum length = current length
function Value (S : String) return Variable_String;
-- Value function with specified maximum length
function Value (S : String; Max_Length : Natural) return Variable_String;
procedure Free (V : in out Variable_String);
procedure Append (Target : in out Variable_String;
  Source : Variable_String);
procedure Append (Target : in out Variable_String; Source : String);
procedure Append (Target : in out Variable_String; Source : Character);
procedure Append (Target : in out Variable_String;
  Source : Character;
  Count : Natural);
procedure Insert (Target : in out Variable_String;
  At_Pos : Positive;
  Source : Variable_String);
procedure Insert (Target : in out Variable_String;
  At_Pos : Positive;
  Source : String);
procedure Insert (Target : in out Variable_String;
  At_Pos : Positive;
  Source : Character);
procedure Insert (Target : in out Variable_String;
  At_Pos : Positive;
  Source : Character;
  Count : Natural);
procedure Delete (Target : in out Variable_String;
  At_Pos : Positive;
  Count : Natural := 1);
procedure Replace (Target : in out Variable_String;
  At_Pos : Positive;
  Source : Character);
procedure Replace (Target : in out Variable_String;
  At_Pos : Positive;
  Source : Character);

```

Bounded\_String, !Tools

```

Count : Natural);
procedure Replace
  (Target : in out Variable_String;
   At_Pos : Positive;
   Source : String);
procedure Replace
  (Target : in out Variable_String;
   At_Pos : Positive;
   Source : Variable_String);

procedure Set_Length (Target : in out Variable_String;
  New_Length : Natural;
  Fill_With : Character := ' ');
-- Truncate or extend with fill

function Length (Source : Variable_String) return Natural;
-- Get information about or contents of a string

function Char_At (Source : Variable_String; At_Pos : Positive)
  return Character;

function Extract (Source : Variable_String;
  Start_Pos : Positive;
  End_Pos : Natural) return String;

function Max_Length (Source : Variable_String) return Natural;
-- get the allocated length of the string

private
type Variable_String (Maximum_Length : String_Length) is
  record
    Length : String_Length := 0;
    Contents : String (1 .. Maximum_Length);
  end record;
end Bounded_String;

```

```

with Profile;
package Check is

-- Check runtime compatibility on the R1000 of individual units,
-- lists of units, views, directories, or activities.

type Status is (Identical, Upward-Compatible, Incompatible, Error);

subtype Unit_Name is String;
subtype Activity_Name is String;

subtype Response_Profile is Profile.Response_Profile;

Current_Activity : constant Activity_Name := "<ACTIVITY>";

-- Check whether the exported functionality of Name1 is
-- implemented by the corresponding implementation.

-- If Name2 is specified, check Name1 against Name2; otherwise, check
-- Name1 against its correspondent in the provided activity.
-- Wildcards specified in Name1 cause checking of against Name2 of each
-- object specified by Name1.

-- Name1 may be fully qualified (starts with | or some other character
-- recognized by the environment) or may be relative to the current context
-- or to Name2.

-- Activity => "" or "<ACTIVITY>" (default) means to use the current activity
-- otherwise, Activity specifies the name of an activity file.

-- If Closure is true, also compatibility check all units used by the
-- requested objects.

-- Examples:

Check_Units (Name1 => "Check", Closure => True);
Determine_Whether_Check (and everything required for it to run) is
compatible with the implementation specified in the current activity

Check_Units (Name1 => "!Tools.Compatibility.Rev9_2_Spec",
  Name2 => "!Tools.Compatibility.Rev9_1_Working");
Since both names are specified, the activity is unused. If Closure
were specified, the activity would be used for determining the
compatibility of units in the closure. Specifying a view name is
equivalent to specifying "view_name?Spec"; it means "check
all ADA specs in the view".

Check_Units (Name1 => "!Subsys.Rev9_1_spec.Pack1",
  Name2 => "!Subsys.Rev9_1_Working")
Specifying names of a unit and a view means "check the unit in the
spec view against the corresponding unit in the load view".

Check_Units (Name1 => "Pack1",
  Name2 => "!Subsys.Rev9_1_Working");
Assuming that we are not in a context where "Pack1" is recognized:
The name "Pack1" is resolved in the context specified by Name2, resulting
in the object "!Subsys.Rev9_1_Working.Pack1" which is a unit in a load
view. Name2 is also used to specify the object to check Name1 against.
However, instead of checking the load view unit "Pack1" against itself

```

```

-- compatibility checker substitutes the unit "pack1" in the spec view,
-- which it check against the unit "Pack1" in the load view. The activity
-- is used to find the spec view corresponding to the load view.
--
-- Check_Units (Name1 => "Machine.Release.Current.Activity");
-- Since Name1 specifies an activity, check the consistency of the
-- activity's view pairs. In this case, Closure, Name2, and Activity
-- are meaningless.
--
-- Check_Units (Name1 => "<ACTIVITY>");
-- Check the consistency of the current default activity's view pairs.
--
-- Check_Units (Name1 => "");
-- The standard resolution of "" is the enclosing library, so this
-- checks @Spec against the current activity. (Specifying a directory
-- is equivalent to specifying all ADA specs in the directory).
--
-- procedure Check_Units (Name1 : Unit_Name := ">>Name<<";
--                        Name2 : Unit_Name := "";
--                        Activity : Activity_Name := "<ACTIVITY>";
--                        Closure : Boolean := False;
--                        Response : Response_Profile := Profile.Get);
--
-- function Check_Units
-- (Name1 : Unit_Name := ">>Name<<";
--  Name2 : Unit_Name := "";
--  Activity : Activity_Name := "<ACTIVITY>";
--  Closure : Boolean := False;
--  Response : Response_Profile := Profile.Get) return Status;
--
end Check;

```

```

generic
  Size : Integer;
  -- number of buckets

  type Domain_Type is private;
  type Range_Type is private;
  -- both types are pure values
  -- no initialization or finalization is necessary
  -- = and := can be used for equality and copy

  with function Hash (Key : Domain_Type) return Integer is <>;
  -- for efficiency, spread hash over an interval at least as size
  pragma Must_Be_Constrained (Yes => Domain_Type, Range_Type);

package Concurrent_Map_Generic is

  pragma Subsystem (Tools);
  pragma Module_Name (4, 3984);

  type Map is private;

  type Pair is
  record
    D : Domain_Type;
    R : Range_Type;
  end record;

  function Eval (The_Map : Map; D : Domain_Type) return Range_Type;
  procedure Find (The_Map : Map;
                 D : Domain_Type;
                 R : in out Range_Type;
                 Success : out Boolean);

  procedure Find (The_Map : Map;
                 D : Domain_Type;
                 P : in out Pair;
                 Success : out Boolean);

  procedure Define (The_Map : in out Map;
                   D : Domain_Type;
                   R : Range_Type;
                   Trap_Multiples : Boolean := False);

  procedure Undefine (The_Map : in out Map; D : Domain_Type);

  procedure Initialize (The_Map : out Map);
  function Is_Empty (The_Map : Map) return Boolean;
  procedure Make_Empty (The_Map : in out Map);

  procedure Copy (Target : in out Map; Source : Map);

  type Iterator is private;

  procedure Init (Iter : out Iterator; The_Map : Map);
  procedure Next (Iter : in out Iterator);
  function Value (Iter : Iterator) return Domain_Type;
  function Done (Iter : Iterator) return Boolean;

  Undefined : exception;
  -- raised by eval if the domain value is not in the map

Concurrent_Map_Generic, !Tools

```

```

Size : Natural := 0;
end record;

end Cocurrent_Map_Generic;

```

```

Multiply_Defined : exception;
-- raised by define if the domain value is already defined and
-- the trap_multiples flag has been specified (ie. is true)

function Nil return Map;
function Is_Nil (The_Map : Map) return Boolean;

function Cardinality (The_Map : Map) return Natural;

----- Implementation Notes and Non-Standard Operations -----
-- := and = operate on references
-- := implies sharing (introduces an alias)
-- = means is the same map, not the same value of type map
-- Initializing a map also makes it empty
-- Maps must be initialized before use.
-- garbage may be generated
-- Concurrent Properties
-- any number of find/eval/is_empty/copy may be safely done while one
-- define/undefine/make_empty is taking place.
-- Define/undefine/make_empty operations are serialized.
-- Iterators may be used asynchronously, however the sequence of values
-- yielded may never have been in the map at any one time.

private
type Node;
type Set is access Node;
type Node is
record
Value : Pair;
Link : Set;
end record;
type Map_Data;
type Map is access Map_Data;
type Iterator is
record
The_Map : Map;
Index_Value : Natural;
Set_Iter : Set;
Done : Boolean;
end record;
subtype Index is Integer range 0 .. Size - 1;
type Table is array (Index) of Set;
type Map_Data is
record
Cache : Set;
Bucket : Table;
-- of at most one node

```

```

function Get_Raise_Location (Fully_Qualify : Boolean := True;
                             Machine_Info : Boolean := False) return String;
-- return a string representation of the location of the exception
-- most recently executed by the calling task. Get_Raise_Location
-- must be called either directly or indirectly from an exception
-- handler. If no exception is found or other problems encountered,
-- the null string is returned.

generic
  type T is limited private;
  -- Type that will be displayed using Image procedure as part
  -- of debugger's object display procedure.

  with function Image (Value : T;
                      Level : Natural;
                      Prefix : String;
                      Expand_Pointers : Boolean) return String;

  -- Given the Value, return its image. Level specifies the number
  -- of levels to detail to be displayed. Expand_Pointers indicates
  -- that internal pointers should be expanded in the image.
  -- Level = 0 => entire object should be elided.

  -- If any ASCII.LF's are emitted, the following line should
  -- start with Prefix as an "indent" value.

procedure Register;
-- Make this special display procedure known to the debugger
-- of the session making the call.

generic
  type T is limited private;
  procedure Un_Register;
  -- Remove the display procedure from the calling session's
  -- database of special types for the type T given.

pragma Subsystem (Native_Debugger);
pragma Module_Name (4, 3802);

end Debug_Tools;

```

T-10

```

package Debug_Tools is
  procedure Debug_On;
  procedure Debug_Off;
  -- Enable or disable debugging for the calling task's job. When enabled,
  -- only tasks that are descendants of the caller can be debugged.
  -- When debugging is disabled, the task is released to execute, and all
  -- active debugger "hooks" are deactivated (eg, breakpoints, etc).

  function Debugging return Boolean;
  -- return true if calling task is being debugged.

  procedure Message (Info : String);
  -- Print the message string in the debugger window. No operation if
  -- the debugger is not activated

  procedure User_Break (Info : String);
  -- "Break" in the debugger. The calling task stops as though it
  -- encountered a breakpoint. If the debugger is not active, no action
  -- is performed. Otherwise, the task remains stopped until the
  -- debugger user explicitly continues its execution.

  procedure Set_Task_Name (Name : String);
  function Get_Task_Name return String;
  -- Set or retrieve a string "synonym" for the calling task. This name
  -- is used within the debugger to make identifying task easier.
  -- It is also useful for multiple instances of the same task type
  -- to distinguish themselves in the debugger.
  -- No operation if the debugger is not activate.

  function Ada_Location (Frame_Number : Natural := 0;
                        Fully_Qualify : Boolean := True;
                        Machine_Info : Boolean := False) return String;
  -- Return a string name for the Ada location of execution in the
  -- specified stack frame. Frame_Number = 0 refers to the caller
  -- of Ada_Location. Frame_Number = 1 refers to its caller, and so
  -- on. The null string is returned if the frame is nonexistent or
  -- its location cannot be found for some other serious reason.

  -- This procedure works independent of whether there is an active
  -- debugger for the calling tasks, but it may return less information
  -- if there is not.

  function Get_Exception_Name (Fully_Qualify : Boolean := True;
                               Machine_Info : Boolean := False) return String;
  -- return a string representation of the exception most recently
  -- executed by the calling task. Get_Exception_Name must be called
  -- either directly or indirectly from an exception handler. If
  -- no exception is found, the null string is returned.

```

Debug\_Tools, !Tools



```

with Diana;
package Diana_Object_Editor is

-- type safe interfaces for diana types --
procedure Edit (Tree : Diana.Tree);
procedure Edit (Seq_Type : Diana.Seq_Type);
procedure Edit (Sequence : Diana.Sequence);
procedure Edit (Temp_Seq : Diana.Temp_Seq);

-- unsafe interfaces, segment and offset may not be of the right type --
procedure Edit_Tree (Segment, Offset : Long_Integer);
procedure Edit_Seq_Type (Segment, Offset : Long_Integer);
procedure Edit_Sequence (Segment, Offset : Long_Integer);
procedure Edit_Temp_Seq (Segment, Offset : Long_Integer);

-- image functions not provided by r1000 (native) diana --
function Image (Attr_Name : Diana.Attr_Name) return String;

-- functions to read an image --
function Current_Image return Diana.Tree;
function Current_Cursor return Diana.Tree;
function Current_Selection return Diana.Tree;

pragma Subsystem (Object_Editor);
pragma Module_Name (4, 2202);

end Diana_Object_Editor;

```

```

with Action;
with Calendar;
with Diana;
with Directory;
with Error_Messages;
with Profile;
with String_Table;

package Directory_Tools is
-- DIRECTORY TOOLS ORGANIZATION

-- The Directory system provides the structure for storing, managing
-- and naming objects.

-- Each object has a class, which determines the operations that can
-- be applied to object. Program units belong to the class Ada.
-- Libraries, the building blocks of the directory system, are
-- objects of class Library. The class reflects which object manager
-- manages objects of that type, as well as reflecting the type.

-- Except for objects of class Library, each Directory Object has
-- one or more versions, which can be selected by using the
-- appropriate version name.

-- Ada_Unit and Polymorphic_IO.File (and others) are ultimately of
-- type Directory_Object and provide type specific operations. The
-- general paradigm is that type independent operations (traversal,
-- create, copy, destroy, etc.) are provided in directory, while
-- type specific operations are provided by the packages
-- (Directory_Ada, Polymorphic_io, etc.) which introduce specific
-- managed types.

-- No exceptions are propagated from this subsystem, except those
-- associated with type specific operations.

-- . Package Object. Defines the object handle type and the
-- iterator on object types.

-- . Package Naming. Provides facilities for establishing a context
-- for name resolution and facilities for resolving string names.

-- . Package Traversal. Operations for traversing the directory
-- structure (which extends through all Ada units in the system).

-- . Package Any_Object. Standard Directory operations for
-- Creating, Freezing, Destroying and Copying managed objects in the
-- Directory.

-- . Package Library_Object. Defines operations specific to
-- objects of class library. Defines a Library object as a
-- distinguished point (World or Directory) in the directory system.
-- A world specifies the disk volume for storing its contents and
-- the policies which will apply to its contents.

-- . Package Ada_Object. Defines an Ada Unit as a kind of
-- Directory_Object. Provides type-specific operations for
-- constructing and manipulating Ada Units.

```

```

-- Package Statistics. Queries about Directory Objects.
-- This package is the main interface to the directory subsystem.
--
-- Package Object.Low_Level. Defines interface between Object. Handles
-- and the low level types of Directory_Implementation.
--
-- Package Ada_Implementation. Defines operations for gaining
-- access to Diana Trees from Object.Handles;
package Object is
package DI renames Directory;
package St renames String_Table;
-- Herein are defined the principle structures for accessing the
-- Directory System objects: Object.Handle and Object.Iterator
type Handle is private;
-- Objects in the directory system and the Versions of those
-- Objects are accessed via an Object.Handle. A handle may denote
-- all Versions of an Object collectively or a specific Version of
-- an Object. Most operations operate on specific Versions of an
-- Object. If the Object.Handle passed to the operation denotes no
-- specific Version, the Default Version is used.
function Nil return Object.Handle;
function Is_Nil (The_Object : Object.Handle) return Boolean;
function Hash (The_Object : Object.Handle) return Integer;
function Unique (The_Object : Object.Handle) return Long_Integer;
function Image (The_Object : Object.Handle;
Level : Natural;
Prefix : String;
Expand_Pointers : Boolean) return String;
-- See debug_tools.special_display
function Version (The_Object : Object.Handle) return String;
-- Returns a name of the form "V(nn)", for the Version denoted by the
-- Object.Handle.
function Same_Object (Left, Right : Object.Handle) return Boolean;
function Equal (Left, Right : Object.Handle) return Boolean
renames Same_Object;
-- Compare two handles to see if they refer to the same directory
-- entity. Please note, "=" will give incorrect results
--
--
type Class_Enumeration is new Natural range 0 .. 63;
Unknown_Class : constant Class_Enumeration := 0;
Library_Class : constant Class_Enumeration := 1;
Ada_Class : constant Class_Enumeration := 2;
File_Class : constant Class_Enumeration := 3;
User_Class : constant Class_Enumeration := 4;
Session_Class : constant Class_Enumeration := 5;

```

```

Pipe_Class : constant Class_Enumeration := 6;
Terminal_Class : constant Class_Enumeration := 7;
Tape_Class : constant Class_Enumeration := 8;
--
function Class (The_Object : Object.Handle) return Class_Enumeration;
function Equal (Class1, Class2 : Class_Enumeration) return Boolean;
function Image (The_Class : Class_Enumeration) return String;
function Value (S : String) return Class_Enumeration;
--
type Subclass is private;
function Nil return Subclass;
function Is_Nil (The_Subclass : Subclass) return Boolean;
function Unique (The_Subclass : Subclass) return Integer;
function Subclass_Of (The_Object : Object.Handle) return Subclass;
function Image (The_Subclass : Subclass) return String;
function Value (S : String) return Subclass;
function Class_Of (The_Subclass : Subclass) return Class_Enumeration;
--
--
type Iterator is private;
-- Representation of an ordered set of Objects (Object.Handles);
procedure Next (Iter : in out Object.Iterator);
function Done (Iter : Object.Iterator) return Boolean;
function Value (Iter : Object.Iterator) return Object.Handle;
procedure Reset (Iter : Object.Iterator);
-- reset the iterator to the beginning of the list.
function Nil return Object.Iterator;
function Is_Nil (Iter : Object.Iterator) return Boolean;
function Image (The_Iterator : Object.Iterator;
Level : Natural;
Prefix : String;
Expand_Pointers : Boolean) return String;
function Create return Object.Iterator;
-- create a new (empty) iterator. Note: an empty iterator is different
-- than a 'nil' iterator
function Has (Iter : Object.Iterator; An_Object : Object.Handle)
return Boolean;
procedure Add (Iter : Object.Iterator;
An_Object : Object.Handle;
Duplicate : out Boolean;
Before : Object.Handle := Object.Nil);
-- The given Object is added to the Iterator just before the object
-- denoted by the Before parameter. If the Before parameter is not
-- found, the object is added at the end of the list of Objects.
procedure Remove (Iter : Object.Iterator;
An_Object : Object.Handle);

```

```

Found : out Boolean);

-- The specified object is removed from the iterator if it is there.
procedure Invert (Iter : Object.Iterator);
-- reverse the ordering of the given object list.
-----
-- procedures and functions to handle errors
type Error_Code is private;
-- All procedures return an object.Code, which describes the
-- success or failure of the operation.
function Err_Code (The_Object : Object.Handle) return Object.Error_Code;
function Err_Code (The_Objects : Object.Iterator)
return Object.Error_Code;

-- Object handles and Iterators contains an object.Code for the last
-- action performed on them. This error code is the only way problems
-- are reported by the Directory System functions. This error code is
-- a copy of the error code returned by Directory.System.procedures.
-- Procedures and functions within the Directory System propagate bad
-- error codes from their input to their outputs. Except for this
-- propagation of error code, procedures passed bad objects are no-ops.
function Is_Bad (Error_Code : Object.Error_Code) return Boolean;
function Is_Bad (The_Object : Object.Handle) return Boolean;
function Is_Bad (The_Objects : Object.Iterator) return Boolean;

function Is_Ok (Error_Code : Object.Error_Code) return Boolean;
function Is_Ok (The_Object : Object.Handle) return Boolean;
function Is_Ok (The_Objects : Object.Iterator) return Boolean;

-- Test the object.Code for Success/Failure status
function Message (Error_Code : Object.Error_Code) return String;
function Message (The_Object : Object.Handle) return String;
function Message (The_Objects : Object.Iterator) return String;
function Message (Error_Code : Object.Error_Code) return St.Item;
function Message (The_Object : Object.Handle) return St.Item;
function Message (The_Objects : Object.Iterator) return St.Item;

-- Transforms the object.Code into an English explanation of the
-- Error.
procedure Report (Error_Code : Object.Error_Code;
Response : Profile.Response_Profile := Profile.Get);
procedure Report (The_Object : Object.Handle;
Response : Profile.Response_Profile := Profile.Get);
procedure Report (The_Objects : Object.Iterator;
Response : Profile.Response_Profile := Profile.Get);

-- If the object.Code is Bad, a message is formulated from the code a
-- the current Log device. If requested by the given response profile,
-- the exception Failure or Abandon is raised.

```

```

-- If the Code is Bad, a message is formulated from the code and sent
-- the current Log device. If requested by the given response profile,
-- the exception Failure or Abandon is raised.
Error : exception;
-- Raised by Report in the event of an error when the given Response
-- Profile asks that an exception be propagated to the caller.
Abandon : exception;
-- Raised by Report in the event of an error when the given response
-- Profile asks that the operation be Abandoned without propagating
-- exceptions to the caller.
-----
-- Depending on the operation that generated a bad error code, additi
-- useful information may be associated with an error code. First, each
-- error code is assigned to one of the following categories:
type Category_Enumeration is
(Successful,
-- No problems encountered.
Warning,
-- Some non-fatal error.
Lock_Error,
-- Some synchronization error occurred,
-- usually failure to acquire access to some
-- object within the specified maximum delay
Semantic_Error,
-- An operation requiring (Ada) semantic
-- consistency discovered semantic errors.
Code_Generation_Error,
-- An error was detected during cg.
Obsolescence_Error,
-- A change was prevented because it
-- obsolesced installed declarations.
Bad_Tree_Parameter,
-- An actual tree parameter failed to meet
-- the requirements of the formal subtype.
Illegal_Operation,
-- The attempted operation is not legal
-- when applied to the given parameters.
Consistency_Error,
-- The operation is inconsistent with the
-- current state of the universe.
Version_Error,
-- The specified version does not exist.
Policy_Error,
-- The operation violates some other policy
-- that applies at this point.
Bad_Naming_Context,
-- The context was not a valid context for
-- name resolution.
Ill_Formed_Name,
-- The name was not well formed lexically or

```

```

-- syntactically.
Undefined_Name,
-- The name could not be found in the given
-- context.
Ambiguous_Name,
-- Because of overloading or wildcards, the
-- name resolved to more than one entity.
Name_Error,
-- other errors occurred resolving a name.
Access_Error,
-- The operation violates access control
-- policies.
Class_Error,
-- The class of the object passed to the
-- operation is incompatible with op
-- either because the op expects a
-- particular class, or because the
-- op is a type independent op which
-- is not supported for the given class.
-- various selection errors
No_Selection, Cursor_Not_In_Selection,
Selections_Not_Supported, No_Declaration, No_Object, No_Editor,
Other_Error);
-- When all else fails
function Category (Error_Code : Object.Error_Code)
return Object.Category_Enumeration;
-- Extracts from each error code, the category it belongs to.
-----
-- Error codes in the category Semantic_Error and Code_Generation_Error
-- (and perhaps others) may have a list of error messages associated
type Message_List is private;
type Severity_Enumeration is (Note, Warning, Lra_Error,
Internal_Error, Exception_Handled);
function Severity (Result : Object.Message_List)
return Severity_Enumeration;
function Message (Result : Object.Message_List) return String;
function Message (Result : Object.Message_List) return St.Item;
-- Properties of the current message in the list
function Next (Result : Object.Message_List) return Object.Message_List;
function Done (Result : Object.Message_List) return Boolean;
function Nil return Object.Message_List;
procedure Report (Messages : Object.Message_List;
Response : Profile.Response_Profile := Profile.Get;
Top_Only : Boolean := False);
-- Displays the error messages in the standard format according to the
-- supplied profile. If Top_Only is true, only the first message in the
-- list will be displayed; other wise all messages in the list will be
-- Displayed.

```

```

function Messages (Error_Code : Object.Error_Code)
return Object.Message_List;
-- Extracts the message list from the error code. Returns the Nil list
-- if there are no messages.
-----
-- Error codes in the category Obsolescence_Error may have a list of
-- the objects that would have been obsolesced.
function Change_Impact (Error_Code : Object.Error_Code)
return Object.Iterator;
-- Extracts the list of obsolesced objects from the error code. Returns
-- the Nil iterator if there are none.
function Modified_Units
(Error_Code : Object.Error_Code) return Object.Iterator;
-- Extracts the list of units that were implicitly coded/uncoded by the
-- operation that returned the error code.
function Value (Category : Object.Category_Enumeration;
Message : String := "");
Messages : Object.Message_List := Object.Nil;
Change_Impact : Object.Iterator := Object.Nil;
Modified_Units : Object.Iterator := Object.Nil)
return Object.Error_Code;
function Value (Category : Object.Category_Enumeration;
Message : St.Item := St.Nil;
Messages : Object.Message_List := Object.Nil;
Change_Impact : Object.Iterator := Object.Nil;
Modified_Units : Object.Iterator := Object.Nil)
return Object.Error_Code;
function Nil return Object.Error_Code;
-- Construct an Error code of a the given class and associated message.
function Image (The_Code : Object.Error_Code;
Level : Natural;
Prefix : String;
Expand_Pointers : Boolean) return String;
-----
package Low_Level is
-- lower level routines to build and extract from handles,
-- error codes, and iterators
-----
-- Routines to set and retrieve the default action. This action is
-- the action with a handle if one is not explicitly supplied.
-- default action is never finished. It is initially the

```

```

-- null action id.
-- Directory_tools routines take a handle as an argument will use
-- the action from the handle. If source and destination are
-- both arguments, the action comes from the destination.
-- If the routine takes a name and context, the action from the
-- context is used. If neither these are true, the default action
-- is used directly.

-- Handles created by the directory system are created with the
-- default action, with three exceptions. The first exception is
-- traversal.recursion. This routine propagates the action
-- from the supplied handle instead of using the default.

-- The second exception is ada_implementation.open, which
-- will start an action if one is not supplied and the access
-- mode is READ. This action is finished by closing the unit.

-- The third exception is library_object.create, which
-- will always start and finish an action, making the addition
-- permanent. The user must manually back out by destroying any
-- created directories if the desire is to abandon the operation.
-- After the library is created, the action is set to the
-- default action. As such, objects created in it will be covered
-- by the default in force at the time the directory is created.

procedure Set_Default_Action_Id (The_Action : Action.Id);

function Default_Action_Id return Action.Id;
-----

-- object handle routines. These extract the underlying
-- components supplied by the environment directory system.
-- If no action is supplied, the action from the handle is used.
-- If the handle's action is NULL, the default action is used.

procedure Get_Declaration (Handle : Object.Handle;
  The_Decl : out Di.Declaration;
  Status : out Di.Error_Status;
  Action_Id : Action.Id := Action.Null_Id;
  Max_Wait : Duration := Di.Default_Wait);

procedure Get_Object (Handle : Object.Handle;
  The_Object : out Di.Object;
  Status : out Di.Error_Status);

procedure Get_Version (Handle : Object.Handle;
  The_Version : out Di.Version;
  Status : out Di.Error_Status;
  Action_Id : Action.Id := Action.Null_Id;
  Max_Wait : Duration := Di.Default_Wait);

procedure Get_Root (Handle : Object.Handle;
  The_Root : out Diana.Tree;
  Status : out Di.Error_Status;
  Action_Id : Action.Id := Action.Null_Id;
  Max_Wait : Duration := Di.Default_Wait);

function Get_Class (Handle : Object.Handle) return Di.Class;

```

```

function Get_Subclass (The_Subclass : Subclass) return Di.Subclass;
function Get_Subclass (Handle : Object.Handle) return Di.Subclass;

-- return the string supplied as the object name.

function Object_Name (Handle : Object.Handle) return String;
function Object_Name (Handle : Object.Handle) return St.Item;

function Action_Id (Handle : Object.Handle) return Action.Id;
function Finish_Action_On_Close
  (Handle : Object.Handle) return Boolean;

-- constructors to make object.handles

procedure Set_Class (Handle : Object.Handle; Class : Di.Class);

procedure Set_Object_Name
  (Handle : Object.Handle; The_Name : String);

procedure Set_Object_Name
  (Handle : Object.Handle; The_Name : St.Item);

procedure Set_Error_Code (Handle : Object.Handle;
  The_Code : Object.Error_Code);

procedure Set_Action_Id (Handle : Object.Handle;
  The_Action : Action.Id;
  Finish_Action_On_Close : Boolean := False);

procedure Set_Root (Handle : Object.Handle; The_Root : Diana.Tree);

procedure Set_Handle_Data
  (Handle : Object.Handle;
  The_Error : Object.Error_Code := Object.Nil;
  The_Name : String := "";
  The_Object : Di.Object := Di.Nil;
  The_Version : Di.Version := Di.Nil;
  The_Class : Di.Class := Di.Nil;
  The_Declaration : Di.Declaration := Diana.Empty;
  The_Root : Diana.Tree := Diana.Empty;
  The_Action : Action.Id :=
    Object.Low_Level.Default_Action_Id);

function Make_Handle
  (The_Error : Object.Error_Code;
  The_Name : String := "";
  The_Object : Di.Object := Di.Nil;
  The_Version : Di.Version := Di.Nil;
  The_Class : Di.Class := Di.Nil;
  The_Declaration : Di.Declaration := Diana.Empty;
  The_Root : Diana.Tree := Diana.Empty;
  The_Action : Action.Id :=
    Object.Low_Level.Default_Action_Id)
  return Object.Handle;

function Make_Handle
  (The_Error : Object.Error_Code;
  The_Name : St.Item := St.Nil;
  The_Object : Di.Object := Di.Nil;
  The_Version : Di.Version := Di.Nil;
  The_Class : Di.Class := Di.Nil;
  The_Declaration : Di.Declaration := Diana.Empty);

```

```

The_Root : Diana.Tree := Diana.Empty;
The_Action : Action.Id :=
  Object.Low_Level.Default_Action_Id)
return Object.Handle;

-- object iterator constructors

procedure Set_Error_Code (Iter : Object.Iterator;
  Code : Object.Error_Code);

procedure Set_Pattern (Iter : Object.Iterator; Pattern : String);
procedure Set_Pattern (Iter : Object.Iterator; Pattern : St.Item);
function Pattern (Iter : Object.Iterator) return String;
function Pattern (Iter : Object.Iterator) return St.Item;

-- If possible, the iterator uses the environment iterators.
-- To do this, constructors are supplied for the types of interest.

function Make_Iterator
  (Code : Object.Error_Code;
  An_Object : Object.Handle;
  Pattern : String := "");
The_Class : Object.Class_Enumeration :=
  Object.Unknown_Class) return Object.Iterator;

function Make_Iterator
  (Code : Object.Error_Code;
  A_Naming_Iter : Di.Naming.Iterator;
  Pattern : String := "");
The_Class : Object.Class_Enumeration :=
  Object.Unknown_Class) return Object.Iterator;

function Make_Iterator
  (Code : Object.Error_Code;
  A_Version_Iter : Di.Traversal.Version.Iterator;
  Pattern : String := "");
The_Class : Object.Class_Enumeration :=
  Object.Unknown_Class) return Object.Iterator;

function Make_Iterator
  (Code : Object.Error_Code;
  An_Object : Object.Handle;
  Pattern : St.Item := St.Nil;
  The_Class : Object.Class_Enumeration :=
  Object.Unknown_Class) return Object.Iterator;

function Make_Iterator
  (Code : Object.Error_Code;
  A_Naming_Iter : Di.Naming.Iterator;
  Pattern : St.Item := St.Nil;
  The_Class : Object.Class_Enumeration :=
  Object.Unknown_Class) return Object.Iterator;

function Make_Iterator
  (Code : Object.Error_Code;
  A_Version_Iter : Di.Traversal.Version.Iterator;
  Pattern : St.Item := St.Nil;
  The_Class : Object.Class_Enumeration :=
  Object.Unknown_Class) return Object.Iterator;

-- the class argument specifies a filter class. Only items of the
-- specified class will be returned. If unknown_class is given,
-- the filter is disabled.

```

```

-- routines that handle low level error code actions.

function Translate_Status (The_Status : Di.Error_Status)
  return Object.Category_Enumeration;
function Translate_Status (The_Status : Di.Naming_Name_Status)
  return Object.Category_Enumeration;

procedure Set_Category (Code : Object.Error_Code;
  Class : Object.Category_Enumeration);

procedure Set_Message (Code : Object.Error_Code; Msg : String);
procedure Set_Message (Code : Object.Error_Code; Msg : St.Item);
procedure Set_Message_List (Code : Object.Error_Code;
  List : Error_Messages.Errors);

end Low_Level;

private
type Handle_Data;
type Handle is access Handle_Data;
pragma Segmented_Heap (Handle);

type Iterator_Kind_Enum is (Version_Iter, Name_Iter, No_Iter);
type Iterator_Data (Iterator_Kind : Iterator_Kind_Enum);
type Iterator is access Iterator_Data;
pragma Segmented_Heap (Iterator);

type Error_Code_Data;
type Error_Code is access Error_Code_Data;
pragma Segmented_Heap (Error_Code);

type Message_List is new Error_Messages.Errors;

type Subclass is new Di.Subclass;
end Object;

package Naming is
  -- Provides mechanisms for manipulating and resolving names and for
  -- establishing a context for name resolution.
  subtype String_Name is String;

  -- Lexically and syntactically a Directory system string name.
  subtype Simple_String_Name is String;

  -- A single segment of a string name: an Ada identifier with or without
  -- attributes

  subtype Context is Object.Handle;

  -- The Directory System Object that serves as the initial context
  -- for name resolution. May be any Object

  procedure Set_Default_Context (The_Context : Naming.String_Name;
    Status : out Object.Error_Code);

```

```

procedure Set_Default_Context (The_Context : Naming.Context;
                             Status : out Object.Error_Code);
-- Establishes the default naming context for the job.
function Default_Context return Naming.String_Name;
function Default_Context return Naming.Context;

-- Returns the default name resolution context for the job.
function Is_Well_Formed (A_Name : String_Name) return Boolean;
-- Tests whether a name is lexically and syntactically valid.
function Prefix (The_Name : String_Name) return String_Name;
-- Removes the last segment from a selected name and returns
-- the prefix.
-- Prefix ("A.B.C") => "A.B"
-- Prefix ("A") => ""

function Simple_Name (The_Name : String_Name) return Simple_String_Name;
-- Returns only the last segment of a selected name, without attributes
-- Simple_Name ("A.B.C") => "C"
-- Simple_Name ("A") => "A"

function Head (The_Name : String_Name) return Simple_String_Name;
-- Returns only the first segment of a selected name.
-- Head ("A.B.C") => "A"
-- Head ("A") => "A"
-- Head ("IA") => "I"

function Tail (The_Name : String_Name) return String_Name;
-- Removes the first segment from a selected name and returns the tail.
-- Tail ("A.B.C") => "B.C"
-- Tail ("A") => ""

function Attributes (A_Name : String_Name) return String;
-- Returns the Attributes at the end of the given string name.
-- If the simple name of the given string name has no attributes,
-- the null string is returned. The returned string starts with ' '.

function Attribute
  (A_Name : String_Name; Kind : String := "C") return String;
-- Returns the argument of the attribute designated by the kind
-- parameter that appears in the simple name of the given name.
-- If no argument follows the named attribute, the name of the attribute
-- is returned. (Parentheses are not part of the returned string.) If
-- the named attribute does not appear, the null string is returned.

function Nickname_Attribute
  (A_Name : String_Name; Kind : String := "N") return String

```

```

renames Attribute;
function Class_Attribute
  (A_Name : String_Name; Kind : String := "C") return String
renames Attribute;
function Version_Attribute
  (A_Name : String_Name; Kind : String := "V") return String
renames Attribute;

-- returns the argument to the Nickname, Class and Version attributes
-- respectively.
function Part_Attribute (A_Name : String_Name) return String;
-- Returns either "SPEC" or "BODY" or the null string if neither
-- of these are present
function Expanded_Name (The_Name : Naming.String_Name;
                       Context : Naming.Context := Default_Context)
  return String_Name;
-- Expands any prefix characters in the name appropriately.

function Full_Name (The_Object : Object.Handle)
  return Naming.String_Name;
function Full_Name (The_Object : String_Name) return Naming.String_Name;
-- Computes the fully qualified string name for the The_Object
-- exclusive of qualifying attributes.
function Simple_Name (The_Object : Object.Handle)
  return Simple_String_Name;
-- Computes the simple name for the The_Object exclusive of qualifying
-- attributes. (= Simple_Name (Full_Name (The_Object)));
function Unique_Full_Name
  (The_Object : Object.Handle) return String_Name;
function Unique_Full_Name (The_Object : String_Name) return String_Name;
-- Full_Name with 'body', 'n()', and 'v()' attributes as needed.
function Unique_Simple_Name
  (The_Object : Object.Handle) return Simple_String_Name;
function Unique_Simple_Name
  (The_Object : String_Name) return Simple_String_Name;
-- Simple_Name with 'body', 'n()', and 'v()' attributes as needed.
function Ada_Name (The_Object : Object.Handle) return String_Name;
function Ada_Name (The_Object : String_Name) return String_Name;
-- Returns a valid ada name for an Object. (No extra attributes,
-- no library names, no "!", etc.)

function Resolution (Name : Naming.String_Name;
                    Context : Naming.Context := Default_Context;
                    Object_Only : Boolean := True)
  return Object.Handle;

```

```

-- Resolve name to a single Object. Wild cards may be used, but
-- the name must resolve to a unique Object.

function Resolution (Name : Naming.String_Name;
Context : Naming.Context := Default_Context;
Objects_Only : Boolean := True)
return Object.Iterator;

-- Resolves (ambiguous) Source name in the given context. If
-- Objects_Only is true, only (separate) Objects that match the
-- name will be included; when false, Ada declarations will
-- be included even if no separate Object is associated with them.
-- Resolution is more efficient if Objects_Only is true.

function Pattern (Iter : Object.Iterator) return String_Name;
function Pattern (Iter : Object.Iterator) return String_Table.Item;

-- Returns a string_name that describes the objects of the iterator. For
-- the iterator returned by resolution, this is the value of the Name
-- parameter.

function Has_Substitution_Characters
(Target : String_Name) return Boolean;

function Target_Name (Iter : Object.Iterator; Target : String_Name)
return String_Name;

-- Replaces the substitution characters in the given Target name
-- with the appropriate values derived from the current Object of
-- the iteration (using the Pattern of the iterator).

function Target_Name (The_Object : Object.Handle;
Pattern : String_Name;
Target : String_Name) return String_Name;

function Target_Name (Source : String_Name; Target : String_Name)
return String_Name;

-- Given an Object and a source name (with wild cards) that
-- matches the name of the The_Object, returns a target string in which
-- substitution characters have been replaced by the matching
-- portions of the The_Object's name as indicated by the source name
-- pattern.

function Nickname (Def_Id : Object.Handle) return String;
function Nickname (Def_Id : Naming.String_Name) return String;

-- Returns the user-defined nickname associated with Def_Id, if one has
-- been specified; returns the system-defined nickname otherwise.

function System_Nickname (Def_Id : Object.Handle) return String;
function System_Nickname (Def_Id : Naming.String_Name) return String;

-- returns the system-assigned nickname for the given Def_Id, whether
-- or not a user-defined nickname has been assigned. The

```

```

-- system-assigned nickname is the image of the ordinal position
-- (1-origin) of the def_id among its namespaces in its declarative
-- region.

function Is_Overloaded (Def_Id : Object.Handle) return Boolean;
function Is_Overloaded (Def_Id : Naming.String_Name) return Boolean;

-- returns true if the given Def_Id is an overloaded Ada declaration.

end Naming;

package Traversal is
-- Provides operations for traversing the Directory System
-- in a variety of ways.

function Position (The_Object : Naming.String_Name) return Natural;
function Position (The_Object : Object.Handle) return Natural;

-- The position of the object in its declarative context. The first
-- item is position '0'.

Default_Position : constant Natural := Natural'Last;

-- Specifies the end of the list.

function Universe return Object.Handle;

-- Returns the (somewhat special) Object corresponding to the
-- root of the universe.

function Parent (The_Object : Object.Handle) return Object.Handle;

-- Returns the parent Object for The_Object.

function Enclosing_World
(The_Object : Object.Handle) return Object.Handle;

function Enclosing_Library
(The_Object : Object.Handle) return Object.Handle;

-- Returns the nearest enclosing Library/World that contains
-- the specified Object.

function Associated_World
(The_Object : Object.Handle) return Object.Handle;

function Associated_Library
(The_Object : Object.Handle) return Object.Handle;

-- Returns the nearest enclosing Library/World that contains
-- the specified Object, but if the object is a Library/World, that
-- value is returned.

function Child (The_Object : Object.Handle;
Child_Name : Naming.Simple_String_Name)
return Object.Handle;

-- Retrieve the named subobject.

```



```

function Children (The_Object : Object.Handle;
Pattern : Naming.Simple_String_Name := "@";
Declared : Boolean := True;
Class : Object.Class_Enumeration :=
  Object.Unknown_Class) return Object.Iterator;

-- Initializes the iteration over the children that match the
-- specified name pattern. If Declared is True, only children
-- that are declared in the given Version of the Object will be
-- returned. If Declared is false, all existing children of the
-- Object are returned even if they have no stub declaration in the
-- given Version of the unit.
-- If a class is provided, only children of that class are returned

function Versions (The_Object : Object.Handle;
Forward : Boolean := True) return Object.Iterator;

-- Get all versions of the given object; iterator is ordered forward
-- or backward according to creation time based on the Forward boolean.

type Control_Enumeration is (Continue, Abandon_Level,
  Abandon_Recursion, Skip_Children);

generic
  type State_Record is private;
  with procedure Op (Depth : Positive;
    State : in out State_Record;
    The_Object : Object.Handle;
    Status : out Object.Error_Code;
    Control : in out Traversal.Control_Enumeration);
  procedure Recursion (State : in out State_Record;
    The_Object : Object.Handle;
    Status : out Object.Error_Code;
    Pattern : Naming.Simple_String_Name := "@";
    Class : Object.Class_Enumeration :=
      Object.Unknown_Class;
    Subunits : Boolean := True;
    Directories : Boolean := True;
    Worlds : Boolean := False;
    Objects_Only : Boolean := True;
    Deleted : Boolean := False);
  -- Performs a depth-first traversal of the Directory structure
  -- rooted at the given Object.
  -- The Boolean parameters control scope of the traversal as follows:
  -- Subunits : Subunits of Ada units are included
  -- Directories : Nested directories are included
  -- Worlds : Nested worlds are included
  -- Objects_Only : Only separate Ada objects are included
  -- Deleted : deleted objects are included
  -- The formal procedure Op is called for each Object visited. The
  -- State variable is passed from call to call. The traversal will
  -- terminate immediately if the error code parameter has a bad value
  -- when the Op procedure returns. This error code is returned as the
  -- error code of the Recursion procedure. Recursion can also be
  -- controlled by the Control parameter.

```

```

-- Only Objects with a simple name that matches the given Pattern are
-- visited.
-- Only Objects of the given Class (if not nil) are visited.
-- The Pattern and Class attributes do not affect the scope of the
-- traversal, just the Objects on which Op is called. For example, if
-- Class is Object.Class (Object.Ada), Op will be called for each Ada
-- object (including subunits) nested within the given object and within
-- any directory nested within the given Object and in the same world as
-- the given object.

generic
  type State_Record is private;
  with procedure Op (State : in out State_Record;
    The_Object : Object.Handle;
    The_Objects : out Object.Iterator;
    Status : out Object.Error_Code;
    Control : in out Traversal.Control_Enumeration);
  procedure Closure (State : in out State_Record;
    In_Objects : Object.Iterator;
    Out_Objects : out Object.Iterator;
    Status : out Object.Error_Code);
  -- Computes the transitive closure of the Op procedure applied to the
  -- input objects.
  -- The only control_enumeration values supported are
  -- Continue, Abandon_Recursion
  -- The objects in the iterator Out_Objects are in a known order. This
  -- order is such that if the "op" procedure were computing the closure
  -- of "with x;" statements, the objects in the iterator could be
  -- promoted in order without incurring 'uninstalled' errors. If the
  -- desire is to demote the objects, the list should be reversed.
  -- A formal way of stating the order is:
  -- For each object, operate on its children, then operate on the object
  -- The algorithm is:
  -- 1. result_iterator := object.create;
  -- 2. For each object in in_objects:
  --   i. If the object is not 'in' the result_iterator, then
  --     a. 'Visit' the object.
  --     b. Recurse, (step 2) with the result of visit as the
  --        new in_objects.
  --     c. add the object to the result_iterator.
  -- 3. out_Objects := result_iterator.

end Traversal;

package Any_Object is
  -- Operations to Create, Copy and Destroy Objects.
  -- Several versions of an object may exist simultaneously. One of
  -- these versions may be designated the default version, which will
  -- be the one accessed if no specific version is referenced. For
  -- Ada objects, only the default version may be at the installed
  -- state or higher. For an object in a library, if there is no

```

```

-- default version, there will be no declaration for the object
-- visible. (However, for an Ada subunit, a stub declaration may be
-- visible in the parent object even though there is no default
-- version for the subunit.)

-- The versions that are not the default version are called deleted
-- versions. The number of versions of an object that are retained in
-- the system is controlled by a retention count parameter, which may
-- be set for each object individually. When the number of deleted
-- versions exceeds the retention count (either because the count has
-- been changed, or additional versions are deleted), existing versions
-- are destroyed (oldest first) until the count is satisfied.

Default_Retention_Count : constant := -1;

-- A special retention count value that directs an operation to use the
-- existing count for the object or inherit one from the parent object.
function Retention_Count (The_Object : Object.Handle) return Natural;

procedure Set_Retention_Count
(The_Object : Object.Handle;
 Retention_Count : Integer := Default_Retention_Count;
 Status : out Object.Error_Code);

function Is_Visible (The_Object : Object.Handle) return Boolean;
function Is_Visible (The_Object : Naming.String_Name) return Boolean;

function Has_Versions (The_Object : Object.Handle) return Boolean;
function Has_Versions (The_Object : Naming.String_Name) return Boolean;

function Has_Default_Version
(The_Object : Object.Handle) return Boolean;
function Has_Default_Version
(The_Object : Naming.String_Name) return Boolean;

procedure Create (The_Object : Object.Handle;
 New_Version : out Object.Handle;
 Status : out Object.Error_Code);

-- Create a new version of an existing object. The new version becomes
-- the default version.

procedure Create (Object_Name : Naming.String_Name;
 New_Version : out Object.Handle;
 Status : out Object.Error_Code;
 Class : Object.Class_Enumeration :=
 Object.Unknown_Class;
 Context : Naming.Context := Naming.Default_Context;
 Position : Natural := Traversal.Default_Position;
 Subclass : Object.Subclass := Object.Nil);

-- Creates a new version of the named object, which becomes the default
-- version. If an Object does not
-- yet exist with that name, one is created at the indicated class.
-- The declaration for the object is placed at the indicated position in
-- its parent context.

procedure Copy (Source : Object.Handle;

```

```

Destination : Object.Handle;
 New_Version : out Object.Handle;
 Status : out Object.Error_Code);

-- The version of the Source Object specified by the Source handle is
-- copied to the Destination Object, where it becomes the default
-- version of the Destination Object.

procedure Copy (Source : Naming.String_Name;
 Destination : Naming.String_Name;
 New_Version : out Object.Handle;
 Status : out Object.Error_Code;
 Source_Context : Naming.Context :=
 Naming.Default_Context;
 Destination_Context : Naming.Context :=
 Naming.Default_Context;
 Position : Natural := Traversal.Default_Position;
 Subclass : Object.Subclass := Object.Nil);

-- Copies the value. Creates an entirely new declaration and Object
-- at the destination if one did not exist. If the declaration
-- existed, but the specified Version did not, creates
-- a new Version of the destination Object and makes it the default.
-- If the destination Version already
-- exists, overwrites the old value with the new. Copied Ada
-- units are source only, regardless of the state of the Source.
-- Copies only the Source Object (no sub-Objects).

procedure Delete (The_Object : Object.Handle;
 Status : out Object.Error_Code;
 Retention_Count : Integer := Default_Retention_Count);

procedure Delete (The_Object : Naming.String_Name;
 Status : out Object.Error_Code;
 The_Context : Naming.Context :=
 Naming.Default_Context;
 Retention_Count : Integer := Default_Retention_Count);

-- Deletes the default version of the specified Object. If,
-- after the deletion, the number of deleted versions exceeds the ret
-- count, the oldest version will be destroyed.
-- If the target of any delete is an installed Ada unit, first
-- attempts to withdraw the unit, then (if there were no errors)
-- performs the delete.

procedure Undelete (The_Object : Object.Handle;
 Status : out Object.Error_Code);

procedure Undelete (The_Object : Naming.String_Name;
 Status : out Object.Error_Code;
 The_Context : Naming.Context :=
 Naming.Default_Context);

-- Make the specified version of the given object the default version.
-- Reinstates the declaration (visibility) of the object if it had been
-- deleted.

procedure Expunge (The_Object : Naming.String_Name;
 Status : out Object.Error_Code;

```

```

Retention_Count : Integer := 0;
Context : Naming.Context := Naming.Default_Context);

procedure Expunge (The_Object : Object.Handle;
                  Status : out Object.Error_Code;
                  Retention_Count : Integer := 0);
-- Destroy deleted versions of an Object (oldest first) until
-- the number of deleted versions remaining is no more than the retention
-- count.

procedure Destroy (The_Object : Object.Handle;
                  Status : out Object.Error_Code;
                  Retention_Count : Integer :=
                    Default_Retention_Count);

procedure Destroy (The_Object : Naming.String_Name;
                  Status : out Object.Error_Code;
                  The_Context : Naming.Context :=
                    Naming.Default_Context;
                  Retention_Count : Integer :=
                    Default_Retention_Count);
-- Destroys the specified Version of the Object. If this is the
-- default version of the object, the declaration is deleted as well.
-- If the target of any destroy is an installed Ada unit, Destroy first
-- attempts to withdraw the unit, then (if there were no errors)
-- performs the destroy. After the destroy the object is expunged,
-- using the supplied retention count.

function Is_Frozen (The_Object : Naming.String_Name) return Boolean;
function Is_Frozen (The_Object : Object.Handle) return Boolean;
-- Test whether an Object is frozen. Frozen objects cannot be changed.

procedure Freeze (The_Object : Object.Handle;
                  Recursive : Boolean := False;
                  Status : out Object.Error_Code);
-- Freeze an Object or a unit (and its children which are in the
-- same control point) so that it cannot be changed.

procedure Unfreeze (The_Object : Object.Handle;
                   Recursive : Boolean := False;
                   Status : out Object.Error_Code);
-- Unfreeze an Object or a unit (and its children which are in
-- the same control point) so that it can be manipulated normally.

end Any_Object;

package Ada_Object is
-- Directory operations specific to the class ADA are defined here.
-- Objects of class Ada correspond to the notion of Compilation Unit
-- as defined by the LRM. These Ada units can be in one of six states:
type Unit_State is

```

```

(Nonexistent, Archived, -- text only; no Diana tree
Source, -- Source ready to be installed
Installed, -- Semantically consistent.
Coded -- Has been code generated.
);

function State (For_Unit : Object.Handle) return Ada_Object.Unit_State;
function State (For_Unit : Naming.String_Name)
return Ada_Object.Unit_State;

function Is_Source (The_Unit : Object.Handle) return Boolean;
function Is_Source (The_Unit : Naming.String_Name) return Boolean;

function Is_Installed (The_Unit : Object.Handle) return Boolean;
function Is_Installed (The_Unit : Naming.String_Name) return Boolean;

type Compilation_Kind is (Not_Class_Ada, Uncertain,
Library_Unit, Library_Unit_Body,
Subunit, Internal_Declaration);
-- Uncertain is returned for source units in which insufficient
-- text is present to determine it's kind. Internal_declaration
-- is returned for declarations with bodies contained in a
-- library unit

function Kind (Ada_Unit : Object.Handle) return Compilation_Kind;
function Kind (Ada_Unit : Naming.String_Name) return Compilation_Kind;

type Unit_Kind is (Not_Class_Ada, Uncertain, Function_Spec,
Function_Body, Procedure_Spec, Procedure_Body,
Package_Spec, Package_Body, Function_Instantiation,
Procedure_Instantiation, Package_Instantiation,
Generic_Function, Generic_Procedure, Generic_Package,
Function_Rename, Procedure_Rename, Package_Rename,
Task_Spec, Task_Type, Task_Body, Not_A_Unit);
-- More specific classification of type of Ada unit (declaration).

function Kind (Ada_Unit : Object.Handle) return Unit_Kind;
function Kind (Ada_Unit : Naming.String_Name) return Unit_Kind;

function Is_Visible_Part (Ada_Unit : Object.Handle) return Boolean;
function Is_Visible_Part (Ada_Unit : Naming.String_Name) return Boolean;
-- Determines whether the given unit corresponds to a visible part.

function Is_Subunit (Ada_Unit : Object.Handle) return Boolean;
function Is_Subunit (Ada_Unit : Naming.String_Name) return Boolean;

function Other_Part (Ada_Unit : Object.Handle) return Object.Handle;
-- Given the visible part, return the body, and vice versa. Returns
-- a nil unit if there is no complement.
-- May have to actually create the Object.

function Subunit (Ada_Unit : Object.Handle;
                  Subunit_Name : Naming.Simple_String_Name)
return Object.Handle;

```

```

-- Retrieve the named subunit.
function Subunits (Ada_Unit : Object.Handle;
  Pattern : Naming.Simple_String_Name := "@";
  Declared : Boolean := True) return Object.Iterator;

-- Computes a list of all subunits of the given unit whose name
-- matches the given Pattern. If Declared is True, only subunits
-- that are declared in the given Version of the unit will be
-- returned. If Declared is false, all existing subunits of the unit
-- are returned even if they have no stub declaration in the given
-- Version of the unit.
function Depends_On (Defining_Id : Naming.String_Name;
  The_Context : Naming.Context :=
    Naming.Default_Context) return Object.Iterator;
function Depends_On (Defining_Id_Handle : Object.Handle)
  return Object.Iterator;

-- Computes the set of ada units that depend upon the defining_id given.
-- A defining_id is the full name of the defining occurrence of the item,
-- and can be any ada object, from a package to a variable.
function List_Of_Withs (Ada_Unit : Naming.String_Name;
  The_Context : Naming.Context :=
    Naming.Default_Context)
  return Object.Iterator;
function List_Of_Withs (Ada_Unit_Handle : Object.Handle)
  return Object.Iterator;

-- computes the set of units 'with'ed by the supplied unit.
-----
-- Operations to promote and demote declarations. Promoting
-- declarations moves them "up" to higher declaration states (toward
-- Coded), while demotion moves declarations "down" to lower
-- declaration states (toward Nonexistent). Promoting to a lower
-- state or demoting to a higher state is an Illegal_Operation.

procedure Promote (Ada_Unit : Object.Handle;
  Status : out Object.Error_Code;
  Goal_State : Ada_Object.Unit_State :=
  Ada_Object.Installed;
  Switches : Object.Handle := Object.Nil);

procedure Promote (Ada_Unit : Naming.String_Name;
  Status : out Object.Error_Code;
  Goal_State : Ada_Object.Unit_State :=
  Ada_Object.Installed;
  Switches : Object.Handle := Naming.Default_Context);

-- A subunit may not be promoted to a state higher than that of
-- its parent, except that a subunit may be coded before the parent
-- is coded.

procedure Demote (Location : Object.Handle;
  Status : out Object.Error_Code;

```

```

Goal_State : Ada_Object.Unit_State :=
  Ada_Object.Source;
Switches : Object.Handle := Object.Nil);

procedure Demote (Location : Naming.String_Name;
  Status : out Object.Error_Code;
  Goal_State : Ada_Object.Unit_State :=
  Ada_Object.Source;
  Switches : Object.Handle := Object.Nil;
  Context : Naming.Context := Naming.Default_Context);

-- This operation will fail with obsolescence error if any
-- declarations (including installed subunits) depend upon
-- demoted declarations.
end Ada_Object;

package Library_Object is

-- Directory operations specific to Libraries. Unlike other objects,
-- there can be only one version of a Library object.

type Library_Kind is (Not_A_Library, Directory, World);

function Kind (Any_Object : Object.Handle)
  return Library_Object.Library_Kind;
function Kind (Any_Object : Naming.String_Name)
  return Library_Object.Library_Kind;

function Is_Library (Any_Object : Object.Handle) return Boolean;
function Is_Library (Any_Object : Naming.String_Name) return Boolean;

-- Returns true IFF the indicated object is an object of class Library

function Is_World (Any_Object : Object.Handle) return Boolean;
function Is_World (Any_Object : Naming.String_Name) return Boolean;

function Is_Directory (Any_Object : Object.Handle) return Boolean;
function Is_Directory (Any_Object : Naming.String_Name) return Boolean;

procedure Set_Switch_Object (The_Library : Object.Handle;
  The_File : Object.Handle;
  Status : out Object.Error_Code);
procedure Set_Switch_Object (The_Library : Naming.String_Name;
  The_File : Naming.String_Name;
  Status : out Object.Error_Code);

function Switch_Object
  (The_Library : Object.Handle) return Object.Handle;

function Switch_Object (The_Library : Naming.String_Name)
  return Naming.String_Name;

-- Used to manipulate the file Object used to store switch files.
subtype Volume_Id is Natural range 0 .. 31;
-- Used to represent a disk volume.

function Nil return Library_Object.Volume_Id;
function Is_Nil (The_Volume : Library_Object.Volume_Id) return Boolean;

```

```

function Volume (The_Library : Object.Handle)
return Library_Object.Volume_Id;

procedure Create (Name : Naming.Simple_String_Name;
Kind : Library_Object.Library_Kind;
New_Library : out Object.Handle;
Status : out Object.Error_Code;
Volume : Library_Object.Volume_Id :=
Library_Object.Nil;
Context : Naming.Context := Naming.Default_Context;
Position : Natural := Traversal.Default_Position;
Subclass : Object.Subclass := Object.Nil);

-- Creates a new Library (Directory or World) at the indicated
-- position. If an appropriate declaration already exists and has
-- no directory Object associated with it, that stub will be used
-- rather than creating a new one.
end Library_Object;

package Ada_Implementation is
subtype Root is Diana.Tree;
subtype Any_Node is Diana.Tree;

package AI renames Ada_Implementation;

function Is_Source (For_Node : Any_Node) return Boolean;
function Is_Installed (For_Node : Any_Node) return Boolean;

procedure Will_Be_A_Comp_Unit (Root : AI.Root;
Verdict : out Boolean;
Status : out Object.Error_Code);

-- A predicate which determines if the root of a child unit will
-- be promoted in place or made into a comp_unit.

-- procedure Replace_Comment (Node : Diana.Tree;
-- Pre_Comment : Comment_Definitions.Co
-- Status : Boolean := True;
-- Status : out Object.Error_Code);

-- Make New_Comment the Pre_/Post_Comment of Node.
-- Node must be in an installed unit.

type Open_Mode is
(None, -- Mode None only applies to installed units.
Read -- Mode Read applies to either source or installed
units, and acquires a non-exclusive read lock
(exclusive of update, but not other readers)
);

procedure Open (The_Unit : Object.Handle;
Mode : Open_Mode;
Root : out AI.Root;
Status : out Object.Error_Code);

-- Returns the root of the separate tree designated by The_Unit.
-- Opens the unit with the specified access Mode.
-- Incompatible access modes Error in queuing or Lock_Error.

-- Open and Close invoke policy specific pre and post operations

```

```

-- before and after execution.
-- Open first tries to open an object. If that fails, it
-- then tries to open the object containing the declaration.
-- In the latter case, the return value is the declaration within
-- the object.

procedure Close (The_Unit : Object.Handle;
Status : out Object.Error_Code);

-- Closes the indicated unit, releasing access.

procedure Get_Root (Node : Any_Node;
Root : out AI.Root;
Status : out Object.Error_Code);

-- Returns the Root of the unit represented by the Node.

procedure Get_Handle (Node : Any_Node;
Handle : out Object.Handle;
Status : out Object.Error_Code);

-- Returns the Object containing the Node.

end Ada_Implementation;

package Statistics is
subtype User is Object.Handle;
subtype Session is Object.Handle;

function Time_Of_Last_Update
(The_Object : Object.Handle) return Calendar.Time;

function Time_Of_Last_Read
(The_Object : Object.Handle) return Calendar.Time;

function Time_Of_Creation
(The_Object : Object.Handle) return Calendar.Time;

function Last_Updater (The_Object : Object.Handle) return User;

function Session_Of_Last_Updater
(The_Object : Object.Handle) return Session;

function Last_Reader (The_Object : Object.Handle) return User;

function Session_Of_Last_Reader
(The_Object : Object.Handle) return Session;

function Creator (The_Object : Object.Handle) return User;

function Session_Of_Creator (The_Object : Object.Handle) return Session;

function Total_Size (The_Object : Object.Handle) return Long_Integer;

function Header_Size (The_Object : Object.Handle) return Natural;

function Object_Size (The_Object : Object.Handle) return Long_Integer;
function Last_Edit_Time (The_Unit : Object.Handle) return Calendar.Time;

```

```

end Statistics;
pragma Subsystem (Cavc, Closed);
pragma Module_Name (4, 3525);
end Directory_Tools;

```

```

with Calendar;
with Machine;

```

```

package Disk_Daemon is
  pragma Subsystem (Disk_Cleaner);
  pragma Module_Name (4, 3401);

```

```

  subtype Volume_Number is Integer range 0 .. 31;
  subtype Task_Vpids is Machine.Job_Id;

```

```

  -- The procedural operations of this package are noops when given
  -- invalid parameters (out of range, volume does not exist, ...).
  -- The functions return 'first when given invalid parameters.

```

```

type Threshold_Kinds is
  (Start_Collection, -- Default 25%; start collection on this volume.
  Raise_Priority,    -- Default 15%; raise priority of all collection
  -- until this volume gets above this threshold.
  Stop_Jobs,        -- Default 10%; stop user jobs and max priority
  -- until this volume gets above this threshold.
  Suspend_System); -- Default 3%; suspend the system.

```

```

subtype Percentage is Natural range 0 .. 100;

```

```

function Exists (Volume : Volume_Number) return Boolean;

```

```

procedure Set_Threshold (Volume : Volume_Number;
  Kind : Threshold_Kinds;
  At_Remaining_Capacity : Percentage);

```

```

function Get_Threshold (Volume : Volume_Number; Kind : Threshold_Kinds)
  return Percentage;

```

```

function Capacity (Volume : Volume_Number) return Natural;

```

```

function Used_Capacity (Volume : Volume_Number) return Natural;

```

```

function Unused_Capacity (Volume : Volume_Number) return Natural;

```

```

  -- The capacity functions return the size in number of 1024 byte pages.

```

```

subtype Collection_Priority is Integer range -1 .. 6;

```

```

  -- -1 => Attempts to collect using just "spare cpu cycles"; if there are
  -- no spare cycles, will wait forever (or until some agent increases
  -- the priority).

```

```

  -- 0 => Slowest priority without backoff; runs on par with background jobs
  -- that do not use 'priority. Small impact on performance.

```

```

  -- 2 => Will preempt most background jobs. Runs on par with a background
  -- job that uses the best 'priority.

```

```

  -- 3 => Runs on par with most foreground jobs. Tends to have a big impact
  -- on performance, since it will compete with commands.

```

```

  -- 4 => Preempts most foreground jobs. Should still be able to edit.
  -- But commands will run VERY slowly.

```

```

  -- 6 => Preempts virtually all activity, except that from the console.

```

```

  -- Note that there is a policy function which places a lower bound on the

```

Disk\_Daemon, !Tools

-- current collection priority. See the Set\_Priority\_Policy operation,  
-- below.

```
procedure Perform_Garbage_Collection  
(Volume : Volume_Number;  
  Max_Wait : Duration;  
  Desired_Priority : Collection_Priority := 0;  
  This_Call_Did_A_Gc_Pass : out Boolean);
```

-- Causes a garbage collection pass on the specified volume. Caller will  
-- wait for at most MAX\_WAIT seconds for the garbage collection pass to  
-- begin. (A low space condition will cause garbage collection to happen  
-- automatically.) If this call starts a garbage collection pass, then  
-- This\_Call\_Did\_A\_Gc\_Pass will be returned true, and the caller will wait  
-- until the pass is completed (a potentially long time, since that  
-- involves lots of work).

-- The garbage collection pass will be executed at the better of the  
-- specified collection priority and that determined by the priority  
-- policy function, defined below (with the Set\_Priority\_Policy  
-- operation).

-- There is only 1 worker task. This serializes garbage collection to  
-- help prevent anomalous situations in which the garbage collector runs  
-- for extraordinarily long periods of time.

```
generic  
with procedure Put_Line (S : String);  
procedure Display_Current_Gc_State;
```

-- Displays various pieces of gc state.

```
function Garbage_Collector_Is_Running  
(Volume : Volume_Number) return Boolean;
```

-- Returns true iff the garbage collector is running on the specified  
-- volume.

```
function Garbage_Collector_Is_Running return Boolean;
```

-- Returns true iff the garbage collector is running.

```
function Jobs_Are_Stopped_By_Volume (Volume : Volume_Number) return Boolean;
```

-- Returns true iff all jobs were stopped because the specified Volume  
-- reached the warning threshold.

```
function Jobs_Are_Stopped return Boolean;
```

-- Returns true iff all jobs were stopped because some volume reached the  
-- warning threshold.

```
function Start_Time return Calendar.Time;
```

-- Returns 'first when gc is not running.

```
type Collection_Phase is (Idle, Waiting_For_Backup_To_Finish,  
  Taking_Snapshot, Deleting_Segments,  
  Traversing_Virtual_Memory, Reclaiming_Blocks);
```

```
function Current_Phase return Collection_Phase;
```

-- Returns Idle when gc is not running.

```
function Current_Priority (Volume : Volume_Number)  
  return Collection_Priority;
```

-- If the Volume does not exist, or the garbage collector for that volume  
-- has never run, returns worst priority (-1). If the garbage collector is  
-- currently running, returns the value that it is running at; otherwise  
-- returns the value it last ran at.

```
procedure Set_Current_Priority (Volume : Volume_Number;  
  Desired_Priority : Collection_Priority);
```

-- If the garbage collector is currently running in the  
-- Traversing\_Virtual\_Memory phase, this operation will change the rate  
-- at which its running, otherwise has no effect.

-- Note that there is a policy function that places a lower bound on the  
-- current collection priority. (See the Set\_Priority\_Policy operation,  
-- below.) If the specified priority is below that specified by the  
-- priority policy, the Set\_Current\_Priority operation is a noop.

```
function Reclaimed_Segments return Natural;
```

-- Returns the number of segments which were deleted by gc.

-- These values are set to 0 when gc starts. Don't expect them to  
-- become non-zero until Current\_Phase is >= Deleting\_Segments.

```
function Visited_Segments return Natural;
```

```
function Time_Spent_Delaying return Duration;
```

-- The first 2 functions return the number of segments/blocks which have  
-- been "visited" by the virtual memory traversal. The 3rd function  
-- returns an approximate value for how much time gc has spent backing off,  
-- as a result of the load parameters. These values are set to 0 when  
-- gc starts. Don't expect them to become non-zero until Current\_Phase is  
-- >= Traversing\_Virtual\_Memory.

```
function Stop_Run_Load return Natural; -- default: 250
```

```
procedure Set_Stop_Run_Load (Run_Load : Natural);
```

```
function Stop-Withheld_Load return Natural; -- default: 1
```

```
procedure Set_Stop-Withheld_Load (Withheld_Load : Natural);
```

```
function Start_Run_Load return Natural; -- default: 125
```

```
procedure Set_Start_Run_Load (Run_Load : Natural);
```

```
function Start-Withheld_Load return Natural; -- 0
```

```
procedure Set_Start-Withheld_Load (Withheld_Load : Natural);
```

```
procedure Use_Standard_Backoff_Algorithm;
```

-- During the Traversing\_Virtual\_Memory phase, the collector follows  
-- roughly the following algorithm:

```
-- loop
```

```
-- do about 500 milliseconds of work, and a few disk accesses  
-- if (Run_Load > Stop_Run_Load) or  
-- (Withheld_Load > Stop-Withheld_Load) then
```

```
-- loop
```

```

-- delay 30.0; -- seconds
-- exit when (Run_Load < Start_Run_Load) and
-- (Mithheld_Load < Start_Mithheld_Load);
--
-- end loop;
-- end if;
--
-- The loads for the stop tests are sampled over the last minute. The
-- load for the restart tests are sampled over the last 5 minutes.
--
-- The values supplied to the Set_ operations should be MTS loads,
-- multiplied by 100. For example, 125 means an MTS load of 1.25.
--
-- The Use_Standard_Backoff_Algorithm procedure will revert the collector
-- to using its built in backoff algorithm.
--
function Footprinting_Enabled return Boolean;
procedure Set_Footprinting (Desired_Value : Boolean);
--
-- When the system boots, footprinting is disabled. True means footprinting
-- is enabled. When footprinting is enabled, reclaimed disk blocks are
-- cleared. This lengthens the amount of time spent in the
-- Reclaiming_Blocks phase, since 1 disk I/O is required for every
-- block reclaimed.
--
function Backup_Killing_Enabled return Boolean;
procedure Set_Backup_Killing (Desired_Value : Boolean);
--
-- When the system boots, backup kill mode is enabled. True means to
-- enable backup kill mode. In backup kill mode, if the garbage collector
-- is caused to run (via any method, including direct call, daemon
-- scheduling, or disk space thresholds) and a backup is in progress,
-- the backup will be terminated. Recall that this whole issue stems
-- from the fact that the garbage collection mechanisms don't work in
-- the case where backup has a retained snapshot which is earlier than
-- the retained snapshot which the garbage collector is using.
--
function Prevent_Stop_By_Warning (Job : Task_Vpids) return Boolean;
procedure Set_Prevent_Stop_By_Warning (Desired_Value : Boolean);
--
-- This mechanism allows individual jobs to prevent themselves from being
-- stopped when the garbage collector reaches the warning threshold (and
-- stops all jobs). By default, jobs 4 & 5 are registered as not being
-- stopped; otherwise the compaction daemons (such as DDB) and the gc can
-- get deadlocked. This mechanism is also used by backup to prevent a
-- deadlock.
--
function Get_Priority_Policy
  (Raised_Count : Integer; Started_Count : Integer)
  return Collection_Priority;
procedure Set_Priority_Policy (Raised_Count : Integer;
  Started_Count : Integer;
  Desired_Priority : Collection_Priority);
--
-- There is a policy function which specifies the minimum allowable
-- collection priority as a function of the number of volumes which have
-- reached the Start_Collection threshold and the number of volumes which
-- have reached the Raise_Priority threshold. The default policy is:
--
-- If there are 0 volumes past the Raise_Priority threshold, then the

```

```

-- priority policy is the following function of the number of volumes
-- past the Start_Collection threshold:
--
-- 1 volume => Collection_Priority'(-1)
-- 2 volumes => 0
-- 3 volumes => 1
--
-- For all remaining combinations, the policy is Collection_Priority'(2).
--
-- The default policy is intended to have the following properties:
-- When just a single volume wants to collect, let it run using
-- spare cycles. When more than one volume wants to collect, stop
-- backing off. When any Raise_Priority threshold is reached,
-- raise priority such that the collection will preempt most background
-- jobs.
--
-- The set operation allows you to define your own priority policy. Hints
-- for "rolling your own": (1) Setting the policy to return -1 for all
-- inputs effectively disables the policy altogether. (2) The default
-- policy does not increase the priority to 3 on the assumption that
-- either (a) the mts parameters in effect limit the length of time an
-- attached job can remain in the foreground or (b) that people do not
-- typically leave attached jobs run for long periods of time. (3)
-- Creating a policy which increases the priority to 5 or greater will
-- probably have roughly the same impact as reaching the Stop_Jobs
-- threshold.
--
end Disk_Daemon;

```



```

package Hash is
-- simple hash functions to integer and long_integer
-- all functions are guaranteed not to raise an exception

generic
  type T is limited private;
  type Ptr is access T;
  function Pointer_To_Integer (P : Ptr) return Integer;

generic
  type T is limited private;
  type Ptr is access T;
  function Pointer_To_Long_Integer (P : Ptr) return Long_Integer;

generic
  type T is limited private;
  type Ptr is access T;
  pragma Segmented_Heap (Ptr);
  function Heap_Pointer_To_Integer (P : Ptr) return Integer;

generic
  type T is limited private;
  type Ptr is access T;
  pragma Segmented_Heap (Ptr);
  function Heap_Pointer_To_Long_Integer (P : Ptr) return Long_Integer;

function Long_Integer_To_Integer (Value : Long_Integer) return Integer;

pragma Subsystem (Abstract_Types);
pragma Module_Name (4, 731);

end Hash;

```

```

with Directory;

package Library_Object_Editor is
-- Cause the specified library to be displayed. If selection is not
-- nil, then that object will be selected as the library is displayed.
-- If Force_Redraw is true, the image will be redrawn even if it
-- already exists.

procedure Display (Library : Directory.Object;
                  Selection : Directory.Object := Directory.Nil;
                  Force_Redraw : Boolean := False);

procedure Update_Changed_Images;

procedure Create_World (Name : String := ">>WORLD NAME<<<";
                       Volume : Natural := 0);

procedure Create_Directory (Name : String := ">>DIRECTORY NAME<<<");
procedure Create_Unit (Name : String := ">>ADA NAME<<<");

pragma Subsystem (Object_Editor);
pragma Module_Name (4, 2221);

end Library_Object_Editor;

```

```

with Diana;
with Links_Implementation;
with Io_Exceptions;

package Link_Tools is
pragma Subsystem (Tools, Private_Part => Closed);
pragma Module_Name (4, 3975);

package Links renames Links_Implementation;

subtype World_Name is String;
This_World : constant World_Name := "0";

-- The string name for the World associated with the current naming
-- context.

subtype Link_Name is String;
-- An Ada simple name. When used as an in-parameter, except in Add and
-- Replace, it may contain wildcard characters. In Add and Replace it
-- may contain substitution characters.

subtype Source_Name is String;
-- A directory string name that specifies an existing Ada Library Unit.
-- (The unit does not have to be installed, but its declaration must be
-- in a library.) May contain wildcard characters when used as an
-- in-parameter.

subtype Link_Kind is Links.Link_Kind;
-- If the source unit is in the same world as the link.
-- the link is Internal; otherwise it is External.

Any : constant Link_Kind := Links.Any;
External : constant Link_Kind := Links.External;
Internal : constant Link_Kind := Links.Internal;

function "=" (L, R : Link_Kind) return Boolean renames Links."=";

Name_Error : exception renames Io_Exceptions.Name_Error;
-- Raised if a World parameter cannot be resolved.

Use_Error : exception renames Io_Exceptions.Use_Error;
-- Raised of the link pack cannot be opened for other reasons.

function Has (Link : Link_Name := "0";
Source : Source_Name := "0";
Kind : Link_Kind := Link_Tools.Any;
World : World_Name := This_World) return Boolean;

-- Returns true iff the pack contains at least one link that matches the
-- given link, source, and kind parameters.

function Link (Source : Source_Name;
Kind : Link_Kind := Link_Tools.Any;
World : World_Name := This_World) return Link_Name;

function Link (Source : Directory_Object;
Kind : Link_Kind := Link_Tools.Any;
World : World_Name := This_World) return Link_Name;

-- Given a Source_Name, Def_ID, or Directory_Object for an Ada unit,
-- returns a link for that unit in the given world that matches the
-- Kind parameter. A null string is returned if no such link can be
-- found.

function Source (Link : Link_Name;
Kind : Link_Kind := Link_Tools.Any;
World : World_Name := This_World) return Source_Name;

function Source (Link : Link_Name;
Kind : Link_Kind := Link_Tools.Any;
World : World_Name := This_World) return Diana.Tree;

function Source (Link : Link_Name;
Kind : Link_Kind := Link_Tools.Any;
World : World_Name := This_World) return Directory_Object;

-- Returns either the Source_Name, Def_ID, or Directory_Object for an Ada
-- unit corresponding to a link that matches the Link and Kind parameter.
-- Returns a null object if no match can be found.

type Dependent_Iterator is private;

function Dependents (Link : Link_Name := "0";
Source : Source_Name := "0";
Kind : Link_Kind := Link_Tools.Any;
World : World_Name := This_World)
return Dependent_Iterator;

procedure Next (Iter : in out Dependent_Iterator);
function Value (Iter : Dependent_Iterator) return Diana.Tree;
function Done (Iter : Dependent_Iterator) return Boolean;

-- Computes the Library Units of the world that are installed or coded
-- and reference any of the links specified by the Source, Link and Kind
-- parameters.

type Link_Iterator is private;

procedure Init (Iter : out Link_Iterator;
Source : Source_Name := "0";
Link : Link_Name := "0");

```

```

with Diana;
with Links_Implementation;
with Io_Exceptions;

package Link_Tools is
pragma Subsystem (Tools, Private_Part => Closed);
pragma Module_Name (4, 3975);

package Links renames Links_Implementation;

subtype World_Name is String;
This_World : constant World_Name := "0";

-- The string name for the World associated with the current naming
-- context.

subtype Link_Name is String;
-- An Ada simple name. When used as an in-parameter, except in Add and
-- Replace, it may contain wildcard characters. In Add and Replace it
-- may contain substitution characters.

subtype Source_Name is String;
-- A directory string name that specifies an existing Ada Library Unit.
-- (The unit does not have to be installed, but its declaration must be
-- in a library.) May contain wildcard characters when used as an
-- in-parameter.

subtype Link_Kind is Links.Link_Kind;
-- If the source unit is in the same world as the link.
-- the link is Internal; otherwise it is External.

Any : constant Link_Kind := Links.Any;
External : constant Link_Kind := Links.External;
Internal : constant Link_Kind := Links.Internal;

function "=" (L, R : Link_Kind) return Boolean renames Links."=";

Name_Error : exception renames Io_Exceptions.Name_Error;
-- Raised if a World parameter cannot be resolved.

Use_Error : exception renames Io_Exceptions.Use_Error;
-- Raised of the link pack cannot be opened for other reasons.

function Has (Link : Link_Name := "0";
Source : Source_Name := "0";
Kind : Link_Kind := Link_Tools.Any;
World : World_Name := This_World) return Boolean;

-- Returns true iff the pack contains at least one link that matches the
-- given link, source, and kind parameters.

function Link (Source : Source_Name;
Kind : Link_Kind := Link_Tools.Any;
World : World_Name := This_World) return Link_Name;

function Link (Source : Directory_Object;
Kind : Link_Kind := Link_Tools.Any;
World : World_Name := This_World) return Link_Name;

-- Given a Source_Name, Def_ID, or Directory_Object for an Ada unit,
-- returns a link for that unit in the given world that matches the
-- Kind parameter. A null string is returned if no such link can be
-- found.

function Source (Link : Link_Name;
Kind : Link_Kind := Link_Tools.Any;
World : World_Name := This_World) return Source_Name;

function Source (Link : Link_Name;
Kind : Link_Kind := Link_Tools.Any;
World : World_Name := This_World) return Diana.Tree;

function Source (Link : Link_Name;
Kind : Link_Kind := Link_Tools.Any;
World : World_Name := This_World) return Directory_Object;

-- Returns either the Source_Name, Def_ID, or Directory_Object for an Ada
-- unit corresponding to a link that matches the Link and Kind parameter.
-- Returns a null object if no match can be found.

type Dependent_Iterator is private;

function Dependents (Link : Link_Name := "0";
Source : Source_Name := "0";
Kind : Link_Kind := Link_Tools.Any;
World : World_Name := This_World)
return Dependent_Iterator;

procedure Next (Iter : in out Dependent_Iterator);
function Value (Iter : Dependent_Iterator) return Diana.Tree;
function Done (Iter : Dependent_Iterator) return Boolean;

-- Computes the Library Units of the world that are installed or coded
-- and reference any of the links specified by the Source, Link and Kind
-- parameters.

type Link_Iterator is private;

procedure Init (Iter : out Link_Iterator;
Source : Source_Name := "0";
Link : Link_Name := "0");

```

```

Kind : Link_Kind := Link_Tools.Any;
World : World_Name := This_World;

procedure Next (Iter : in out Link_Iterator);
function Link (Iter : Link_Iterator) return Link_Name;
function Source (Iter : Link_Iterator) return Source_Name;
function Source (Iter : Link_Iterator) return Diana_Tree;
function Kind (Iter : Link_Iterator) return Directory_Object;
function Done (Iter : Link_Iterator) return Link_Kind;
function Done (Iter : Link_Iterator) return Boolean;

-- For iterating over the Links in a link pack that match given Source
-- and Link patterns

end Link_Tools;

```

```

generic
type Element is private;
-- must be a pure value
-- ie. no initialization or finalization is necessary
-- = and := are equality and copy
pragma Must_Be_Constrained (Yes => Element);
package List_Generic is
pragma Subsystem (Abstract_Types, Private_Part => Open);
pragma Module_Name (4, 710);
type List is private;
-- may generate garbage
-- = and := operate on references
-- "make" constructs lists with structural sharing
-- constraint error is raised when nil i provided to any of
-- first, rest, set_first, or set_rest
function Make (X : Element; L : List) return List;
function Nil return List;
function Is_Empty (L : List) return Boolean;
procedure Free (L : in out List);
-- make L empty
function First (L : List) return Element;
function Rest (L : List) return List;
procedure Set_Rest (L : List; To_Be : List);
procedure Set_First (L : List; To_Be : Element);
function Length (L : List) return Natural;
type Iterator is private;
procedure Init (Iter : out Iterator; L : List);
procedure Next (Iter : in out Iterator);
function Value (Iter : Iterator) return Element;
function Done (Iter : Iterator) return Boolean;
private
type Listdata;
type List is access Listdata;
-- variables of type list are initialized to null
type Listdata is
record
First : Element;
Rest : List;
end record;
type Iterator is new List;
end List_Generic;

```

```

generic
  Size : Integer;
  -- number of buckets

  type Domain_Type is private;
  type Range_Type is private;
  -- both types are pure values
  -- no initialization or finalization of values of either
  -- domain_type or range_type is necessary
  -- = and := can be used for equality and copy

  with function Hash (Key : Domain_Type) return Integer is <>;
  -- efficiency => spread hash over an interval at least as great as size

  pragma Must_Be_Constrained (Yes => Domain_Type, Range_Type);

package Map_Generic is

  type Map is private;

  type Pair is
    record
      D : Domain_Type;
      R : Range_Type;
    end record;

  function Eval (The_Map : Map; D : Domain_Type) return Range_Type;
  procedure Find (The_Map : Map;
    D : Domain_Type;
    R : in out Range_Type;
    Success : out Boolean);

  procedure Find (The_Map : Map;
    D : Domain_Type;
    P : in out Pair;
    Success : out Boolean);

  procedure Define (The_Map : in out Map;
    D : Domain_Type;
    R : Range_Type;
    Trap_Multiples : Boolean := False);

  procedure Undefined (The_Map : in out Map; D : Domain_Type);

  procedure Initialize (The_Map : out Map);
  function Is_Empty (The_Map : Map) return Boolean;
  procedure Make_Empty (The_Map : in out Map);

  procedure Copy (Target : in out Map; Source : Map);

  type Iterator is private;

  procedure Init (Iter : out Iterator; The_Map : Map);
  procedure Next (Iter : in out Iterator);
  function Value (Iter : Iterator) return Domain_Type;
  function Done (Iter : Iterator) return Boolean;

  Undefined : exception;

Map_Generic, !Tools

```

```

-- raised by eval if the domain value is not in the map
Multiply_Defined : exception;
-- raised by define if the domain value is already defined and
-- the trap_multiples flag has been specified (i.e. is true)
function Cardinality (The_Map : Map) return Natural;
function Nil return Map;
function Is_Nil (The_Map : Map) return Boolean;
-----
-- Implementation Notes and Non-Standard Operations --
-- := and = operate on references
-- := implies sharing (introduces an alias)
-- = means is the same set, not the same value of type set
-- Initializing a map also makes it empty
-- Accessing an uninitialized map will raise CONSTRAINT_ERROR.
-- garbage may be generated
-- Concurrent Properties
-- any number of find/eval/is_empty may be safely done while one
-- define/undefine is taking place. If the define is redefining an
-- existing element in the domain of the map, concurrent reading is
-- safe if and only if := on range_type is atomic.

private
  type Node;
  type Set is access Node;

  type Node is
    record
      Value : Pair;
      Link : Set;
    end record;

  subtype Index is Integer range 0 .. Size - 1;

  type Table is array (Index) of Set;

  type Map_Data is
    record
      Bucket : Table;
      Size : Integer := 0;
    end record;

  type Map is access Map_Data;

  type Iterator is
    record
      The_Map : Map;
      Index_Value : Index;
      Set_Iter : Set;
      Done : Boolean;
    end record;

  end Map_Generic;

```

```

with Default;
with Directory;
with Machine;

package Object_Editor is
    pragma Subsystem (Object_Editor, Closed);
    pragma Module_Name (4, 2223);
    -- Detached jobs and jobs initiated via the program package have
    -- no associated image.
    type Focus is (Selection, Cursor, Image, Region);
    -- If not named specifically, an object (or collection of objects) can
    -- be designated by a highlighted selection on an image or by the
    -- location of the cursor within an image. Different algorithms
    -- for determining such designated objects are implemented by this
    -- package to satisfy the differing requirements of environment
    -- commands (taking one argument denoted in this way)
    -- Selection
    -- An object is selected in this mode only if its declaration is
    -- highlighted and the cursor is within or immediately adjacent to
    -- that highlighted region.
    -- This is the most restrictive selection mode and would be used
    -- by commands, such as delete and demote, that must have a
    -- strong indication from the user of the intended object.
    -- Cursor
    -- An object is selected in this mode if its declaration is
    -- selected as described in the preceding case or, if there is no
    -- highlighted selection on the image of the cursor, the cursor
    -- is on a portion of the object's declaration.
    -- This is less restrictive than the preceding case in that it
    -- accepts any placement of the cursor within an image. Commands
    -- such as definition or help would use this mode because it
    -- reduces the number of key strokes required by the user.
    -- Image
    -- An object is selected in this mode if its declaration is
    -- selected as described in the Selection mode or, if there is no
    -- highlighted selection on the image of the cursor, the cursor is
    -- on the image associated with the object.
    -- This mode accepts the same configurations of cursor and
    -- selection as in the Cursor mode, but treats the cursor less
    -- specifically by selecting the object that is represented by the
    -- entire image rather than the object that is represented by the
    -- portion of the image that the cursor is on.

```

```

-- Region
-- An object is selected in this mode if its declaration is
-- highlighted. The cursor does NOT have to be in the region.
-- This is less restrictive than Selection focus and would be
-- used by commands such as copy to obtain their source object.
-- Some object editors will behave the same in the Cursor and
-- Image modes because they do not support the notion of designations
-- nested within an image. For such editors, the Selection mode will
-- always fail unless the selection constitutes the entire image.
procedure Get_Declaration
    (Decl : out Directory.Declaration;
     Status : out Directory.Naming.Name_Status;
     Precision : Object_Editor.Focus := Object_Editor.Selection;
     Job : Default.Process_Id := Default.Process);
-- Returns the declaration for the object designated by the current
-- selection according to the specified algorithm.
procedure Get_Object (Object : out Directory.Object;
                     Status : out Directory.Naming.Name_Status;
                     Class : Directory.Class := Directory.Nil;
                     Precision : Object_Editor.Focus :=
                         Object_Editor.Selection;
                     Job : Default.Process_Id := Default.Process);
-- Returns the Directory object designated by the current selection
-- according to the specified algorithm.
function Get_Text
    (Precision : Object_Editor.Focus := Object_Editor.Region;
     Job : Default.Process_Id := Default.Process) return String;
-- Returns the text contained in the current focus. Region selections
-- are allowed. Text may include Ascii.LF's to indicate end of line.
function Get_Name
    (Precision : Object_Editor.Focus := Object_Editor.Image;
     Job : Default.Process_Id := Default.Process) return String;
-- Return the name of the image, which may not be a valid directory name,
-- or the name of the declaration/object associated with the
-- current cursor/selection/region/window.
-- IMAGE MANIPULATION OPERATIONS
procedure Release_Access (The_Object : Directory.Object);
-- acquire rights to the designated object if they are held by the current
-- session; allows tools to guard against being locked out by other windows
procedure Display_Declaration (For_Object : Directory.Object;
                               In_Library : Boolean := False;
                               In_Place : Boolean := False);
procedure Display_Declaration (For_Version : Directory.Version;
                               In_Library : Boolean := False;
                               In_Place : Boolean := False);

```

```

-- procedure Display (The_Object : Directory.Object;
--                   In_Place : Boolean := False);
-- procedure Display (The_Version : Directory.Version;
--                   In_Place : Boolean := False);
--
-- In_Library will cause ada subunits to have their enclosing
-- library displayed instead of their stub declaration in their
-- parent ada unit.
-- pragma Consume_Offset (4);
--
-- procedure Update_Changed_Image (The_Object : Directory.Object);
-- -- notify the editor that changes have been made to the object that it
-- -- doesn't know about yet
--
-- procedure Update_Changed_Images;
-- -- non-specific (and slower) version of the above
--
-- subtype Editor_Name is String;
--
-- function Name return Editor_Name;
-- -- Returns the name of the object editor associated with the current image.
--
-- function Supports_Declarations
--   (Editor : Editor_Name := Name) return Boolean;
-- function Supports_Subobjects (Class : Directory.Class := Directory.Nil;
--   Editor : Editor_Name := Name) return Boolean;
--
-- -- Indicates whether the named editor can ever associate
-- -- objects/declarations with the cursor or selection (other than the
-- -- object/declaration associated with the containing image).
--
-- -- Iterate over all object editors. Primarily for completeness, but
-- -- allows tool writers to determine the set of applicable OE's.
--
-- type Iterator is private;
-- procedure Init (Iter : out Iterator);
-- function Done (Iter : Iterator) return Boolean;
-- function Value (Iter : Iterator) return Editor_Name;
-- procedure Next (Iter : in out Iterator);
--
-- -- Returns the job that the editor regards as current
-- function Current_Job return Machine.Job_Id;
--
-- procedure Display_Declaration (For_Object : Directory.Object;
--                               In_Library : Boolean := False;
--                               In_Place : Boolean := False);
-- procedure Display_Declaration (For_Version : Directory.Version;
--                               In_Library : Boolean := False;
--                               In_Place : Boolean := False);
--
-- procedure Display (The_Object : Directory.Object;
--                   In_Place : Boolean := False);
-- procedure Display (The_Version : Directory.Version;
--                   In_Place : Boolean := False);
--
-- end Object_Editor;

```

```

with Directory;

-- All options appear in Adaesque aggregate notation, with appropriate
-- relaxations of the rules. Switch values and switch names, where the set
-- of choices is static (i.e. fixed set of switches, but not for User
-- names), should recognize unique prefix. A package is available for
-- parsing parameter lists that adhere to this convention.

-- Options are formed from the following lexical components:

-- Punctuation ::= '>' | '=' | ':' | '|' | '..'
--              ::= Separator
--
-- Separator    ::= ',' | ';'
--
-- Value        ::= Directory-String-Name
--              ::= Integer-Literal
--              ::= Float-Literal
--              ::= Literal
--              ::= '<'
--
--              -- The default value defined for an
--              -- option.
--              -- Any sequence of contiguous char-
--              -- acters not including separators;
--              -- leading and trailing blanks are
--              -- removed.
--              -- Any sequence of contiguous char-
--              -- acters, balanced with respect to
--              -- parentheses, nested within
--              -- parentheses. The outer-most
--              -- enclosing parentheses are not part
--              -- of the value, but all contained
--              -- characters, including blanks, are.
--
-- Name         ::= Simple-Ada-Name
--              ::= Other-Name
--
--              -- Any sequence of contiguous char-
--              -- acters not including punctuation
--              -- or blanks
--
-- Literal      ::= Simple-Ada-Name
--
-- Any two-character sequence beginning with \ is interpreted as a single,
-- non-special occurrence of the second character. Thus, "\." is NOT a
-- separator character, but a benign occurrence of '.'.
--
-- Blanks are allowed around the special characters:
--
-- Since Blanks are not allowed in a Name, a blank following a Name may be
-- used as a separator.
--
-- The syntax is (enclosing quotes not included):
--
-- Options    ::= [Option {Separator Option}]
-- Option     ::= Range {'|' Range} [('>' | ':' | '=' | '..') Value]
--            ::= ['-'] Range {'|' Range}
--            ::= Literal
--            ::= '-' File-Name
-- Range      ::= Name ['..'] Name

```

```

-- ::= 'others' -- Denotes names not otherwise
-- -- specified.
--
-- General semantics:
--
-- A Name denotes an option. A Range denotes the options in a predefined
-- sequence of options from the option denoted by the first name to and
-- including the option denoted by the last name of the range. The
-- specified Value is associated with each option denoted by the Names and
-- Ranges of the Option.
--
-- If a Value clause is omitted, the options denoted by the Names and
-- Ranges of the Option must be Boolean-valued. If ':' precedes
-- the Names of an Option, the options assume the value False, otherwise they
-- assume the value True.
--
-- A Literal denotes a value of a specific option. When it appears as an
-- Option, it denotes both the Name and the Value of that option.
--
-- If a name appears more than once, the value associated with the leftmost
-- instance of the name is the one used.
--
-- Examples:
--
-- Access_List.Set ("Public=>RW");
-- Profile.Set ("+++,---,+++,lll,l=>80");
-- Profile.Set ("Persevere,+++ | --- | +++ | lll => true,v => 80");
-- Source_Archive.Restore (... Form => "New_Units, Acl => (Public=>RW)");
-- Switches.set ("Semantics.Ignore_Minor_Errors := True");
--
generic
type Option_Id is (<);
--
-- This discrete type defines the ordering of option names used to
-- define ranges. Its values are used in the programmatic interface to
-- identify options. For a static collection of options, such as in
-- Profile, Option_Id would probably be an enumerated type; its
-- enumeration ids would define the names of the options. For
-- non-static options, such as access lists, Option_Id would be an
-- integer type and most names would be defined using the define
-- procedure exported by the generic.
Nil : in Option_Id := Option_Id'First;
First : in Option_Id := Option_Id'Succ (Nil);
Last : in Option_Id := Option_Id'Last;
--
-- Nil is an Option_Id that represents no option name. Only option_id's
-- in the range First .. Last are definable; Nil should not be in that
-- range;
Option_Kinds : in String := "others => Unspecified";
--
-- Specification of the kind of each option. The string must satisfy
-- the syntax for a forms parameter in which Names are taken from the
-- set of Option_Id images and Values are the enumeration ids of the
-- type Option_Kind, defined below. (The Option_Kind 'Literal' may not
-- be specified in this string; use the Define function.) The default
-- kind for all options is 'Unspecified'. Options that are not

```

```

-- Boolean-Valued must be followed by a Value clause when they are used
-- in a parameter string. If the Kind of an option is other than
-- 'Unspecified', the parser will verify that the associated value is of
-- the proper form for the specified Kind.
Default_Values : in String := "";
--
-- Default values for the options. The string must satisfy the syntax
-- for a forms parameter in which Names are taken from the set of
-- Option_Id images. Not all Option_Ids need to have Default_Values.
-- Default values are substituted for the special symbol '<' when it
-- appears in a Value clause. If no default value has been specified
-- for an option and the option appears with a '<' value, the reference
-- to the option is deleted by the option parser.
Alternate_Names : in String := "";
--
-- Alternate names for the options. The string must satisfy the syntax
-- for a forms parameter in which Names are taken from the set of
-- Option_Id images, and the Values obey the syntax for Other_Names.
-- Not all Option_Ids need to have Alternate_Names. The standard name
-- for an option is the image of the Option_Id. The Undefine function
-- may be used to remove the standard name from the set of permitted
-- names. All names for the same option_id value have the same kind and
-- default value.
From : Option_Id := First;
To : Option_Id := Last;
--
-- From and To define the range of Option_Id's that make up the
-- initial set of defined options. This set can be expanded or
-- reduced using the Define and Undefine procedures defined in the
-- package.
package Parameter_Parser is
pragma Subsystem (Directory);
pragma Module_Name (4, 3528);
type Option_Kind is (Unspecified, Directory_String_Name, Boolean_Valued,
Integer_Valued, Float_Valued, Literal);
procedure Define (Option : Option_Id;
Name : String := "",
Kind : Option_Kind := Unspecified;
Default_Value : String := "";
Allow_Name_Prefix : Boolean := False);
--
-- Defines a new Name to be associated with the given Option_Id. The
-- default Name is the Option_Id' image of the Option. Any number of
-- names may be associated with an Option_Id value. The parameter
-- specification may use any of these names to set the option.
--
-- Allow_Name_Prefix allows a unique prefix of the Name to be used in
-- place of the Name in a parameter specification.
--
-- The Default_Value string is parsed as if it were a Value
-- specification; a balanced string or '\' must be used to protect
-- separators in the default value. If Default_Value is the null

```

```

-- string, no default value is assigned to the option. If you want the
-- default value to be a null string, use "".
-- If Kind is 'Literal', Name must be non-null. The given Name must
-- never appear with a Value clause. When it appears by itself, without
-- a Value clause, the implied value is the Name itself. (The generic
-- package Enumerated_Value, defined below, provides a convenient way to
-- define all enumeration ids of an option as literals.)

procedure Undefine (Name : String);

-- Removes the Name (and its prefixes) as a possible option name.

procedure Undefine (From : Option_Id; To : Option_Id := Nil);

-- Removes all names and prefixes that denote an Option_Id in the given
-- range as possible option names.

procedure Allow_Name_Prefix (Name : String; Value : Boolean := True);

procedure Allow_Name_Prefix (Value : Boolean := True;
                             From : Option_Id := Nil;
                             To : Option_Id := Nil);

-- The Allow_Name_Prefix flag for a name, when set, allows a unique
-- prefix of the name to be used in place of the name in a parameter
-- specification. The default setting of this flag for the initial set
-- of Option_Ids is true.

-- The first procedure sets the Allow_Name_Prefix flag for the named
-- option.

-- The second procedure sets the Allow_Name_Prefix flag for all defined
-- Names that map to an Option_Id in the specified range. The range
-- implied by the default values is the full set of Option_Ids defined
-- at the time of call. If From is non-Nil and To is Nil, the single
-- Option_Id From is implied. If From and To are both non-Nil, all
-- Option_Ids in the range From .. To are implied.

type Iterator is private;

procedure Parse (Parameter : String;
                Options : out Iterator;
                Success : out Boolean);

function Parse (Parameter : String) return Iterator;

function Is_Successful (Iter : Iterator) return Boolean;

-- Success is True, iff all options were parsed correctly. When no
-- options parsed correctly, a Done iterator is returned, which may be
-- passed to the Diagnosis function to obtain more information. If
-- some, but not all, options were parsed correctly the returned
-- iterator will be non-null. Iterations (positions in the iterator)
-- are allocated for erroneous specifications as well as for correctly
-- parsed specifications. The Is_Ok() predicate distinguishes between
-- them.

-- The iterator returned by Parse represents an expanded, unfactored
-- specification, equivalent to the input specification; each iteration

```

```

-- represents a simple specification of the form, "Name [=> Value]". All
-- Names are returned with their full spelling. Ranges and the reserved
-- name 'others' are expanded so that there is one iteration for each
-- option_id covered by the Range or 'others.' Duplicate specifications
-- have been removed, leaving the last specification at its point of
-- occurrence. Except for the deleted duplications all specifications
-- of the input string are present in the iterator in the same order as
-- in the input string.

-- In the following subprograms, the optional Name parameter is used
-- to interrogate the iterator as a set. The default value, Nil,
-- addresses the current iteration of the iterator.

function Is_Ok (Iter : Iterator; Name : Option_Id := Nil) return Boolean;

-- Indicates whether the designated option was syntactically correct in
-- the specification; If the named option was not specified, Is_Ok()
-- returns False;

function Is_Present (Iter : Iterator; Name : Option_Id) return Boolean;

-- Indicates whether the indicated option was present in the option
-- parameter string. An option is present if its name was
-- parsable. It may otherwise be in error.

function Diagnosis (Iter : Iterator; Name : Option_Id := Nil) return String;

-- Returns text for a message that describes what was wrong with the
-- option specification, if anything. If the named option was not
-- specified, Diagnosis returns a message to this effect. If an
-- option Is_Ok(), Diagnosis returns the null string.

function Done (Iter : Iterator) return Boolean;
procedure Next (Iter : in out Iterator);
procedure Reset (Iter : in out Iterator);

-- Advances the iterator to the next iteration.

function Name (Iter : Iterator) return Option_Id;

-- Returns Nil if the iteration corresponds to an unparseable
-- specification.

function Name (Iter : Iterator; Name : Option_Id := Nil) return String;

-- Returns the name that was used in the specification to denote the
-- indicated Option_Id. The full name is returned even if a prefix was
-- used.

function Has_Value
  (Iter : Iterator; Name : Option_Id := Nil) return Boolean;

-- Indicates whether the Value clause for the indicated option was
-- specified or not.

function Get_Image (Iter : Iterator;
                  Name : Option_Id := Nil;
                  Default : String := "") return String;

```



```
-- Get_Image returns the uninterpreted image of the Value associated
-- with the indicated option. If Is_Ok() or Has_Value() is false,
-- the null string is returned. If Is_Present() is false, the
-- default value associated with the named option is returned.
```

```
-- In the returned value, the two-character sequences beginning with \
-- have been reduced to a single character.
```

```
-- The Get_xxx functions defined below use Get_Image to obtain a
-- string to interpret. Thus they operate on the default value
-- when the option has not been named.
```

```
function Kind (Iter : Iterator; Name : Option_Id := Nil) return Option_Kind;
```

```
-- The value of Get_Image() on the specified option is inspected to
-- determine its type. Kind returns Unspecified if the kind cannot be
-- determined.
```

```
function Get_Object (Iter : Iterator;
  Name : Option_Id := Nil;
  Default : Directory_Object := Directory.Nil)
  return Directory_Object;
```

```
-- The value of Get_Image() is evaluated by Directory.Naming.
-- Resolve. Directory.Nil is returned if it cannot return a
-- Directory_Object value. The results of the attempt to resolve
-- the directory name will also be reflected in Is_Ok() and
-- Diagnosis() after the call to Get_Object.
```

```
function Get_Objects (Iter : Iterator;
  Name : Option_Id := Nil;
  Deleted_Ok : Boolean := False;
  Objects_Only : Boolean := False)
  return Directory.Naming.Iterator;
```

```
-- The value of Get_Image() is evaluated by Directory.Naming.
-- Resolve. A Done iterator is returned if it cannot be resolved.
-- The results of the attempt to resolve the directory name will
-- also be reflected in Is_Ok() and Diagnosis() after the call to
-- Get_Objects.
```

```
function Get_Boolean (Iter : Iterator;
  Name : Option_Id := Nil;
  Default : Boolean := False) return Boolean;
```

```
-- If Get_Image() is non-null, Get_Boolean tries to interpret it as
-- a Boolean_Literal and returns the denoted value. If the it is not
-- a Boolean_Literal, False is returned, and Is_Ok() and Diagnosis()
-- will indicate an error and the nature of the error. A
-- Boolean_Literal may be any prefix of True or False.
```

```
-- If Get_Image() is null, Get_Boolean returns false if the name
-- appeared with a '-' and it returns true otherwise.
```

```
function Get_Integer (Iter : Iterator;
  Name : Option_Id := Nil;
  Default : Integer := Integer'Last) return Integer;
```

```
-- Get_Integer tries to parse the Get_Image() value as an
-- integer and returns the denoted value. If Integer'Value fails,
```

```
-- Integer'last is returned and Is_Ok() and Diagnosis() will
-- identify the error.
```

```
function Get_Float (Iter : Iterator;
  Name : Option_Id := Nil;
  Default : Float := Float'Safe_Large) return Float;
```

```
-- Get_Float tries to parse the Get_Image() value as a float
-- literal and returns the denoted value. If it cannot parse the
-- value as a float value, Float'large is returned and Is_Ok() and
-- Diagnosis() will identify the error.
```

```
generic
  type T is (<>);
  Nil : in T := T'First;
  Id : Option_Id := Parameter_Parser.Nil;
  Allow_Name_Prefix : Boolean := True;
  package Enumerated_Value is
  function Get_Enumeration (Iter : Iterator;
    Name : Option_Id := Parameter_Parser.Nil;
    Allow_Value_Prefix : Boolean := True;
    Default : T := Nil) return T;
  end Enumerated_Value;
```

```
-- Get_Enumeration tries to interpret the Get_Image() value as the
-- unique prefix of an image of a component of T. It returns Nil and
-- prepends Is_Ok() and Diagnosis() if no such interpretation is possible.
-- If Allow_Value_Prefix is false, only full spellings of values of type
-- T are recognized.
```

```
-- If the Id formal parameter is not Nil, the values of type T will be
-- defined as legal option names. If one of these values is found in an
-- option list in the place of a name, it will be treated as a value of
-- the option denoted by Id (i.e., "Id =>" is implicitly inserted before
-- the value). If Allow_Name_Prefix is True, unique prefixes of the
-- values of T will be recognized.
```

```
generic
  type T is private;
  Nil : in T;
  with function Value (S : String) return T is <>;
  with function Diagnosis (S : String) return String;
  function Get_Value (Iter : Iterator;
    Name : Option_Id := Parameter_Parser.Nil;
    Default : T := Nil) return T;
```

```
-- Get_Value applies the formal Value function to the Get_Image()
-- value. Value should return Nil if the passed value is
-- unacceptable. If Nil is returned, the next call to Diagnosis
-- will return the value returned by Get_Value.Diagnosis.
```

```
private
  type Iterator_Data;
  type Iterator is access Iterator_Data;
  pragma Segmented_Heap (Iterator);
  end Parameter_Parser;
```

```

with Directory;
with Simple_Status;
package Profile is
  pragma Subsystem (Directory);
  pragma Module_Name (4, 3219);
  -- A collection of job-related utilities for control of log generation,
  -- Activity files, and error reactions. These facilities are used by
  -- Commands.Compilation, Directory, and Tape, Source_Archive, and may
  -- be used by user-written procedures.
  type Response_Profile is private;
  -- The aggregate of all components of the job response profile
  function Get return Response_Profile;
  -- Profile for the current job
  function Get_Default return Response_Profile;
  -- Profile for the Session (which is the default for a job that
  -- does not specify one)
  procedure Set (Profile : Response_Profile);
  -- Set profile for rest of current job
  procedure Set_Default (Profile : Response_Profile);
  -- Set profile for session; this is the value used for the job
  -- response profile if none is otherwise specified.
  function Default_Profile return Response_Profile;
  -- If the user has established no response profile for his session,
  -- this profile is used; it is the aggregate of all the Default_xxx
  -- constants defined in this package.
  subtype Name is String; -- an unambiguous string name
  -- A map is maintained between a job id and the following values:
  -- The ERROR_REACTION specifies which of several options a command is
  -- to follow in reacting to error situations.
  -- The LOG_FILTER specifies in some detail, the desired content of the
  -- log that is generated by the command.
  -- The LOG_PREFIXES specifies the format of messages entered into the log.
  -- The WIDTH specifies the number of columns in the log display.
  -- The LOG_FILE specifies the predefined file to be used by the Log
  -- package to generate output.
  -- The ACTIVITY specifies the activity file used for loading subsystems.
  -- The REMOTE_PASSWORDS specifies the file in which usernames and
Profile, !Tools

```

```

-- passwords for remote machines are stored.
-- The REMOTE_SESSIONS specifies the file in which session names
-- for remote machines are stored.
-- Procedures are provided for setting each of these values into
-- the map on a job or session basis, and functions are provided
-- for retrieving the current value from these maps.
-- Error_Reaction specifies how a command is to respond to errors
-- along two dimensions:
-- Perseverance whether to continue processing or stop at the first
-- error. (If a log is to be generated, the process
-- perseveres long enough to print an error message
-- regardless of the setting of this option.)
-- Exception whether or not a process is to propagate an
-- Propagation exception to its caller when it terminates a run in
-- which errors occurred. (If processing has
-- persevered, ERROR is raised.)
type Error_Reaction is (Quit, Propagate, Persevere, Raise_Error);
-- Quit Command terminates at the first error. It may log
-- the error in the job error map, but may not
-- propagate an exception to its caller
-- Propagate Command terminates at the first error by raising
-- an exception. An entry should be made to the job
-- error map. Profile_Error may be raised if no other
-- exception is appropriate.
-- Persevere Command continues after errors at its discretion.
-- Exceptions may not be propagated to its caller.
-- Raise_Error Command continues after errors at its discretion.
-- The command must raise an exception if it was unable
-- to complete successfully.
Default_Reaction : constant Error_Reaction := Persevere;
procedure Set_Reaction (Reaction : Error_Reaction);
procedure Set_Default_Reaction (Reaction : Error_Reaction);
function Reaction (Response : Response_Profile := Profile.Get)
return Error_Reaction;
function Persevere
(Response : Response_Profile := Profile.Get) return Boolean;
-- true if error reaction is Persevere or Raise_Error; i.e. if command
-- is to continue after an error
function Propagate
(Response : Response_Profile := Profile.Get) return Boolean;
-- true if error reaction is Propagate or Raise_Error; i.e., if a
-- command is to raise an exception when it is done.
-- Log Filter

```

```

type Msg_Kind is (Auxiliary_Msg, Debug_Msg, Note_Msg, Positive_Msg,
Position_Msg, Negative_Msg, Warning_Msg, Error_Msg,
Exception_Msg, Sharp_Msg, At_Msg, Dollar_Msg);

-- Log messages of any class can be filtered out of the log as it is
-- being generated using the procedures defined below.

type Log_Filter is array (Msg_Kind) of Boolean;

-- The filter specifies what types of messages are to be printed.

Quiet : constant Log_Filter := Log_Filter'(others => False);
Full : constant Log_Filter := Log_Filter'(
(Debug_Msg => False, others => True);
Terse : constant Log_Filter := Log_Filter'(
(False, False, False, others => True);
Errors : constant Log_Filter :=
Log_Filter'(Negative_Msg .. Exception_Msg => True, others => False);
Summary : constant Log_Filter :=
Log_Filter'(Positive_Msg | Negative_Msg => True, others => False);
Default_Filter : constant Log_Filter := Full;

function Filter (Response : Response_Profile := Profile.Get)
return Log_Filter;

function Includes
(Kind : Msg_Kind; Response : Response_Profile := Profile.Get)
return Boolean;

-- iff includes (Kind, Response) is true then messages of that Kind are
-- sent to the log.

procedure Include (Kind : Msg_Kind; Value : Boolean := True);
procedure Include_In_Default (Kind : Msg_Kind; Value : Boolean := True);
-- Change the filter value for the given message kind

procedure Set_Filter (Auxiliaries : Boolean := True;
Diagnostics : Boolean := True;
Notes : Boolean := True;
Positives : Boolean := True;
Negatives : Boolean := True;
Positions : Boolean := True;
Warnings : Boolean := True;
Errors : Boolean := True;
Exceptions : Boolean := True;
Sharps : Boolean := True;
Dollars : Boolean := True;
Ats : Boolean := True);

procedure Set_Default_Filter (Auxiliaries : Boolean := True;
Diagnostics : Boolean := True;
Notes : Boolean := True;
Positives : Boolean := True;
Negatives : Boolean := True;
Positions : Boolean := True;
Warnings : Boolean := True;
Errors : Boolean := True;
Exceptions : Boolean := True;
Sharps : Boolean := True);

```

```

Dollars : Boolean := True;
Ats : Boolean := True);

procedure Set_Filter (Filter : Log_Filter);
-- Establishes the given filter value(s) as the current filter value
-- for the job.

procedure Set_Default_Filter (Filter : Log_Filter);
-- Establishes the given filter value(s) as the default filter value
-- for the session.

-- Log Format
-- Specifies the prefixes desired for each message and the length of
-- the line used in formatting messages

type Log_Prefix is (Nil, Time, -- 11:00:00 PM
Hr_Mn_Sc, -- 23:00:00
Hr_Mh, -- 23:00
Date, -- September 29, 1983
Mn_Dy_Yr, -- 09/29/83
Dy_Mon_Yr, -- 29-SEP-83
Yr_Mn_Dy, -- 83/09/29
Symbols -- +, +, +, etc
);

-- Each prefix (except Nil) is followed by a single blank
type Log_Prefixes is array (1..3) of Log_Prefix;

-- Any combination of up to three prefixes may be specified
Default_Prefixes : constant Log_Prefixes := (Yr_Mn_Dy, Hr_Mn_Sc, Symbols);

function Prefixes (Response : Response_Profile := Profile.Get)
return Log_Prefixes;

procedure Set_Prefixes (Prefixes : Log_Prefixes);
procedure Set_Default_Prefixes (Prefix1, Prefix2, Prefix3 : Log_Prefix := Nil);
-- Prefix1, Prefix2, Prefix3 : Log_Prefix := Nil);

Default_Width : constant Natural := 77; -- default width of log display

procedure Set_Width (Width : Natural);
procedure Set_Default_Width (Width : Natural);

function Width (Response : Response_Profile := Profile.Get) return Natural;

-- ACTIVITY;

subtype Activity_Type is Directory.Object;

function Default_Activity return Activity_Type;

procedure Set_Activity (Activity : Activity_Type);
procedure Set_Default_Activity (Activity : Activity_Type);

```

```

function Activity (Response : Response_Profile := Profile.Get)
return Activity_Type;

-- LOG_FILE

type Log_Output_File is (Use_Output, Use_Error,
Use_Standard_Output, Use_Standard_Error);

Default_Log_File : constant Log_Output_File :=
Use_Output;
-- default file for log

procedure Set_Log_File (Log_File : Log_Output_File);
procedure Set_Default_Log_File (Log_File : Log_Output_File);

function Log_File (Response : Response_Profile := Profile.Get)
return Log_Output_File;

function Response (Reaction : Error_Reaction := Profile.Reaction;
Filter : Log_Filter := Profile.Filter;
Prefixes : Log_Prefixes := Profile.Prefixes;
Width : Natural := Profile.Width;
Activity : Activity_Type := Profile.Activity;
Log_File : Log_Output_File := Profile.Log_File)
return Response_Profile;

function Response (Reaction : Error_Reaction := Profile.Reaction;
Filter : Log_Filter := Profile.Filter;
Prefixes : Log_Prefixes := Profile.Prefixes;
Width : Natural := Profile.Width;
Activity : Activity_Type := Profile.Activity;
Log_File : Log_Output_File := Profile.Log_File);

procedure Set_Response (Reaction : Error_Reaction := Profile.Reaction;
Filter : Log_Filter := Profile.Filter;
Prefixes : Log_Prefixes := Profile.Prefixes;
Width : Natural := Profile.Width;
Activity : Activity_Type := Profile.Activity;
Log_File : Log_Output_File := Profile.Log_File);

procedure Set_Default_Response
(Reaction : Error_Reaction :=
Profile.Reaction (Profile.Get_Default);
Filter : Log_Filter := Profile.Filter (Profile.Get_Default);
Prefixes : Log_Prefixes :=
Profile.Prefixes (Profile.Get_Default);
Width : Natural := Profile.Width (Profile.Get_Default);
Activity : Activity_Type :=
Profile.Activity (Profile.Get_Default);
Log_File : Log_Output_File :=
Profile.Log_File (Profile.Get_Default));

function Ignore (Reaction : Error_Reaction := Profile.Persevere;
Filter : Log_Filter := Profile.Quiet;
Prefixes : Log_Prefixes := Profile.Prefixes;
Width : Natural := Profile.Width;
Activity : Activity_Type := Profile.Activity;
Log_File : Log_Output_File := Profile.Log_File)
return Response_Profile renames Response;

function Warn (Reaction : Error_Reaction := Profile.Persevere;
Filter : Log_Filter := Profile.Full;
Prefixes : Log_Prefixes := Profile.Prefixes;
Width : Natural := Profile.Width;

```

```

Activity : Activity_Type := Profile.Activity;
Log_File : Log_Output_File := Profile.Log_File)
return Response_Profile renames Response;

function Verbose (Reaction : Error_Reaction := Profile.Reaction;
Filter : Log_Filter := Profile.Full;
Prefixes : Log_Prefixes := Profile.Prefixes;
Width : Natural := Profile.Width;
Activity : Activity_Type := Profile.Activity;
Log_File : Log_Output_File := Profile.Log_File)
return Response_Profile renames Response;

function Raise_Exception (Reaction : Error_Reaction := Profile.Propagate;
Filter : Log_Filter := Profile.Filter;
Prefixes : Log_Prefixes := Profile.Prefixes;
Width : Natural := Profile.Width;
Activity : Activity_Type := Profile.Activity;
Log_File : Log_Output_File := Profile.Log_File)
return Response_Profile renames Response;

No_Prefixes : constant Log_Prefixes := (Nil, Nil, Nil);

function Nil (Reaction : Error_Reaction := Profile.Quit;
Filter : Log_Filter := Profile.Quiet;
Prefixes : Log_Prefixes := Profile.No_Prefixes;
Width : Natural := Profile.Default_Width;
Activity : Activity_Type := Profile.Default_Activity;
Log_File : Log_Output_File := Profile.Default_Log_File)
return Response_Profile renames Response;

Error : exception;

-----
----- String Interface -----

-- The strings accepted and returned by the following subprograms obey
-- the syntax for form parameters. The Option names and expected values
-- are as follows:

-- Option Name Value
-- REACTION QUIT, PROPAGATE, PERSEVERE, RAISE_ERROR
-- AUXILIARY_MSG, ::: TRUE or FALSE
-- DEBUG_MSG, ??? "
-- NOTE_MSG, --- "
-- POSITIVE_MSG, +++ "
-- POSITION_MSG, >>> "
-- NEGATIVE_MSG, +-+ "
-- WARNING_MSG, !!! "
-- ERROR_MSG, !!! "
-- EXCEPTION_MSG, XXX "
-- SHARP_MSG, ### "
-- AT_MSG, @@@ "
-- DOLLAR_MSG, $$$ "
-- PREFIX (form[, form[, form]]) where form is one of
-- TIME, HR_MN_SC, HR_MN,

```

```

-- DATE, MN_DY_YR, DY_MON_YR, YR_MN_DY,
-- SYMBOLS
-- e.g. (TIME, DATE, SYMBOLS)
-- (HR_MN_SC, SYMBOLS)
--
-- LINE_WIDTH      positive integer in 1..1024
-- ACTIVITY        name of activity file
-- LOG_FILE        USE_OUTPUT, USE_ERROR, USE_STANDARD_OUTPUT, or
--                USE_STANDARD_ERROR
--
-- The Options enclosed in "<" brackets take no value, but denote the
-- current value of the named profile. These values may be used by
-- themselves to denote all components of the named profile or as the value
-- of one of the profile components to denote just that component of the
-- named file, e.g. Activity=><DEFAULT>. Where special symbols appear,
-- e.g. +++, any symbols not mentioned are turned off.
--
-- <DEFAULT>      System default, ignores user profile.
-- PERSEVERE, :::, "???", ---.###, Width=>77
-- Prefix=>YR_MN_DY, HR_MN_SC SYMBOLS, USE_OUTPUT
--
-- <ERRORS>      +++, +...XXX, <PROFILE>
--
-- <IGNORE>      *:::..###. <PROFILE>
--
-- <NIL>          Empty, ignores user profile.
-- Quit, "::::.###, Prefix=>, Width=>77, USE_OUT
--
-- <PROFILE>     Values set by job/session profile.
--
-- <QUIET>       Equivalent of <NIL>
--
-- <PROGRESS>    +++, +...XXX, +...XXX, <PROFILE>
--
-- <RAISE_EXCEPTION> PROPAGATE, <PROFILE>
--
-- <SESSION_PROFILE> Use session profile rather than job.
--
-- <WARN>        +...XXX, <PROFILE>
--
-- <VERBOSE>    :::, "???", ---.###, <PROFILE>
--
-- function Get return String;
-- Profile for the current job as a form parameter
--
-- function Get_Default return String;
-- Profile for the Session (which is the default for a job that
-- does not specify one) as a form parameter
--
-- function Image (Profile : Response_Profile := Standard.Profile.Get)
-- return String;
-- function Value (Image : String := Profile.Get) return Response_Profile;
--
-- procedure Convert (Image : String;
-- Response : out Response_Profile;
-- Status : in out Simple_Status.Condition);

```

```

-- Convert between form parameter representation of a profile and the
-- internal representation. The Value function ignores invalid options
-- in the form parameter.
--
-- procedure Set (Profile : String; Status : in out Simple_Status.Condition);
-- Set profile for rest of current job; An error Status is returned and
-- the profile is not changed if the profile string is invalid.
--
-- procedure Set_Default (Profile : String;
-- Status : in out Simple_Status.Condition);
-- Set profile for session; this is the value used for the job
-- response profile if none is otherwise specified.
--
-- procedure Get_Cached_Resolution
-- (Name : String;
-- The_Declaration : out Directory_Declaration;
-- The_Object : out Directory_Object;
-- The_Version : out Directory_Version;
-- Status : out Directory_Naming.Name_Status);
--
-- Retrieve the resolution of the Name as cached at job initiation. Only
-- resolution of <IMAGE>, <CURSOR>, <REGION>, and <SELECTION> are cached.
--
-- function Cached_Selected_Text return String;
--
-- Retrieve the Selected text at job initiation.
--
-- REMOTE_PASSWORDS:
--
-- subtype Remote_Passwords_Type is Directory_Object;
-- function Remote_Passwords (Response : Response_Profile := Profile.Get)
-- return Remote_Passwords_Type;
--
-- function Default_Remote_Passwords return Remote_Passwords_Type;
-- procedure Set_Remote_Passwords (Passwords : Remote_Passwords_Type);
-- procedure Set_Default_Remote_Passwords (Passwords : Remote_Passwords_Type);
--
-- REMOTE_SESSIONS:
--
-- subtype Remote_Sessions_Type is Directory_Object;
-- function Remote_Sessions (Response : Response_Profile := Profile.Get)
-- return Remote_Sessions_Type;
--
-- function Default_Remote_Sessions return Remote_Sessions_Type;
-- procedure Set_Remote_Sessions (Sessions : Remote_Sessions_Type);
-- procedure Set_Default_Remote_Sessions (Sessions : Remote_Sessions_Type);
--
-- end Profile;

```

```

end record;
type Iterator is new Queue;
end Queue_Generic;

```

```

generic
type Element is private;
pragma Must_Be_Constrained (Yes => Element);
package Queue_Generic is
pragma Subsystem (Abstract_Types, Private_Part => Open);
pragma Module_Name (4, 712);
type Queue is private;
procedure Initialize (Q : out Queue);
function Is_Empty (Q : Queue) return Boolean;
procedure Make_Empty (Q : in out Queue);
procedure Copy (Target : in out Queue; Source : Queue);
procedure Add (Q : in out Queue; X : Element);
procedure Delete (Q : in out Queue);
function First (Q : Queue) return Element;
-- on calls to delete and first, not is_empty(q) is assumed
-- constraint error will be raises is is_empty(q)
type Iterator is private;
procedure Init (Iter : out Iterator; Q : Queue);
procedure Next (Iter : in out Iterator);
function Value (Iter : Iterator) return Element;
function Done (Iter : Iterator) return Boolean;
-- Implementation Notes and Non-Standard Operations --
-- variables of type queue are initially empty
-- therefore, the call to initialize is optional
-- := and = are meaningless
-- := implies sharing (introduces an alias) for sub-structures
-- garbage may be generated
private
type Node;
type Pointer is access Node;
type Node is
record
Value : Element;
Link : Pointer;
end record;
type Queue is
record
Head : Pointer;
Tail : Pointer;
end record;

```

Queue\_Generic, !Tools

```

with System;
package Random is
  type Handle is private;
  function Float_Value (The_Handle : Handle) return Float;
  -- Will return values in the range [0.0 .. 1.0), but note that type
  -- conversion may cause rounding.
  function Natural_Value (The_Handle : Handle; Max : Natural) return Natural;
  -- Return a value in the range 0 .. Max with uniform distribution.
  generic
    type Result_Type is (<>);
  function Enumeration_Value_Generic (The_Handle : Handle) return Result_Type;
  -- Return a uniform distribution of enumeration literals.
  function String_Value (The_Handle : Handle;
    Max_Length : Natural;
    Min_Length : Natural := 0;
    Anchored : Boolean := False) return String;
  -- Return a random string. If Anchored is true, the lower bound
  -- will always be 1.
  -----
  subtype Seed_Type is Integer;
  function Generate_Seed return Seed_Type;
  -- Construct a Seed based on the current time of day.
  procedure Initialize (The_Handle : out Handle;
    Seed : Seed_Type := Generate_Seed;
    Storage : System.Segment := System.Null_Segment);
  -- Start the generator.
  function Initial_Seed (The_Handle : Handle) return Seed_Type;
  -- Return the seed used to initialize this handle.
  function Calls (The_Handle : Handle) return Natural;
  -- Return the number of calls to this handle.
  pragma Subsystem (Tools);
  pragma Module_Name (4, 3563);
end Random;

```

```

package Script is
  procedure Pretty_Print (Script_File : String; Command_File : String);
  pragma Subsystem (Command);
  pragma Module_Name (4, 2026);
end Script;

```

```

generic
type Element is private;
-- must be a pure value
-- ie. no initialization or finalization is necessary
-- = and := are equality and copy
pragma Must_Be_Constrained (Yes => Element);

package Set_Generic is
pragma Subsystem (Tools);
pragma Module_Name (4, 3978);
type Set is private;
procedure Initialize (S : out Set);
function Is_Empty (S : Set) return Boolean;
procedure Make_Empty (S : in out Set);
procedure Copy (Target : in out Set; Source : Set);
function Is_Member (S : Set; X : Element) return Boolean;
procedure Add (S : in out Set; X : Element);
procedure Delete (S : in out Set; X : Element);
-- X is (is not) in S then the operation add (delete) is a no op.
type Iterator is private;
procedure Init (Iter : out Iterator; S : Set);
procedure Next (Iter : in out Iterator);
function Value (Iter : Iterator) return Element;
function Done (Iter : Iterator) return Boolean;
-----
-- Implementation Notes and Non-Standard Operations --
--
-- variables of type set are initially empty
-- therefore, the call to initialize is optional
-- initialize does make the set empty
-- := and = operate on references
-- := implies sharing (introduces an alias)
-- = means is the same set, not the same value of type set
-- garbage may be generated
--
-- Concurrency Properties
-- any number of read operations (is_empty, is_member) can procede
-- concurrently with one write operations (add/delete/make_empty)

private
type Node is
record
Value : Element;
Link : Set;

```

Set\_Generic, !Tools

```

end record;

type Set is access Node;
type Iterator is new Set;
end Set_Generic;

```



```

package Simple_Status is
-- Error status reporting package

-- A simple_status.condition can be used to return error information from
-- procedure calls. They are relatively large and should always
-- be passed in out (by convention to avoid copies).

-- A Condition consists of a Condition_Name and a Message.
-- The Condition_Name indicates the type of error (if any) and how
-- how serious the error is (or if completion was
-- successful). The Message provides additional information about
-- the error.

-- In simple applications, A Condition_Name alone can be used to
-- indicate status.

-- By convention, condition names in an application should be
-- standardized so that error conditions can be tested
-- programmatically.

type Condition_Name is private; -- A short name for the error type
type Condition_Class is
(Normal, -- operation completed normally
Warning, -- operation completed, but something unexpected happened
Problem, -- operation did not complete, but no harm done
Fatal); -- operation did not complete. Proceeding is dangerous.
type Condition is private; -- Contains the above plus a message
-- Conditions are self-initializing to
-- severity Normal and null names

procedure Initialize (Status : in out Condition);
-- The empty condition has null name and severity normal
-- a declared condition is initialized; This procedure will set the
-- Condition to be Normal (ie, successful).

function Name (Error_Type : Condition_Name) return String;
function Name (Status : Condition) return String;
-- get the human-readable name of this Condition_Name (Condition)

function Severity (Error_Type : Condition_Name) return Condition_Class;
function Severity (Status : Condition) return Condition_Class;

function Error_Type (Status : Condition) return Condition_Name;
-- provide the Condition_Name on which a Condition is built

function Error (Error_Type : Condition_Name;
Level : Condition_Class := Warning) return Boolean;
function Error (Status : Condition; Level : Condition_Class := Warning)
return Boolean;
-- True <=> Severity (Error_Type/Status) >= Level;
-- usage:
-- Do_Something (Status);
-- if Simple_Status.Error (Status) then
-- ... Put (Display_Message (Status));

function Display_Message (Status : Condition) return String;
Simple_Status, !Tools

```

```

-- given a condition that indicates and error, this function returns
-- a string suitable for display to users. It includes the
-- string form of the condition name and any additional problem-
-- specific information recorded in the condition.

function Message (Status : Condition) return String;
-- return just the message part of the Condition.

procedure Create_Condition (Status : in out Condition;
Error_Type : String;
Message : String := "";
Severity : Condition_Class := Problem);

procedure Create_Condition (Status : in out Condition;
Error_Type : Condition_Name;
Message : String := "");
-- Create a new error condition. The Error_Type is intended to
-- specify the class of error (limited to 63 characters), generally
-- in a few words (eg, "illegal name"). Message is intended to
-- supplement the error_type with more specific information
-- (eg, " : '#' is an illegal character"): Function Display_Message
-- would then return "Illegal name: '#' is an illegal character".

function Create_Condition_Name
(Error_Type : String; Severity : Condition_Class := Problem)
return Condition_Name;

function Equal (Status : Condition; Error_Type : String) return Boolean;
function Equal (Status : Condition; Error_Type : Condition_Name)
return Boolean;
function Equal (Status : Condition_Name; Error_Type : String)
return Boolean;
function Equal (Status : Condition_Name; Error_Type : Condition_Name)
return Boolean;

-- return true if the error_type string of Status is equal to the
-- right error_type string (the second parameter). The severity
-- does not participate in the comparison. The strings must
-- match exactly (except for index range). Sample usage:
-- Directory.Open(File, Status);
-- if SS.Equal (Status, "Nonexistent file") then ...
-- elsif SS.Equal (Status, "Internal error") then ...
-- etc.

-- The strings in the example should be constants, of course.

pragma Subsystem (Miscellaneous);
pragma Module_Name (4, 810);

end Simple_Status;

```

```

generic
  type Element is private;
  pragma Must_Be_Constrained (Yes => Element);
  package Stack_Generic is
    pragma Subsystem (Abstract_Types, Private_Part => Open);
    pragma Module_Name (4, 714);
    type Stack is private;
    Empty_Stack : constant Stack;
    -- It is expected that a declared stack is initialized to Empty_Stack
    procedure Make_Empty (S : in out Stack);
    procedure Pop (S : in out Stack);
    procedure Push (X : Element; S : in out Stack);
    function Empty (S : Stack) return Boolean;
    function Top (S : Stack) return Element;
    procedure Copy (Target : in out Stack; Source : Stack);
    Underflow : exception;
    type Iterator is private;
    procedure Init (Iter : out Iterator; S : Stack);
    procedure Next (Iter : in out Iterator);
    function Value (Iter : Iterator) return Element;
    function Done (Iter : Iterator) return Boolean;
  private
    type Stack_Mode;
    type Stack is access Stack_Mode;
    Empty_Stack : constant Stack := null;
    type Iterator is new Stack;
  end Stack_Generic;

```

```

generic
  Size : Integer;
  type Range_Type is private;
  -- Range_Type is a pure value
  -- no initialization or finalization of values of range_type is
  -- necessary
  -- = and := can be used for equality and copy
  Ignore_Case : Boolean := True;
  pragma Must_Be_Constrained (Yes => Range_Type);
  package String_Map_Generic is
    pragma Subsystem (Tools);
    pragma Module_Name (4, 3980);
    type Map is private;
    function Eval (The_Map : Map; D : String) return Range_Type;
    procedure Find (The_Map : Map;
                   D : String;
                   R : in out Range_Type;
                   Success : out Boolean);
    procedure Define (The_Map : in out Map;
                     D : String;
                     R : Range_Type;
                     Trap_Multiples : Boolean := False);
    procedure Undefine (The_Map : in out Map; D : String);
    procedure Initialize (The_Map : out Map);
    function Is_Empty (The_Map : Map) return Boolean;
    procedure Make_Empty (The_Map : in out Map);
    procedure Copy (Target : in out Map; Source : Map);
    type Iterator is private;
    procedure Init (Iter : out Iterator; The_Map : Map);
    procedure Next (Iter : in out Iterator);
    function Value (Iter : Iterator) return String;
    function Done (Iter : Iterator) return Boolean;
  Undefine : exception;
  -- raised by eval if the domain value is not in the map
  Multiply_Defined : exception;
  -- raised by define if the domain value is already defined and
  -- the trap_multiples flag has been specified (i.e. is true)
  function Nil return Map;
  function Is_Nil (The_Map : Map) return Boolean;
  function Cardinality (The_Map : Map) return Natural;
  -- Implementation Notes and Non-Standard Operations --
  String_Map_Generic, !Tools

```

```

-----
-- := and = operate on references
-- := implies sharing (introduces an alias)
-- = means is the same set, not the same value of type set
-- Initializing a map also makes it empty
-- Accessing an uninitialized map will raise CONSTRAINT_ERROR.
-- garbage may be generated

private
subtype Index is Natural range 0 .. Size - 1;
type Node (Size : Natural);
type Set is access Node;

type Table is array (Index) of Set;
type Map_Data is
record
  Bucket : Table;
  Size : Integer := 0;
end record;
type Map is access Map_Data;

type Iterator is
record
  The_Map : Map;
  Index_Value : Index;
  Set_Iter : Set;
  Done : Boolean;
end record;

type Node (Size : Natural) is
record
  Link : Set;
  Value : Range_Type;
  Name : String (1 .. Size);
end record;

end String_Map_Generic;

```

```

package String_Table is
pragma Subsystem (Tools, Private_Part => Closed);
pragma Module_Name (4, 3981);
type Item is private;
type Table is private;
Table_Full : exception;
-- create a table for unique strings
function New_Table (Minimum_Table_Size : Natural := 127) return Table;
function Nil return Item;
-- return unique item in table, ignore_case => upper_case storage
function Unique (Source : String;
  In_Table : Table;
  Ignore_Case : Boolean := True) return Item;
-- return item if present, otherwise Nil
function Find (Source : String;
  In_Table : Table;
  Ignore_Case : Boolean := True) return Item;
-- return an item without entering in table
function Allocate (Source : String; In_Table : Table) return Item;
-- compare strings for identity, then same contents
function Equal (L, R : Item) return Boolean;
-- representation of string, suitable for hashing
function Unique_Index (U : Item) return Integer;
-- value of character or entire string
function Char_At (Source : Item; At_Pos : Natural) return Character;
function Image (Source : Item) return String;
function Length (Source : Item) return Natural;
function Is_Nil (Source : Item) return Boolean;
type Iterator is private;
procedure Init (Iter : out Iterator; The_Table : Table);
procedure Next (Iter : in out Iterator);
function Value (Iter : Iterator) return Item;
function Done (Iter : Iterator) return Boolean;
end String_Table;

```

```

package String_Utillities is
function Hash_String (S : String) return Integer;
procedure Upper_Case (C : in out Character);
procedure Lower_Case (C : in out Character);
function Upper_Case (C : Character) return Character;
function Lower_Case (C : Character) return Character;
procedure Upper_Case (S : in out String);
procedure Lower_Case (S : in out String);
-- string returned has same 'First and 'Last as S
function Upper_Case (S : String) return String;
function Lower_Case (S : String) return String;

function Number_To_String (Value : Integer;
Base : Natural := 10;
Width : Natural := 0;
Leading : Character := ' ') return String;

function Number_To_String (Value : Long_Integer;
Base : Natural := 10;
Width : Natural := 0;
Leading : Character := ' ') return String;

procedure String_To_Number (Source : String;
Target : out Integer;
Worked : out Boolean;
Base : Natural := 10);

procedure String_To_Number (Source : String;
Target : out Long_Integer;
Worked : out Boolean;
Base : Natural := 10);

function Strip_Leading
(From : String; Filler : Character := ' ') return String;
function Strip_Trailing
(From : String; Filler : Character := ' ') return String;
function Strip (From : String; Filler : Character := ' ') return String;

-- Searches and compares
-- Locate returns the index value in Within if found, 0 otherwise
function Locate (Fragment : String;
Within : String;
Ignore_Case : Boolean := True) return Natural;

function Locate (Fragment : Character;
Within : String;
Ignore_Case : Boolean := True) return Natural;

function Reverse_Locate (Fragment : String;
Within : String;
Ignore_Case : Boolean := True) return Natural;

```

```

function Reverse_Locate (Fragment : Character;
Within : String;
Ignore_Case : Boolean := True) return Natural;

function Equal (Str1 : String; Str2 : String; Ignore_Case : Boolean := True)
return Boolean;

function Less_Than
(Str1 : String; Str2 : String; Ignore_Case : Boolean := True)
return Boolean;

function Greater_Than
(Str1 : String; Str2 : String; Ignore_Case : Boolean := True)
return Boolean;

procedure Capitalize (S : in out String);
function Capitalize (S : String) return String;

pragma Subsystem (Tools);
pragma Module_Name (4, 303);

end String_Utillities;

```

```

with Directory;
with Machine;
with Calendar;
with System;

package System_Uilities is

pragma Subsystem (Tools, Closed);
pragma Module_Name (4, 3973);

subtype Job_Id is Machine.Job_Id range 4 .. 255;
subtype Session_Id is Machine.Session_Id;
subtype Version is Directory.Version;
subtype Object is Directory.Object;
subtype Port is Natural range 0 .. 4 * 16 + 16;
subtype Tape is Natural range 0 .. 4;
subtype Byte_String is System.Byte_String;

-- Job (Process) characteristics
function Get_Job return Job_Id;
function Priority
  (For_Job : Job_Id := System_Uilities.Get_Job) return Natural;
function Elapsed
  (For_Job : Job_Id := System_Uilities.Get_Job) return Duration;
function Opu
  (For_Job : Job_Id := System_Uilities.Get_Job) return Duration;
function Job_Name
  (For_Job : Job_Id := System_Uilities.Get_Job) return String;

-- Active Session characteristics
function Get_Session return Session_Id;
function Get_Session (For_Job : Job_Id) return Session_Id;
function Session_Name
  (For_Session : Session_Id := System_Uilities.Get_Session)
  return String;
function User_Name
  (For_Session : Session_Id := System_Uilities.Get_Session)
  return String;
function Terminal
  (For_Session : Session_Id := System_Uilities.Get_Session)
  return Port;
function Terminal
  (For_Session : Session_Id := System_Uilities.Get_Session)
  return Version;
function Terminal
  (For_Session : Session_Id := System_Uilities.Get_Session)
  return Object;
function Session
  (For_Session : Session_Id := System_Uilities.Get_Session)
  return Version;
function Session
  (For_Session : Session_Id := System_Uilities.Get_Session)
  return Object;
function User
  (For_Session : Session_Id := System_Uilities.Get_Session)
  return Version;
function User
  (For_Session : Session_Id := System_Uilities.Get_Session)
  return Object;

-- Inactive Session characteristics
function Home_Library (User : String := User_Name) return String;
function Last_Login (User : String; Session : String := "")
  return Calendar.Time;
function Last_Logout (User : String; Session : String := "")
  return Calendar.Time;
function Logged_In (User : String; Session : String := "") return Boolean;

-- Names for Text_IO/Simple_Text_IO standard file names
function Output_Name
  (For_Session : Session_Id := System_Uilities.Get_Session)
  return String;
function Input_Name
  (For_Session : Session_Id := System_Uilities.Get_Session)
  return String;
function Error_Name
  (For_Session : Session_Id := System_Uilities.Get_Session)
  return String;
function Tape_Name (Drive : Tape := 0) return String;

-- Terminal characteristics
subtype Stop_Bits_Range is Integer range 1 .. 2;
subtype Character_Bits_Range is Integer range 5 .. 8;
type Parity_Kind is (None, Even, Odd);

function Terminal_Name
  (Line : Port := System_Uilities.Terminal) return String;
function Terminal_Type
  (Line : Port := System_Uilities.Terminal) return String;
function Input_Count
  (Line : Port := System_Uilities.Terminal) return Long_Integer;
function Output_Count
  (Line : Port := System_Uilities.Terminal) return Long_Integer;
-- The number of characters input/output from/to the specified terminal
-- since the machine was booted. Input from the terminal that has not
-- been read by a session or user program will not be counted as input.
function Input_Rate
  (Line : Port := System_Uilities.Terminal) return String;
function Output_Rate
  (Line : Port := System_Uilities.Terminal) return String;
function Parity
  (Line : Port := System_Uilities.Terminal) return String;
function Stop_Bits
  (Line : Port := System_Uilities.Terminal)
  return Parity_Kind;
function Stop_Bits
  (Line : Port := System_Uilities.Terminal)
  return Stop_Bits_Range;
function Character_Size
  (Line : Port := System_Uilities.Terminal)
  return Character_Bits_Range;
function Xon_Xoff_Characters
  (Line : Port := System_Uilities.Terminal) return String;
function Xon_Xoff_Bytes
  (Line : Port := System_Uilities.Terminal) return Byte_String;
function Receive_Xon_Xoff_Characters
  (Line : Port := System_Uilities.Terminal) return String;
function Receive_Xon_Xoff_Bytes
  (Line : Port := System_Uilities.Terminal) return String;

```

```

package System_Uilities is

pragma Subsystem (Tools, Closed);
pragma Module_Name (4, 3973);

subtype Job_Id is Machine.Job_Id range 4 .. 255;
subtype Session_Id is Machine.Session_Id;
subtype Version is Directory.Version;
subtype Object is Directory.Object;
subtype Port is Natural range 0 .. 4 * 16 + 16;
subtype Tape is Natural range 0 .. 4;
subtype Byte_String is System.Byte_String;

-- Job (Process) characteristics
function Get_Job return Job_Id;
function Priority
  (For_Job : Job_Id := System_Uilities.Get_Job) return Natural;
function Elapsed
  (For_Job : Job_Id := System_Uilities.Get_Job) return Duration;
function Opu
  (For_Job : Job_Id := System_Uilities.Get_Job) return Duration;
function Job_Name
  (For_Job : Job_Id := System_Uilities.Get_Job) return String;

-- Active Session characteristics
function Get_Session return Session_Id;
function Get_Session (For_Job : Job_Id) return Session_Id;
function Session_Name
  (For_Session : Session_Id := System_Uilities.Get_Session)
  return String;
function User_Name
  (For_Session : Session_Id := System_Uilities.Get_Session)
  return String;
function Terminal
  (For_Session : Session_Id := System_Uilities.Get_Session)
  return Port;
function Terminal
  (For_Session : Session_Id := System_Uilities.Get_Session)
  return Version;
function Terminal
  (For_Session : Session_Id := System_Uilities.Get_Session)
  return Object;
function Session
  (For_Session : Session_Id := System_Uilities.Get_Session)
  return Version;
function Session
  (For_Session : Session_Id := System_Uilities.Get_Session)
  return Object;
function User
  (For_Session : Session_Id := System_Uilities.Get_Session)
  return Version;
function User
  (For_Session : Session_Id := System_Uilities.Get_Session)
  return Object;

-- Inactive Session characteristics
function Home_Library (User : String := User_Name) return String;
function Last_Login (User : String; Session : String := "")
  return Calendar.Time;
function Last_Logout (User : String; Session : String := "")
  return Calendar.Time;
function Logged_In (User : String; Session : String := "") return Boolean;

-- Names for Text_IO/Simple_Text_IO standard file names
function Output_Name
  (For_Session : Session_Id := System_Uilities.Get_Session)
  return String;
function Input_Name
  (For_Session : Session_Id := System_Uilities.Get_Session)
  return String;
function Error_Name
  (For_Session : Session_Id := System_Uilities.Get_Session)
  return String;
function Tape_Name (Drive : Tape := 0) return String;

-- Terminal characteristics
subtype Stop_Bits_Range is Integer range 1 .. 2;
subtype Character_Bits_Range is Integer range 5 .. 8;
type Parity_Kind is (None, Even, Odd);

function Terminal_Name
  (Line : Port := System_Uilities.Terminal) return String;
function Terminal_Type
  (Line : Port := System_Uilities.Terminal) return String;
function Input_Count
  (Line : Port := System_Uilities.Terminal) return Long_Integer;
function Output_Count
  (Line : Port := System_Uilities.Terminal) return Long_Integer;
-- The number of characters input/output from/to the specified terminal
-- since the machine was booted. Input from the terminal that has not
-- been read by a session or user program will not be counted as input.
function Input_Rate
  (Line : Port := System_Uilities.Terminal) return String;
function Output_Rate
  (Line : Port := System_Uilities.Terminal) return String;
function Parity
  (Line : Port := System_Uilities.Terminal) return String;
function Stop_Bits
  (Line : Port := System_Uilities.Terminal)
  return Parity_Kind;
function Stop_Bits
  (Line : Port := System_Uilities.Terminal)
  return Stop_Bits_Range;
function Character_Size
  (Line : Port := System_Uilities.Terminal)
  return Character_Bits_Range;
function Xon_Xoff_Characters
  (Line : Port := System_Uilities.Terminal) return String;
function Xon_Xoff_Bytes
  (Line : Port := System_Uilities.Terminal) return Byte_String;
function Receive_Xon_Xoff_Characters
  (Line : Port := System_Uilities.Terminal) return String;
function Receive_Xon_Xoff_Bytes
  (Line : Port := System_Uilities.Terminal) return String;

```

```

-- returns a 2-element string consisting of Xon followed by Xoff
function Flow_Control
  (Line : Port := System_Uilities.Terminal) return String;
function Receive_Flow_Control
  (Line : Port := System_Uilities.Terminal) return String;
-- return one of NONE, XON_XOFF, RTS, DTR
function Enabled (Line : Port := System_Uilities.Terminal) return Boolean;
function Disconnect_On_Disconnect
  (Line : Port := System_Uilities.Terminal) return Boolean;
function Logoff_On_Disconnect
  (Line : Port := System_Uilities.Terminal) return Boolean;
function Disconnect_On_Logoff
  (Line : Port := System_Uilities.Terminal) return Boolean;
function Disconnect_On_Failed_Login
  (Line : Port := System_Uilities.Terminal) return Boolean;
function Log_Failed_Logins
  (Line : Port := System_Uilities.Terminal) return Boolean;
function Login_Disabled
  (Line : Port := System_Uilities.Terminal) return Boolean;
function Detach_On_Disconnect
  (Line : Port := System_Uilities.Terminal) return Boolean;
-- Iterate over all active sessions
type Session_Iterator is private;
procedure Init (Iter : out Session_Iterator);
function Value (Iter : Session_Iterator) return Session_Id;
function Done (Iter : Session_Iterator) return Boolean;
procedure Next (Iter : in out Session_Iterator);
-- Iterate over all jobs for a session
type Job_Iterator is private;
procedure Init (Iter : out Job_Iterator);
function Value (Iter : Job_Iterator) return Job_Id;
function Done (Iter : Job_Iterator) return Boolean;
procedure Next (Iter : in out Job_Iterator);
type Terminal_Iterator is private;
procedure Init (Iter : out Terminal_Iterator);
function Value (Iter : Terminal_Iterator) return Natural;
function Done (Iter : Terminal_Iterator) return Boolean;
procedure Next (Iter : in out Terminal_Iterator);
function System_Up_Time return Calendar.Time;
function System_Boot_Configuration return String;
function Image (Version : Directory.Version) return String;
procedure Set_Page_Limit (Max_Pages : Natural;
  For_Job : Job_Id := System_Uilities.Get_Job);
-- Set the upper limit for pages created by the specified job.

```

```

-- Attempts to create additional pages result in Storage_Error.
-- Requires operator capability if For_Job specifies a job belonging
-- to a user different from the caller. If For_Job parameter defaults,
-- then the limit applies to the calling job. In the worst case,
-- the job can allocate twice Max_Pages pages before getting a storage
-- error. Raises constraint_error if the job_id is illegal.
procedure Get_Page_Counts (Cache_Pages : out Natural;
  Disk_Pages : out Natural;
  Max_Pages : out Natural;
  For_Job : Job_Id := System_Uilities.Get_Job);
-- Return the counts for the specified job. Cache_Pages is the number of
-- pages presently in main memory; Disk_Pages is the number of pages that
-- have disk space allocated for them. Max_Pages is the current page
-- limit.
-- Operations for reading machine information:
type Bad_Block_Kinds is new Long_Integer range 0 .. 7;
Manufacturers_Bad_Blocks : constant Bad_Block_Kinds := 1;
Retargeted_Blocks : constant Bad_Block_Kinds := 2;
All_Bad_Blocks : constant Bad_Block_Kinds := 3;
type Block_List is array (Natural range <>) of Integer;
function Bad_Block_List
  (For_Volume : Natural;
  Kind : Bad_Block_Kinds := Retargeted_Blocks) return Block_List;
-- Return the list of bad blocks of the specified kind on the specified
-- disk. Return null array if kind or volume are illegal.
function Get_Board_Info (Board : Natural) return String;
-- return information about the specified board in the machine. The
-- string identifies the information.
-- Board specifies the particular board:
-- 0 : IOA/IOC
-- 1 : SYS
-- 2 : SEQ
-- 3 : VAL
-- 4 : TYP
-- 5 : FIU
-- 100 : MEM0
-- 101 : MEM1
-- 102 : MEM2
-- 103 : MEM3
-- etc.
end System_Uilities;

```

with Io;

```
-- This package is used to produce neatly formatted tables with centered
-- headers and even amounts of white space between the columns. The first
-- N calls should be to header, which defines a header and a type of
-- justification for the items that will go into each column. Then the M*N
-- items of an M line table are sent into the package a row at a time. An
-- item is defined by either a single call to Item, or a series of zero or
-- more calls to Subitem terminated by a call to Last_Subitem. Multiple
-- parts of an item are separated by the subitem separator. After all the
-- items have been defined, the table is output with a call to Display.

-- The package internally allocates enough memory to save a copy of the
-- entire table. It is therefore a good idea to instantiate this
-- procedure in a local frame so that all the memory it allocates will go
-- away when the frame does.

-- An instantiation of this package will generate at most one table.
-- It is NOT possible to start over calling Header after Display.
```

generic

```
Number_Of_Columns : Positive;
Subitem_Separator : String := " ";
package Table_Formatter is
```

```
type Adjust is (Left, Right, Centered);
procedure Header (S : String; Format : Adjust := Left);
procedure Item (S : String);
procedure Subitem (S : String);
procedure Last_Subitem;
procedure Display (On_File : Io.File_Type);
```

```
type Field_List is array (Integer range <>) of Integer;
-- For N > 0 sort by field N in increasing order.
-- For N < 0 sort by field abs (N) in decreasing order.
-- Sorting is done on input value before right adjustment or centering.
```

```
procedure Sort (On_Field : Integer := 1);
procedure Sort (On_Fields : Field_List);
```

```
pragma Subsystem (Input_Output);
pragma Module_Name (4, 3217);
end Table_Formatter;
```

```
generic
type Element is private;
pragma Must_Be_Constrained (Yes => Element);

type Index is (<>);
type Element_Array is array (Index range <>) of Element;
with function "<" (Left, Right : Element) return Boolean is <>;
procedure Table_Sort_Generic (Table : in out Element_Array);
pragma Subsystem (Interface);
pragma Module_Name (4, 3979); with Device_Independent_Io;
```

```

package Tape_Tools is
pragma Subsystem (Input_Output);
pragma Module_Name (4, 3974);

type Logical_Device is private;
subtype Vol_Id_String is String; -- (1 .. 6);
subtype Vol_Name_String is String; -- (1 .. 76);
subtype Owner_String is String; -- (1 .. 13);
subtype File_Id_String is String; -- (1 .. 17);
subtype File_Name_String is String; -- (1 .. 532);
subtype User_Label_Number is Natural range 0 .. 255;

subtype Byte is Device_Independent_IO.Byte;
subtype Bytes is Device_Independent_IO.Byte_String;

procedure Initialize (Tape : out Logical_Device;
Format : String := "R1000");

function Initialize (Format : String := "R1000") return Logical_Device;

procedure Connect_For_Input (Tape : in out Logical_Device;
Vol_Id : Vol_Id_String;
Vol_Name : Vol_Name_String := "";
To_Operator : String := "Thank You");

-- Request tape with given Vol_Id/Vol_Name be mounted on a drive for
-- the purpose of reading.

procedure Connect_For_Output (Tape : in out Logical_Device;
Vol_Id : Vol_Id_String;
Vol_Name : Vol_Name_String := "";
Owner : String := "";
To_Operator : String := "Thank You");

-- request a tape be mounted on a drive for writing; Assign the
-- Vol_Id/Vol_Name/Owner per given parameters;

function Vol_Id (Tape : Logical_Device) return String;
-- Volume Id of mounted tape.

function Vol_Name (Tape : Logical_Device) return String;
-- Volume name of mounted tape (format dependent)

function Owner (Tape : Logical_Device) return String;
-- Owner of mounted tape (format dependent)

procedure Label (Tape : Logical_Device;
Number : User_Label_Number;
Label : String);

-- Define a User Label for the next Create.
-- Up to 256 labels may be defined for one file

type Record_Format is (Fixed_Length, Variable_Length, Spanned, Undefined);
Default_Record_Format : constant Record_Format := Variable_Length;
Default_Record_Length : constant Natural := 512;
Default_Block_Length : constant Natural := 2048;

procedure Format (Tape : Logical_Device;
Kind : Record_Format := Default_Record_Format;
Record_Length : Natural := Default_Record_Length;
Block_Length : Natural := Default_Block_Length);

-- Define the record format for the next Create. When the
-- Logical_Device is initialized the record format is set to the above
-- defaults. An changes made by this procedure remain in effect until
-- the next call of this procedure or the tape is disconnected.

procedure Create (Tape : in out Logical_Device;
Id : File_Id_String;
Name : File_Name_String := "");

-- Start a new output file with the given Id (and Name) and any user
-- labels defined before this call.

procedure Open (Tape : in out Logical_Device);

-- Open the next file on the tape;

function File_Id (Tape : Logical_Device) return String;

-- File Id of currently open tape file.

function File_Name (Tape : Logical_Device) return String;

-- File name of file opened/created on tape; (Format dependent);

function Labels (Tape : Logical_Device) return Natural;

-- number of user labels associated with the currently open file

function Label (Tape : Logical_Device; Index : Natural) return String;
function Label (Tape : Logical_Device; Index : Natural) return Natural;

-- Index-th user label text or number associated with currently open file.
-- Index must be less than the value returned by Labels

function Format (Tape : Logical_Device) return Record_Format;

-- The record format of the currently open file

function Block_Length (Tape : Logical_Device) return Natural;

-- The block length of the currently open file

function Record_Length (Tape : Logical_Device) return Natural;

-- The Record Length of the currently open file

procedure Skip (Tape : Logical_Device; Number : Integer := 1);

procedure Put_Line (Tape : Logical_Device; Line : String);

```



```

function Get_Line (Tape : Logical_Device) return String;
procedure Get_Line (Tape : Logical_Device;
  Line : out String;
  Length : out Natural);
procedure Get_Line (Tape : Logical_Device;
  Line : out String;
  Length : out Natural;
  Eof : out Boolean);

procedure Put (Tape : Logical_Device; Data : Bytes);
function Get (Tape : Logical_Device) return Bytes;
procedure Get (Tape : Logical_Device;
  Data : out Bytes;
  Length : out Natural;
  Eof : out Boolean);

function End_Of_File (Tape : Logical_Device) return Boolean;
function End_Of_Tape (Tape : Logical_Device) return Boolean;

procedure Close (Tape : Logical_Device);
procedure Disconnect (Tape : Logical_Device);
procedure Abandon (Tape : Logical_Device);

type Condition is (No_Errors, Vol_Already_Open_Or_Created,
  Vol_Not_Open_Or_Created, Not_Initialized,
  Not_Original_Client, Read_While_Writing,
  Write_While_Reading, Position_While_Writing,
  Previous_Fatal_Error, File_Still_Being_Written,
  File_Still_Being_Read, File_Not_Created,
  File_Not_Open, Retry_Count_Exhausted, Vol_Set_Error,
  Unexpected_Tape_Error, No_Drive_Available,
  Desired_Drive_Unavailable, Desired_Volume_Not_Found,
  Vol_Access_Denied, File_Access_Denied,
  File_Expired, End_Of_Vol_Set_Encountered,
  Incorrect_File_Seq_No, Incorrect_File_Sect_No,
  Need_Vol_Completion, Not_At_Eof, Not_At_Eov, Not_At_Eovs,
  Incorrect_Buffer_Size, Block_Length_Too_Short,
  Block_Length_Too_Long, File_Not_In_Vol_Set,
  Record_Not_In_File, Format_Violation,
  Unimplemented_Format, Attempt_To_Read_While_Writing,
  Attempt_To_Write_While_Reading, Other_Error);

function Status (Tape : Logical_Device) return Condition;
function Status (Tape : Logical_Device) return String;
function Is_Fatal (Error : Condition) return Boolean;
function Is_Fatal (Tape : Logical_Device) return Boolean;

private
type Logical_Device_Record;
type Logical_Device is access Logical_Device_Record;
pragma Segment_Heap (Logical_Device);
end Tape_Tools;

```

```

with Calendar;
package Time_Uilities is

Minute : constant Duration := 60.0;
Hour : constant Duration := 3600.0;
Day : constant Duration := 86_400.0;

-----
-- Time_Uilities.Time is a segmented version of Calendar.Time
-- with image and value functions
-----

type Years is new Calendar.Year_Number;
type Months is (January, February, March, April, May, June, July,
  August, September, October, November, December);
type Days is new Calendar.Day_Number;

type Hours is new Integer range 1 .. 12;
type Minutes is new Integer range 0 .. 59;
type Seconds is new Integer range 0 .. 59;

type Sun_Positions is (Am, Pm);

type Time is
record
  Year : Years;
  Month : Months;
  Day : Days;
  Hour : Hours;
  Minute : Minutes;
  Second : Seconds;
  Sun_Position : Sun_Positions;
end record;

function Get_Time return Time;

function Convert_Time (Date : Calendar.Time) return Time;
function Convert_Time (Date : Time) return Calendar.Time;

function Nil return Time;
function Is_Nil (Date : Time) return Boolean;

function Nil return Calendar.Time;
function Is_Nil (Date : Calendar.Time) return Boolean;

type Time_Format is (Expanded, -- 11:00:00 PM
  Military, -- 23:00:00
  Short, -- 23:00
  Ada -- 23_00_00
  );

type Date_Format is (Expanded, -- September 29, 1983
  Month_Day_Year, -- 09/29/83
  Day_Month_Year, -- 29-SEP-83
  Year_Month_Day, -- 83/09/29
  Ada -- 83_09_29
  );

```

```

type Image_Contents is (Both, Time_Only, Date_Only);

function Image (Date : Time;
               Date_Style : Date_Format := Time_Utilities.Expanded;
               Time_Style : Time_Format := Time_Utilities.Expanded;
               Contents : Image_Contents := Time_Utilities.Both)
return String;

function Value (S : String) return Time;

-- Time_Utilities.Interval is a segmented version of Duration
-- with image and value functions

type Day_Count is new Integer range 0 .. Integer'Last;
type Military_Hours is new Integer range 0 .. 23;
type Milliseconds is new Integer range 0 .. 999;

type Interval is
record
  Elapsed_Days : Day_Count;
  Elapsed_Hours : Military_Hours;
  Elapsed_Minutes : Minutes;
  Elapsed_Seconds : Seconds;
  Elapsed_Milliseconds : Milliseconds;
end record;

function Convert (I : Interval) return Duration;
function Convert (D : Duration) return Interval;

function Image (I : Interval) return String;
function Value (S : String) return Interval;

function Image (D : Duration) return String;

function Duration_Until (T : Time) return Duration;
function Duration_Until (T : Calendar.Time) return Duration;
function Duration_Until_Next
(H : Military_Hours; M : Minutes := 0; S : Seconds := 0)
return Duration;

-- Day of week support; Monday is 1.
type Weekday is new Positive range 1 .. 7;

function Day_Of_Week (T : Calendar.Time) return Weekday;
function Day_Of_Week (T : Time := Time_Utilities.Get_Time) return Weekday;
function Image (D : Weekday) return String;

function "+" (D : Weekday; I : Integer) return Weekday;
function "-" (D : Weekday; I : Integer) return Weekday;

pragma Subsystem (Tools);
pragma Module_Name (4, 3972);

end Time_Utilities;

```

```

generic
  Default_Maximum_Length : Natural := 20;
package Unbounded_String is

  pragma Subsystem (Tools, Private_Part => Open);
  pragma Module_Name (4, 3983);

  -- Managed Pointer Sequential Unbounded Strings:
  -- Restrictions and assumptions
  -- 1. Storage management is performed
  -- 2. Extending a string in a way that requires a new allocation allows
  --    space for expansion.
  -- 3. CANNOT be used by multiple tasks unless user provides serialization
  --    := is reference copy, use copy to assign contents
  -- 4. Uninitialized or Freed objects are true null's and changes to one
  --    of the referents will not be reflected in the other;
  -- 5. Use Free prior to assignment to prevent garbage
  -- 6. = is object identity, compare Images for value equality
  -- 7. = is object identity, compare Images for value equality

  subtype String_Length is Natural;
  type Variable_String is private;

  -- release storage associated with a string
  procedure Free (V : in out Variable_String);

  -- Get information about current length or contents of a string
  function Length (Source : Variable_String) return String_Length;
  function Char_At (Source : Variable_String; At_Pos : Positive)
return Character;

  function Extract (Source : Variable_String;
                  Start_Pos : Positive;
                  End_Pos : Natural) return String;

  function Image (V : Variable_String) return String;
  function Value (S : String) return Variable_String;

  -- Image (Target) := Image (Source);
  procedure Copy (Target : in out Variable_String; Source : Variable_String);
  procedure Copy (Target : in out Variable_String; Source : String);
  procedure Copy (Target : in out Variable_String; Source : Character);

  -- Target := Source; Source := ""; with appropriate storage management
  procedure Move (Target : in out Variable_String;
                 Source : in out Variable_String);

  -- Target := Target & Source;
  procedure Append (Target : in out Variable_String;
                  Source : Variable_String);

  procedure Append (Target : in out Variable_String; Source : String);

  procedure Append (Target : in out Variable_String; Source : Character);

  Unbounded_String, Tools
end Unbounded_String;

```

```

Count : String_Length);

-- Target := Target (1..At_Pos-1) & Source & Target (At_Pos..Target'Length)
procedure Insert (Target : in out Variable_String;
  At_Pos : Positive;
  Source : Variable_String);

procedure Insert (Target : in out Variable_String;
  At_Pos : Positive;
  Source : String);

procedure Insert (Target : in out Variable_String;
  At_Pos : Positive;
  Source : Character);

procedure Insert (Target : in out Variable_String;
  At_Pos : Positive;
  Source : Character);
Count : String_Length);

-- Target (At_Pos .. At_Pos + Count - 1) := "";
procedure Delete (Target : in out Variable_String;
  At_Pos : Positive;
  Count : String_Length := 1);

-- Target (At_Pos .. At_Pos + Source'Length - 1) := Source;
procedure Replace (Target : in out Variable_String;
  At_Pos : Positive;
  Source : Character);
Count : String_Length);

procedure Replace (Target : in out Variable_String;
  At_Pos : Positive;
  Source : Character);
Count : String_Length);

procedure Replace (Target : in out Variable_String;
  At_Pos : Positive;
  Source : String);

procedure Replace (Target : in out Variable_String;
  At_Pos : Positive;
  Source : Variable_String);

-- Target'Length := New_Length;
-- Target (Target'Length .. New_Length) := Fill_Mith;
procedure Set_Length (Target : in out Variable_String;
  New_Length : String_Length;
  Fill_Mith : Character := ' ');

-- Determine if a Variable_String is null; different from ""
function Is_Nil (V : Variable_String) return Boolean;

-- Return a null Variable_String. Note that assignment of Nil may
-- create garbage; see procedure Free above.
function Nil return Variable_String;

private
type Pointer is access String;

```

```

type Real_String;
type Variable_String is access Real_String;
subtype String_Bound is Integer range -1 .. Integer'Last;

type Real_String is
  record
    Length : String_Bound;
    Contents : Pointer;
    Next_Free : Variable_String;
  end record;

Null_String : Pointer := new String (1 .. 0);

Free_List_Item : constant String_Bound := -1;

Free_List : Real_String :=
  Real_String'(Free_List_Item, Null_String,
  new Real_String'(Free_List_Item, Null_String, null));
end Unbounded_String;

```

with System;

package Unchecked\_Conversions is

```
generic
  type Source is limited private;
  type Target is limited private;
package Unchecked_Conversion_Package is
```

```
  function Convert (S : Source) return Target;
  -- Package for of LHM Unchecked_Conversion.
```

```
  -- Type-specific calculations are made during package elaboration, so
  -- calls to this convert faster than to an equivalent Unchecked_Conve
  -- instantiation. Speed improvement depends on the type involved.
end Unchecked_Conversion_Package;
```

```
generic
```

```
  type Source is limited private;
  function Convert_To_Byte_String (S : Source) return System.Byte_String;
  -- Convert from Source to a byte string. The byte string may contain
  -- more bits than the object.
```

```
generic
```

```
  type Target is limited private;
  function Convert_From_Byte_String (S : System.Byte_String) return Target;
  -- Convert from a byte string to a Target type. The string should
  -- have been produced by an instantiation of Convert_To_Byte_String
  -- with the same type. A constrained object is always returned.
```

```
pragma Subsystem (Miscellaneous);
```

```
pragma Module_Name (4, 3543);
end Unchecked_Conversions;
```

package Xref is

```
  procedure Used_By (List_Of_Names : String := "<IMAGE>";
    Do_Functions : Boolean := True;
    Do_Generics : Boolean := True;
    Do_Procedures : Boolean := True;
    Do_Attributes : Boolean := False;
    Do_Record_Components : Boolean := False;
    Do_Constants : Boolean := False;
    Do_Entries : Boolean := False;
    Do_Exceptions : Boolean := False;
    Do_Labels : Boolean := False;
    Do_Packages : Boolean := False;
    Do_Parameters : Boolean := False;
    Do_Programs : Boolean := False;
    Do_Task_Bodies : Boolean := True;
    Do_Types : Boolean := False;
    Do_Variables : Boolean := False;
    Exclude_References_From : String := "";
    List_File_Name : String := "");
```

```
-- Produce a report showing all of the units that reference (use) something
-- defined in the units specified in "List_Of_Names". Only the IDs defined
-- in units specified in "List_Of_Names" will be included in the report.
-- "Using" units can be excluded by listing their names in
-- "Exclude_References_From".
```

```
  procedure Uses (List_Of_Names : String := "<IMAGE>";
```

```
    Visible_Declarations_Only : Boolean := True;
    Do_Functions : Boolean := True;
    Do_Generics : Boolean := True;
    Do_Procedures : Boolean := True;
    Do_Attributes : Boolean := False;
    Do_Record_Components : Boolean := False;
    Do_Constants : Boolean := False;
    Do_Entries : Boolean := False;
    Do_Exceptions : Boolean := False;
    Do_Labels : Boolean := False;
    Do_Packages : Boolean := False;
    Do_Parameters : Boolean := False;
    Do_Programs : Boolean := False;
    Do_Task_Bodies : Boolean := True;
    Do_Types : Boolean := False;
    Do_Variables : Boolean := False;
    Exclude_References_To : String := "";
    Only_References_To : String := "";
    List_File_Name : String := "");
```

```
-- produce a report of the items referenced by the units specified
-- in "list_of_names". Units mentioned in "Exclude_references_to"
-- are not included in the report. If units are mentioned in
-- Only_references_to then these are the only units included in the
-- report. It is an error to specify both Exclude... and Only...
end Xref;
```

### STANDARD ABBREVIATIONS

The following abbreviations are provided by the Rational Environment to reduce the typing required for executing frequently used commands and referring to commonly used units. These abbreviations are defined in world !Commands.Abbreviations and are visible through an entry on your searchlist.

The unit name abbreviations defined by links are:

LINK NAME	=>	SOURCE
ACL	=>	!COMMANDS.ACCESS_LIST
COMP	=>	!COMMANDS.COMPILE
DIR	=>	!COMMANDS.LIBRARY
FILE	=>	!COMMANDS.FILE_UTILITIES
LIB	=>	!COMMANDS.LIBRARY
LINK	=>	!COMMANDS.LINKS
MSG	=>	!COMMANDS.MESSAGE
OP	=>	!COMMANDS.OPERATOR
Q	=>	!COMMANDS.QUEUE
SA	=>	!COMMANDS.ARCHIVE
SL	=>	!COMMANDS.SEARCH_LIST
SOURCE_ARCHIVE	=>	!COMMANDS.ARCHIVE
SWITCH	=>	!COMMANDS.SWITCHES
VIEW	=>	!COMMANDS.CMVC
WO	=>	!COMMANDS.WORK_ORDER

The following main procedures provide abbreviations of commands. Note that parameters not included from the underlying command default to the default values defined for the underlying command. For more information on these abbreviations, examine their bodies in world !Commands.Abbreviations.

```
procedure Aedit (The_Activity : String := "<ACTIVITY>");
-- Activity.Edit
```

```
procedure Alist (Pattern : String := '@'C(ADA)";
Descending : Boolean := False;
Response : String := "<PROFILE>";
Options : String := "");
-- Library.Ada_List
```

```
procedure Code (Unit : String := "<IMAGE>";
Limit : String := "<WORLDS>";
Effort_Only : Boolean := False;
Response : String := "<PROFILE>");
-- Compilation.Make; redirects logging output to a file; checks for
-- errors
```

```
procedure Compare (File_1 : String := "<REGION>";
File_2 : String := "<IMAGE>";
Subsubjects : Boolean := False;
Ignore_Case : Boolean := False;
Options : String := "");
-- File.Utilities.Compare
```

```
procedure Ddef (Location : String := "<SELECTION>";
Stack_Frame : Integer := 0);
-- Debug.Source
```

```
procedure Def (Name : String := "<CURSOR>";
In_Place : Boolean := False;
Visible : Boolean := True);
-- Common.Definition
```

```
procedure Diff (File_1 : String := "<REGION>";
File_2 : String := "<IMAGE>";
Result : String := "";
Compressed_Output : Boolean := False;
Subsubjects : Boolean := False);
-- File.Utilities.Difference
```

```
with System_Backup;
procedure Do_Backup (Variety : System_Backup.Kind := System_Backup.Full;
Starting_At : String := "");
-- System_Backup.Backup; performs various other housekeeping functions
```

```
procedure Find (Pattern : String := "";
File : String := "<IMAGE>";
Wildcards : Boolean := False;
Ignore_Case : Boolean := True;
Result : String := "");
-- File.Utilities.Find
```

```
procedure Full_Backup (Starting_At : String := "");
-- Do_Backup (see above)
```

```
procedure Help (Name : String := "Help_On_Help");
-- What.Does
```

```
procedure Input (Name : String := "<CURSOR>");
-- Io.Set_Input
```

```
procedure Install (Unit : String := "<IMAGE>";
Limit : String := "<WORLDS>";
Effort_Only : Boolean := False;
Response : String := "<PROFILE>");
-- Compilation.Promote; redirects logging output to a file; checks for
-- errors
```

```
procedure Ledit (World : String := "<IMAGE>");
-- Links.Edit
```

```

procedure List (Pattern : String := "@";
  Descending : Boolean := False;
  Response : String := "<PROFILE>";
  Options : String := "");
  -- Library.List

```

```

procedure Need (Unit : String := "<IMAGE>";
  Transitive : Boolean := False;
  Response : String := "<PROFILE>");
  -- Compilation.Dependents

```

```

procedure Output (Name : String := ">>FILE NAME<<");
  -- Log.Set_Output; Log.Set_Error

```

```

procedure Primary_Backup (Starting_At : String := "");
  -- Do_Backup (see above)

```

```

procedure Print (Name : String := "<IMAGE>";
  Options : String := "<DEFAULT>";
  Banner : String := "<DEFAULT>";
  Header : String := "<DEFAULT>";
  Footer : String := "<DEFAULT>");
  -- Queue.Print

```

```

procedure Run_Job (S : String := "<SELECTION>";
  Debug : Boolean := False;
  Context : String := "#";
  After : Duration := 0.0;
  Options : String := "";
  Response : String := "<PROFILE>");
  -- Program.Run_Job

```

```

procedure Schedule_Shutdown (At_Time : String := "23:59";
  Reason : String := "COPS";
  Explanation : String := "Cause not entered");
  -- Operator.Shutdown; also sends warning message

```

```

procedure Secondary_Backup (Starting_At : String := "");
  -- Do_Backup (see above)

```

```

procedure Sedit (Switch_File : String := "<SWITCH>");
  -- Switches.Edit

```

```

procedure Send (Who : String := ">>USER or all<<";
  Contents : String := "");
  -- Message.Send

```

```

procedure Sledit (Session : String := ""; User : String := "");
  -- Search_List.Show_List (allows editing)

```

```

procedure Sedit (Switch_File : String := "<SESSION>");
  -- Switches.Edit; by default edits session switches

```

```

procedure Users (Jobs_Too : Boolean := False; Verbose : Boolean := False);
  -- Similar to What.Users, but displays information only for
  -- the current user, and optionally the user's jobs

```

```

procedure Vlist (Pattern : String := "@";
  Descending : Boolean := False;
  Response : String := "<PROFILE>";
  Options : String := "");
  -- Library.Verbose_List

```

!Model.R1000 Links

```

LINK LINK SOURCE
EXT: ACCESS_LIST => !COMMANDS.ACCESS_LIST
EXT: ACCESS_LIST_TOOLS => !TOOLS.ACCESS_LIST_TOOLS
EXT: ACTION_UTILITIES => !COMMANDS.ACTION_UTILITIES
EXT: ACTIVITY => !COMMANDS.ACTIVITY
EXT: ADA => !COMMANDS.ADA
EXT: ADA_OBJECT_EDITOR => !TOOLS.ADA_OBJECT_EDITOR
EXT: ADA_TEXT => !TOOLS.ADA_TEXT
EXT: ALLOWS_DEALLOCATION => !TOOLS.ALLOWS_DEALLOCATION
EXT: ARCHIVE => !COMMANDS.ARCHIVE
EXT: BOUNDED_STRING => !TOOLS.BOUNDED_STRING
EXT: BYTE_STRING_IO => !TOOLS.NETWORKING.BYTE_STRING_IO
EXT: CALENDAR => !LRM.CALENDAR
EXT: CMCV => !COMMANDS.CMCV
EXT: CMCV_MAINTENANCE => !COMMANDS.CMCV_MAINTENANCE
EXT: COMMAND => !COMMANDS.COMMAND
EXT: COMMON => !COMMANDS.COMMON
EXT: COMPILATION => !COMMANDS.COMPIATION
EXT: CONCURRENT_MAP_GENERIC => !TOOLS.CONCURRENT_MAP_GENERIC
EXT: DAEMON => !COMMANDS.DAEMON
EXT: DEBUG => !COMMANDS.DEBUG
EXT: DEBUG_TOOLS => !TOOLS.DEBUG_TOOLS
EXT: DEVICE_INDEPENDENT_IO => !IO.DEVICE_INDEPENDENT_IO
EXT: DIANA_OBJECT_EDITOR => !TOOLS.DIANA_OBJECT_EDITOR
EXT: DIANA_TREE => !COMMANDS.DIANA_TREE
EXT: DIRECTORY_TOOLS => !TOOLS.DIRECTORY_TOOLS
EXT: DIRECT_IO => !IO.DIRECT_IO
EXT: DISK_DAEMON => !TOOLS.DISK_DAEMON
EXT: DISK_SPACE => !COMMANDS.DISK_SPACE
EXT: EDITOR => !COMMANDS.EDITOR
EXT: FILE_TRANSFER => !TOOLS.NETWORKING.FILE_TRANSFER
EXT: FILE_UTILITIES => !COMMANDS.FILE_UTILITIES
EXT: FTP => !COMMANDS.FTP
EXT: FTP_DEFS => !TOOLS.NETWORKING.FTP_DEFS
EXT: FTP_NAME_MAP => !TOOLS.NETWORKING.FTP_NAME_MAP
EXT: FTP_PROFILE => !TOOLS.NETWORKING.FTP_PROFILE
EXT: HASH => !TOOLS.HASH
EXT: HOST_ID_IO => !TOOLS.NETWORKING.HOST_ID_IO
EXT: INCREMENTAL => !COMMANDS.INCREMENTAL
EXT: INTERCHANGE => !TOOLS.NETWORKING.INTERCHANGE
EXT: INTERCHANGE_DEFS => !TOOLS.NETWORKING.INTERCHANGE_DEFS
EXT: IO => !IO.IO
EXT: IO_EXCEPTIONS => !IO.IO_EXCEPTIONS
EXT: JOB => !COMMANDS.JOB
EXT: LIBRARY => !COMMANDS.LIBRARY
EXT: LIBRARY_OBJECT_EDITOR => !TOOLS.LIBRARY_OBJECT_EDITOR
EXT: LINKS => !COMMANDS.LINKS
EXT: LINK_TOOLS => !TOOLS.LINK_TOOLS
EXT: LIST_GENERIC => !TOOLS.LIST_GENERIC
EXT: LOG => !COMMANDS.LOG
EXT: MACHINE_CODE => !LRM.MACHINE_CODE
EXT: MAP_GENERIC => !TOOLS.MAP_GENERIC
EXT: MESSAGE => !COMMANDS.MESSAGE
EXT: OBJECT_EDITOR => !TOOLS.OBJECT_EDITOR
EXT: OBJECT_SET => !IO.OBJECT_SET
EXT: OPERATOR => !COMMANDS.OPERATOR

```

!Model.R1000 Links

```

EXT: PARAMETER_PARSER => !TOOLS.PARAMETER_PARSER
EXT: PIPE => !IO.PIPE
EXT: POLYMORPHIC_IO => !IO.POLYMORPHIC_IO
EXT: POLYMORPHIC_SEQUENTIAL_IO => !IO.POLYMORPHIC_SEQUENTIAL_IO
EXT: PROFILE => !TOOLS.PROFILE
EXT: PROGRAM => !COMMANDS.PROGRAM
EXT: QUEUE => !COMMANDS.QUEUE
EXT: QUEUE_GENERIC => !TOOLS.QUEUE_GENERIC
EXT: RANDOM => !TOOLS.RANDOM
EXT: RPC => !TOOLS.NETWORKING.RPC
EXT: RPC_CLIENT => !TOOLS.NETWORKING.RPC_CLIENT
EXT: RPC_SERVER => !TOOLS.NETWORKING.RPC_SERVER
EXT: SCHEDULER => !COMMANDS.SCHEDULER
EXT: SCRIPT => !TOOLS.SCRIPT
EXT: SEARCH_LIST => !COMMANDS.SEARCH_LIST
EXT: SEQUENTIAL_IO => !IO.SEQUENTIAL_IO
EXT: SET_GENERIC => !TOOLS.SET_GENERIC
EXT: SIMPLE_STATUS => !TOOLS.SIMPLE_STATUS
EXT: STACK_GENERIC => !TOOLS.STACK_GENERIC
EXT: STRING_MAP_GENERIC => !TOOLS.STRING_MAP_GENERIC
EXT: STRING_TABLE => !TOOLS.STRING_TABLE
EXT: STRING_UTILITIES => !TOOLS.STRING_UTILITIES
EXT: SWITCHES => !COMMANDS.SWITCHES
EXT: SYSTEM => !LRM.SYSTEM
EXT: SYSTEM_BACKUP => !COMMANDS.SYSTEM_BACKUP
EXT: SYSTEM_UTILITIES => !TOOLS.SYSTEM_UTILITIES
EXT: TABLE_FORMATTER => !TOOLS.TABLE_FORMATTER
EXT: TABLE_SORT_GENERIC => !TOOLS.TABLE_SORT_GENERIC
EXT: TAPE => !COMMANDS.TAPE
EXT: TAPE_SPECIFIC => !IO.TAPE_SPECIFIC
EXT: TAPE_TOOLS => !TOOLS.TAPE_TOOLS
EXT: TCP_IP_BOOT => !TOOLS.NETWORKING.TCP_IP_BOOT
EXT: TCP_IP_DUMP => !TOOLS.NETWORKING.TCP_IP_DUMP
EXT: TERMINAL => !COMMANDS.TERMINAL
EXT: TERMINAL_SPECIFIC => !IO.TERMINAL_SPECIFIC
EXT: TEXT => !COMMANDS.TEXT
EXT: TEXT_IO => !IO.TEXT_IO
EXT: TIME_UTILITIES => !TOOLS.TIME_UTILITIES
EXT: TRANSFER_GENERIC => !TOOLS.NETWORKING.TRANSFER_GENERIC
EXT: TRANSPORT => !TOOLS.NETWORKING.TRANSPORT
EXT: TRANSPORT_DEFS => !TOOLS.NETWORKING.TRANSPORT_DEFS
EXT: TRANSPORT_INTERCHANGE => !TOOLS.NETWORKING.TRANSPORT_INTERCHANGE
EXT: TRANSPORT_NAME => !TOOLS.NETWORKING.TRANSPORT_NAME
EXT: TRANSPORT_SERVER => !TOOLS.NETWORKING.TRANSPORT_SERVER
EXT: TRANSPORT_SERVER_PROC => !TOOLS.NETWORKING.TRANSPORT_SERVER_PROC
EXT: TRANSPORT_STREAM => !TOOLS.NETWORKING.TRANSPORT_STREAM
EXT: UNBOUNDED_STRING => !TOOLS.UNBOUNDED_STRING
EXT: UNCHECKED_CONVERSION => !LRM.UNCHECKED_CONVERSION
EXT: UNCHECKED_CONVERSIONS => !TOOLS.UNCHECKED_CONVERSIONS
EXT: UNCHECKED_DEALLOCATION => !LRM.UNCHECKED_DEALLOCATION
EXT: WHAT => !COMMANDS.WHAT
EXT: WINDOW_IO => !IO.WINDOW_IO
EXT: WORK_ORDER => !COMMANDS.WORK_ORDER

```

M-1

!Model.R1000\_Implementation Links

LINK	SOURCE
EXT: ACCESS_LIST	=> !COMMANDS.ACCESS_LIST
EXT: ACCESS_LIST_TOOLS	=> !TOOLS.ACCESS_LIST_TOOLS
EXT: ACTION	=> !IMPLEMENTATION.ACTION
EXT: ACTION_UTILITIES	=> !COMMANDS.ACTION_UTILITIES
EXT: ACTIVITY	=> !COMMANDS.ACTIVITY
EXT: ACTIVITY_IMPLEMENTATION	=> !IMPLEMENTATION.ACTIVITY_IMPLEMENTATION
EXT: ADA	=> !COMMANDS.ADA
EXT: ADA_OBJECT_EDITOR	=> !TOOLS.ADA_OBJECT_EDITOR
EXT: ADA_TEXT	=> !TOOLS.ADA_TEXT
EXT: ALLOWS_DEALLOCATION	=> !TOOLS.ALLOWS_DEALLOCATION
EXT: ARCHIVE	=> !COMMANDS.ARCHIVE
EXT: BOUNDED_STRING	=> !TOOLS.BOUNDED_STRING
EXT: BYTE_STRING_IO	=> !TOOLS.NETWORKING.BYTE_STRING_IO
EXT: CALENDAR	=> !LM.MACHINE
EXT: CMVC	=> !COMMANDS.CMVC
EXT: CMVC_IMPLEMENTATION	=> !IMPLEMENTATION.CMVC_IMPLEMENTATION
EXT: CMVC_IMPLEMENTATION_ERRORS	=> !IMPLEMENTATION.CMVC_IMPLEMENTATION_ER
EXT: CMVC_IMPLEMENTATION_UTILITIES	=> !IMPLEMENTATION.CMVC_IMPLEMENTATION_UT
EXT: CMVC_MAINTENANCE	=> !COMMANDS.CMVC_MAINTENANCE
EXT: COMMON	=> !COMMANDS.COMMON
EXT: COMMON	=> !COMMANDS.COMMON
EXT: COMPILATION	=> !COMMANDS.COMPIATION
EXT: CONCURRENT_MAP_GENERIC	=> !TOOLS.CONCURRENT_MAP_GENERIC
EXT: DAEMON	=> !COMMANDS.DAEMON
EXT: DEBUG	=> !COMMANDS.DEBUG
EXT: DEBUG_TOOLS	=> !TOOLS.DEBUG_TOOLS
EXT: DEFAULT	=> !IMPLEMENTATION.DEFAULT
EXT: DEPENDENCY_DATA_BASE	=> !IMPLEMENTATION.DEPENDENCY_DATA_BASE
EXT: DEVICE_INDEPENDENT_IO	=> !IO.DEVICE_INDEPENDENT_IO
EXT: DIANA	=> !IMPLEMENTATION.DIANA
EXT: DIANA_OBJECT_EDITOR	=> !TOOLS.DIANA_OBJECT_EDITOR
EXT: DIANA_RENAMERS	=> !IMPLEMENTATION.DIANA_RENAMERS
EXT: DIANA_TREE	=> !COMMANDS.DIANA_TREE
EXT: DIRECTORY	=> !IMPLEMENTATION.DIRECTORY
EXT: DIRECTORY_TOOLS	=> !TOOLS.DIRECTORY_TOOLS
EXT: DIRECT_IO	=> !IO.DIRECT_IO
EXT: DISK_DAEMON	=> !TOOLS.DISK_DAEMON
EXT: DISK_SPACE	=> !COMMANDS.DISK_SPACE
EXT: EDITOR	=> !COMMANDS.EDITOR
EXT: ERROR_MESSAGES	=> !IMPLEMENTATION.ERROR_MESSAGES
EXT: ERROR_REPORTING	=> !IMPLEMENTATION.ERROR_REPORTING
EXT: FILE_TRANSFER	=> !TOOLS.NETWORKING.FILE_TRANSFER
EXT: FILE_UTILITIES	=> !COMMANDS.FILE_UTILITIES
EXT: FTP	=> !COMMANDS.FTP
EXT: FTP_DEFS	=> !TOOLS.NETWORKING.FTP_DEFS
EXT: FTP_NAME_MAP	=> !TOOLS.NETWORKING.FTP_NAME_MAP
EXT: FTP_PROFILE	=> !TOOLS.NETWORKING.FTP_PROFILE
EXT: HASH	=> !TOOLS.HASH
EXT: HOST_ID_IO	=> !TOOLS.NETWORKING.HOST_ID_IO
EXT: INCREMENTAL	=> !COMMANDS.INCREMENTAL
EXT: INTERCHANGE	=> !TOOLS.NETWORKING.INTERCHANGE
EXT: INTERCHANGE_DEFS	=> !TOOLS.NETWORKING.INTERCHANGE_DEFS
EXT: IO	=> !IO.IO
EXT: IO_EXCEPTIONS	=> !IO.IO_EXCEPTIONS
EXT: JOB	=> !COMMANDS.JOB
EXT: JOB_SEGMENT	=> !IMPLEMENTATION.JOB_SEGMENT

!Model.R1000\_Implementation Links

EXT: LIBRARY	=> !COMMANDS.LIBRARY
EXT: LIBRARY_OBJECT_EDITOR	=> !TOOLS.LIBRARY_OBJECT_EDITOR
EXT: LIMIT_OPERATIONS	=> !IMPLEMENTATION.LIMIT_OPERATIONS
EXT: LINKS	=> !COMMANDS.LINKS
EXT: LINKS_IMPLEMENTATION	=> !IMPLEMENTATION.LINKS_IMPLEMENTATION
EXT: LINK_TOOLS	=> !TOOLS.LINK_TOOLS
EXT: LIST_GENERIC	=> !TOOLS.LIST_GENERIC
EXT: LOAD_VIEW	=> !IMPLEMENTATION.LOAD_VIEW
EXT: LOG	=> !COMMANDS.LOG
EXT: LOW_LEVEL_ACTION	=> !IMPLEMENTATION.LOW_LEVEL_ACTION
EXT: MACHINE	=> !IMPLEMENTATION.MACHINE
EXT: MACHINE_CODE	=> !LM.MACHINE_CODE
EXT: MAP_GENERIC	=> !TOOLS.MAP_GENERIC
EXT: MESSAGE	=> !COMMANDS.MESSAGE
EXT: OBJECT_EDITOR	=> !TOOLS.OBJECT_EDITOR
EXT: OBJECT_SET	=> !IO.OBJECT_SET
EXT: OPERATOR	=> !COMMANDS.OPERATOR
EXT: PARAMETER_PARSER	=> !TOOLS.PARAMETER_PARSER
EXT: PIPE	=> !IO.PIPE
EXT: POLYMORPHIC_IO	=> !IO.POLYMORPHIC_IO
EXT: POLYMORPHIC_SEQUENTIAL_IO	=> !IO.POLYMORPHIC_SEQUENTIAL_IO
EXT: PRODUCT_AUTHORIZATION	=> !IMPLEMENTATION.PRODUCT_AUTHORIZATION
EXT: PROFILE	=> !TOOLS.PROFILE
EXT: PROGRAM	=> !COMMANDS.PROGRAM
EXT: QUEUE	=> !COMMANDS.QUEUE
EXT: QUEUE_GENERIC	=> !TOOLS.QUEUE_GENERIC
EXT: RANDOM	=> !TOOLS.RANDOM
EXT: RPC	=> !TOOLS.NETWORKING.RPC
EXT: RPC_CLIENT	=> !TOOLS.NETWORKING.RPC_CLIENT
EXT: RPC_SERVER	=> !TOOLS.NETWORKING.RPC_SERVER
EXT: SCHEDULER	=> !COMMANDS.SCHEDULER
EXT: SCRIPT	=> !TOOLS.SCRIPT
EXT: SEARCH_LIST	=> !COMMANDS.SEARCH_LIST
EXT: SEMANTIC_ATTRIBUTES	=> !IMPLEMENTATION.SEMANTIC_ATTRIBUTES
EXT: SEQUENTIAL_IO	=> !IO.SEQUENTIAL_IO
EXT: SET_GENERIC	=> !TOOLS.SET_GENERIC
EXT: SIMPLE_STATUS	=> !TOOLS.SIMPLE_STATUS
EXT: SNAPSHOT_NOTIFICATION	=> !IMPLEMENTATION.SNAPSHOT_NOTIFICATION
EXT: STACK_GENERIC	=> !TOOLS.STACK_GENERIC
EXT: STRING_MAP_GENERIC	=> !TOOLS.STRING_MAP_GENERIC
EXT: STRING_TABLE	=> !TOOLS.STRING_TABLE
EXT: STRING_UTILITIES	=> !TOOLS.STRING_UTILITIES
EXT: SWITCHES	=> !COMMANDS.SWITCHES
EXT: SWITCH_IMPLEMENTATION	=> !IMPLEMENTATION.SWITCH_IMPLEMENTATION
EXT: SYSTEM	=> !LM.SYSTEM
EXT: SYSTEM_BACKUP	=> !COMMANDS.SYSTEM_BACKUP
EXT: SYSTEM_UTILITIES	=> !TOOLS.SYSTEM_UTILITIES
EXT: TABLE_FORMATTER	=> !TOOLS.TABLE_FORMATTER
EXT: TABLE_SORT_GENERIC	=> !TOOLS.TABLE_SORT_GENERIC
EXT: TAPE	=> !COMMANDS.TAPE
EXT: TAPE_SPECIFIC	=> !IO.TAPE_SPECIFIC
EXT: TAPE_TOOLS	=> !TOOLS.TAPE_TOOLS
EXT: TCP_IP_BOOT	=> !TOOLS.NETWORKING.TCP_IP_BOOT
EXT: TCP_IP_DUMP	=> !TOOLS.NETWORKING.TCP_IP_DUMP
EXT: TERMINAL	=> !COMMANDS.TERMINAL
EXT: TERMINAL_SPECIFIC	=> !IO.TERMINAL_SPECIFIC
EXT: TEXT	=> !COMMANDS.TEXT
EXT: TEXT_IO	=> !IO.TEXT_IO
EXT: TIME_UTILITIES	=> !TOOLS.TIME_UTILITIES
EXT: TRANSFER_GENERIC	=> !TOOLS.NETWORKING.TRANSFER_GENERIC



```

EXT: TRANSPORT
EXT: TRANSPORT_DEFS
EXT: TRANSPORT_INTERCHANGE
EXT: TRANSPORT_NAME
EXT: TRANSPORT_SERVER
EXT: TRANSPORT_SERVER_PROC
EXT: TRANSPORT_STREAM
EXT: UNBOUNDED_STRING
EXT: UNCHECKED_CONVERSION
EXT: UNCHECKED_CONVERSIONS
EXT: UNCHECKED_DEALLOCATION
EXT: UNIVERSAL
EXT: WHAT
EXT: WINDOW_IO
EXT: WORK_ORDER
EXT: WORK_ORDER_ERRORS
EXT: WORK_ORDER_IMPLEMENTATION

=> !TOOLS.NETWORKING.TRANSPORT
=> !TOOLS.NETWORKING.TRANSPORT_DEFS
=> !TOOLS.NETWORKING.TRANSPORT_INTERCHANGE
=> !TOOLS.NETWORKING.TRANSPORT_NAME
=> !TOOLS.NETWORKING.TRANSPORT_SERVER
=> !TOOLS.NETWORKING.TRANSPORT_SERVER_PROC
=> !TOOLS.NETWORKING.TRANSPORT_STREAM
=> !TOOLS.NETWORKING.UNBOUNDED_STRING
=> !IRM.UNCHECKED_CONVERSION
=> !IRM.UNCHECKED_CONVERSIONS
=> !IRM.UNCHECKED_DEALLOCATION
=> !IMPLEMENTATION.UNIVERSAL
=> !COMMANDS.WHAT
=> !IO.WINDOW_IO
=> !COMMANDS.WORK_ORDER
=> !IMPLEMENTATION.WORK_ORDER_ERRORS
=> !IMPLEMENTATION.WORK_ORDER_IMPLEMENTATION

```

```

!Model.R1000_Portable Links

LINK SOURCE
=>
KIND LINK => SOURCE
EXT: CALENDAR => !IRM.CALENDAR
EXT: DIRECT_IO => !IO.DIRECT_IO
EXT: IO_EXCEPTIONS => !IO.IO_EXCEPTIONS
EXT: SEQUENTIAL_IO => !IO.SEQUENTIAL_IO
EXT: SYSTEM => !IRM.SYSTEM
EXT: TEXT_IO => !IO.TEXT_IO
EXT: UNCHECKED_CONVERSION => !IRM.UNCHECKED_CONVERSION
EXT: UNCHECKED_DEALLOCATION => !IRM.UNCHECKED_DEALLOCATION

```



This section intentionally left blank.

Please use this section as a repository for quick-reference information describing locally-developed tools you use.



This section intentionally left blank.

Please use this section as a repository for release notes distributed by Rational.



## Naming in the Rational Environment Library System

Two distinct forms of names are used in the library system: string names and Ada names. (A subset of the string naming mechanism is used only during debugging.) Every directory object has a string name; only Ada units have Ada names. String names must be enclosed in quotation marks.

String names include special names, wildcards, substitution characters, special characters, and attributes. String names assume a closed scope by default.

The following sections describe string names and Ada names.

### STRING NAMES

#### Special Names

Special names are used as parameter values for many Environment operations to specify text, objects, and regions. Special names allow you to designate without providing a pathname. Listed below are the special names used in the Environment and their references:

- "<SELECTION>" References the highlighted object, if the cursor is in the highlighted area. Otherwise, an error will result.
- "<REGION>" References the highlighted object.
- "<CURSOR>" References the highlighted object, if the cursor is in the highlighted area. If the cursor is not located in the highlighted area, this special name refers to the object associated with the smallest selectable region surrounding the cursor.
- "<IMAGE>" References the highlighted object, if the cursor is in the highlighted area. If the cursor is not located in the highlighted area, this special name references the object corresponding to the image in which the cursor is located.
- "<TEXT>" References the object named in the highlighted text in the image in the window. This is equivalent to typing the contents of the highlighted area as the name.
- "<ACTIVITY>" References the default activity. If an activity is highlighted and the cursor is in the highlight, this special name references that activity rather than the default activity.

### Wildcards

Wildcards allow for both the abbreviation of names and the specifying of several objects with one name. The wildcards are: pound sign (#), at sign (@), question mark (?), and double question mark (??).

#### The Wildcard #

The pound sign (#) matches any single identifier character in a name, including the underscore (\_). It can be used several times within a single name. For example, F## will match the name Food.

#### The Wildcard @

The at sign (@) matches zero or more identifier characters in a name, including the underscore (\_). It does not match a period (.). The wildcard can be used several times within a single name. For example, the name !U@.@.Food matches the name !Users.Fred.Food.

#### The Wildcard ?

The question mark (?) matches zero or more components in a name that are not worlds or objects contained by those worlds. For example, consider the world below:

```
!Users.Stooges : Library (World);
Curly : I Ada (Proc_Spec);
Curly : I Ada (Proc_Body);
.Foo : I Ada (Proc_Body);
Larry : I Ada (Proc_Spec);
Larry : I Ada (Proc_Body);
.Bar : I Ada (Proc_Body);
Moe : C Ada (Proc_Spec);
Moe : C Ada (Proc_Body);
```

The name !Users.Stooges? matches !Users.Stooges itself and units within that world called Larry, Curly, Moe, and any of their subunits. It would not match any worlds within !Users.Stooges.

The periods before and after this wildcard are optional. For example, the name A.?B is equivalent to the name A?B.

#### The Wildcard ??

The double question mark (??) matches zero or more components in a name, including worlds or objects contained by those worlds. For example, the name !Users?? matches the home worlds of all users and the contents of those worlds. !Users.Bill?? matches everything in Bill's home world, including worlds and the objects within those worlds, and !Users.Bill itself. As another example, consider that !?? matches all objects in the directory system on a given machine.

The periods before and after this wildcard are optional. For example, the name A.??B is equivalent to the name A??B.

### Substitution Characters

The Environment also offers support for parameterization within a string name designating the target of an operation--for example, the destination of a Library.Copy. The target string parameters are called substitution characters and represent placeholders in the target string for portions of the source name. The substitution characters are pound sign (#), at sign (@), and question mark (?). Matching is performed from right to left, substituting character against wildcard.

The Substitution Character #

The pound sign (#) is replaced by the next complete (right to left) segment in the source name.

The Substitution Character @

The at sign (@) is replaced by the portion of the source name that is matched by a wildcard in the source name. If there is more than one wildcard in the source name, a separate @ character is needed in the target to match each one.

The Substitution Character ?

The question mark (?) is replaced by successive full segments of the source name, working right to left, until the segment for a world is encountered.

Using the example:

```
!Users.Foo
...
IKM      : Session;
Foo.File : File;
Gen.Data : Ada (Gen_Pack);
Packet   : Ada (Pack_Spec);
Formula  : Ada (Proc_Inst);
Sub_Foo  : Library (Directory);
...
```

```
!User.Foo.Sub_Foo
```

```
...
Gen.Data : File;
Actor     : Ada (Func_Spec);
Actor     : Ada (Func_Body);
Steamer   : Ada (Pack_Spec);
...
```

the following uses of substitution characters can be demonstrated:

Source String	Target String	Target Names
Foo.Packet	Bar.#	Bar.Packet
Foo?Gen@	#.Old@_Save	Foo.Old_Data_Save Sub_Foo.Old_Data_Save
Foo?@=	Bar?	Bar.Formula Bar.Sub_Foo.Steamer

### Special Characters

Special characters can be used in names to specify either relative or absolute contexts or to specify indirect files of names. These special characters apply to names used throughout the Environment.

A special character in a name determines the context in which the remaining portion of the name will be interpreted. A special character of exclamation (!), caret (^), dollar sign (\$), double dollar sign (\$\$), percent (%), underscore (\_), period (.), backslash (\), or grave (`) causes explicit interpretations of the remainder of the name, as described below.

Character pairs are also used to enclose a name and to give that name an additional meaning. Character pairs are brackets ([]) and braces ({}), which are also described below.

The Special Character !

The exclamation mark (!) specifies that the context for resolving the remainder of the name should be set to the root of the library system. This creates a fully qualified name. This character represents the root of the library system in any context.

The Special Character ^

The caret (^) specifies that the context should be set to the immediately enclosing object. The caret permits naming to climb the hierarchy of objects and eventually reach the root of the library system. The caret prefix can be used repeatedly to define the context to be several units above the current context. The parent object of the root of the directory system is itself.

A special use of this character occurs in combination with a bracketed name. A name component of the form "[some\_object]" resolves to the closest containing object whose simple name is Some\_Object.

The caret is frequently used as a shorthand method for referring to objects in a parent unit.

The Special Character \$

The dollar sign (\$) specifies that the context should be set to the immediately enclosing library. If the current context is a library, this character has no effect.

A special use of this character occurs in combination with a bracketed name. A name component of the form \$[some\_library] resolves to the closest containing library whose simple name is Some\_Library.

The Special Character @@

The double dollar sign (@@) specifies that the context should be set to the immediately enclosing world. This is more restrictive than the



single dollar sign (\$) , which is either a world or a directory. If the current context is a world, this character has no effect.

A special use of this character occurs in combination with a bracketed name. A name component of the form \$\$[some\_world] resolves to the closest containing world whose simple name is Some\_World.

#### The Special Character %

The percent (%) , used only in the Rational Debugger, can be used only as the first character of a name. It specifies that the next name component is a task name. Task names are either string names assigned to tasks by calls to the !Commands.Debug.Set\_Task\_Name or the !Tools.Debug\_Tools.Set\_Task\_Name procedure or task numbers assigned by the Environment. The !Commands.Debug.Task\_Display procedure lists all tasks and their names and numbers.

The components of a name that follow the task name are interpreted as objects declared in the named task. If the task name is followed by \_n (where n is a number), the name refers to a stack frame of the named task. Stack frame names are further discussed in "The Special Character ~" below.

#### The Special Character -

The underscore (\_) is interpreted as an indirect file prefix when used in Environment commands. If the first character after the underscore is an alphabetic character, it is assumed to be the first character of the name of a file that contains other names. This provides a way of building lists of objects and referring to that list in a name. The file may be a text or an activity file.

The underscore character is also interpreted as a stack frame prefix when used in the Rational Debugger. If the value of an object declared in a subprogram is to be named, the frame on the run-time stack that contains an activation of that subprogram must be named. Stack frames are numbered for each task, starting at the top with 1.

#### The Special Character .

The period (.) is used both as a name component separator and as a name prefix. As a separator, it is used just as in Ada names to separate components of a name.

As a prefix character, the period specifies that the first component of the name is a library unit name. This is used only in the Rational Debugger. A second component of the name would be an object declared in the named library unit.

#### The Special Character \

The backslash (\) specifies that the next name component be evaluated in the current searchlist.

#### The Special Character `

The grave (`) is used to evaluate names using the current context and the set of links associated with the current context. The grave evaluates the name as if it were the name of an Ada unit in a "with" clause of a unit in the library that contains the current context. For example, the name `Moe resolves to an Ada unit called Moe in the containing library. Moe could be a link to some other library.

#### The Special Characters []

Brackets ([]) define a set notation. Sets are created by enclosing a series of name components, separated by commas, in brackets. The semicolon character in set notation can also be used to separate name components. Commas and semicolons cannot be mixed. If semicolons are used, each name component in the set must resolve to at least one object.

Names can also be excluded from a set with the tilde (~).

The special string [] represents the current context, whether that context is a directory, world, Ada unit, or other object.

#### The Special Characters {}

Braces ({} ) denote objects that have been deleted but not expunged.

#### Attributes

Attributes are special strings that specify a restriction on the evaluation of the name. Syntactically like Ada attributes, these strings are a postfix notation that specifies some restriction on the interpretation of the name. Specific versions of an object, specific classes of objects, either the visible part or the body of an Ada unit, or a nickname can be specified with attributes to remove ambiguity or to specify something other than the default interpretation of the name.

The following is a list of attributes and their meanings:

'body

Matches the body of the specified Ada object.

The 'spec and 'body attributes are mutually exclusive and cover the space of all Ada objects.

'spec

Matches the specification part of a declaration. Single-part declaration objects are also matched by this attribute.

'L(External)link\_name

Matches the named external link in the set of links associated with a world.

'L(Internal)link\_name

Matches the named internal link in the set of links associated with a world.

'L(Any)link\_name

Matches the named internal or external link in the set of links associated with a world.

'N(nickname)

Matches the object whose assigned nickname is the same as the nickname specified.

#### State Attributes

'S(Archived)  
'S(Source)  
'S(Installed)  
'S(Coded)

Matches units in the archived state.  
Matches units in the source state.  
Matches units in the installed state.  
Matches units in the coded state.

#### Version Attributes

'V(ALL)  
'V(ANY)  
'V(MAX)  
'V(MIN)  
'V(n)  
'V(-n)

Matches all versions of the object.  
Matches the default version of the object.  
Matches the newest version of the object.  
Matches the oldest version of the object.  
Matches the version with that version number.  
Matches the nth version preceding the current version.

#### Class Attributes

'C(ADA)  
'C(ARCHIVED\_CODE)  
'C(FILE)  
'C(GROUP)  
'C(LIBRARY)  
'C(NULL\_DEVICE)  
'C(PIPE)  
'C(SESSION)  
'C(TAPE)  
'C(TERMINAL)  
'C(USER)

Matches any Ada program unit.  
Matches objects appearing in a subsystem view for a code-only unit.  
Matches any file.  
Matches any group in the system.  
Matches any directory, world, or subsystem.  
Matches a device that accepts output and discards it.  
Matches any pipe.  
Matches any user's session object.  
Matches any tape drive in the system.  
Matches any terminal in the system.  
Matches any user in the system.

#### Subclass Attributes of Library Class Objects

'C(Comb\_Ss)  
'C(Comb\_View)  
'C(Directory)  
'C(Load\_View)  
'C(Mailbox)  
'C(Spec\_Load)  
'C(Spec\_View)  
'C(Subsystem)  
'C(World)

Matches any subsystem containing combined view that cannot contain spec or load views.  
Matches any combined view of a subsystem.  
Matches any directory.  
Matches any load view of a subsystem.  
Matches any library containing Mail and Mail\_Db files for the Rational Mail Utility.  
Matches any subsystem that cannot contain combined views.  
Matches any spec view of a subsystem.  
Matches any subsystem.  
Matches any world.

#### Subclass Attributes of Ada Class Objects

'C(Alt\_List)  
'C(Comp\_Unit)  
'C(Context)  
'C(Decl\_List)  
'C(Func\_Body)

Matches any alternative list insertion point.  
Matches any compilation unit that has not been semanticized.  
Matches any context clause insertion point.  
Matches any declaration list insertion point.  
Matches any function body.

'C(Func\_Inst)  
'C(Func\_Ren)  
'C(Func\_Spec)  
'C(Gen\_Func)  
'C(Gen\_Pack)  
'C(Gen\_Param)  
'C(Gen\_Proc)  
'C(Insertion)  
'C(Load\_Func)  
'C(Load\_Proc)  
'C(Main\_Body)  
'C(Main\_Func)  
'C(Main\_Proc)  
'C(Pack\_Body)  
'C(Pack\_Inst)  
'C(Pack\_Ren)  
'C(Pack\_Spec)  
'C(Ppragma)  
'C(Proc\_Body)  
'C(Proc\_Inst)  
'C(Proc\_Ren)  
'C(Proc\_Spec)  
'C(Statement)  
'C(Subp\_Body)  
'C(Subp\_Inst)  
'C(Subp\_Ren)  
'C(Subp\_Spec)  
'C(Task\_Body)

Matches any generic function instantiation.  
Matches any function rename.  
Matches any function specification.  
Matches any generic function.  
Matches any generic package.  
Matches any generic parameter insertion point.  
Matches any generic procedure.  
Matches any insertion point.  
Matches any code-only function.  
Matches any code-only procedure.  
Matches any main function body.  
Matches any main procedure body.  
Matches any main function specification.  
Matches any main procedure specification.  
Matches any package body.  
Matches any generic package instantiation.  
Matches any package rename.  
Matches any package specification.  
Matches any pragma insertion point.  
Matches any procedure body.  
Matches any generic procedure instantiation.  
Matches any procedure rename.  
Matches any procedure spec.  
Matches any statement insertion point.  
Matches any subprogram body.  
Matches any generic subprogram instantiation.  
Matches any subprogram rename.  
Matches any subprogram specification.  
Matches any task body.

#### Subclass Attributes of File Class Objects

'C(Activity)  
'C(Binary)  
'C(Cavc\_Db)  
'C(Code\_Db)  
'C(Compat\_Db)  
'C(Config)  
'C(Dictionary)  
'C(Documents)  
'C(File\_Map)  
'C(Log)  
'C(Mail)  
'C(Mail\_Db)  
'C(Msg\_In)  
'C(Msg\_Out)  
'C(Objects)  
'C(Ps)  
'C(Search)  
'C(Switch)  
'C(Switch\_Def)  
'C(Text)  
'C(Venture)  
'C(Work)  
'C(Work\_List)

Matches any activity file.  
Matches any binary file.  
Matches any CMC database.  
Matches any code saved for a subsystem load view.  
Matches any compatibility database for a subsystem.  
Matches any configuration pointer for CMC.  
For future development.  
Matches any document database.  
Matches any file map.  
Matches any log file.  
Matches any collection of messages.  
Matches any user's C mailbox.  
For future development.  
Matches any object set.  
Matches any PostScript file.  
Matches any searchlist file.  
Matches any switch file.  
Matches any switch definition file.  
Matches any text file.  
Matches any collection of work orders for CMC.  
Matches any work order for CMC.  
Matches any work order list for CMC.

A list of subclasses can also be found in Implementation.Object\_Subclass.

#### ADA NAMES

Ada names are implemented as defined by the LRM. Link names can be used to reference library units across world boundaries. Worlds, directories, and files cannot be referenced by Ada names.

In locations other than context clauses, Ada names are always resolved in the context of a library unit and its associated "with" clauses. The simple names in "with" clauses are resolved in a two-step process. First, the immediately containing directory or world is searched for a unit of the same name. If that search fails, the link pac of the closest containing world is searched for the name. If that search fails, the name is undefined.

#### Search Characters

Packages File.Utilities and Editor.Search both contain procedures that allow regular expression pattern matching. The patterns may include the use of the special wildcard characters below. The special interpretation of these characters is enabled by a parameter in packages File.Utilities and Editor.Search procedures. When the special characters are not enabled, they are interpreted literally, just like other characters.

The special wildcard characters for regular expression pattern matching are described below:

- ? Matches any single character.
- % Matches any single character that is legal in an Ada identifier.
- \$ Matches the following characters, which are frequently used as Ada delimiters:  
@'() \*+ , - . / : ; < = > |
- \ When not at the end of the pattern, causes the character immediately following this wildcard to be interpreted as a normal (not a wildcard) character.
- { When at the beginning of the pattern, requires the pattern to match the beginning of the line.
- } When at the end of the pattern, requires the pattern to match the end of the line.
- [] Used around a string of characters, matches any one of the enclosed characters. Each character to be compared must be specified explicitly or by a range (for example, [A-Z]).
- Matches anything except the character (or characters, if inside brackets ( [] )) following this wildcard. If used inside brackets ( [] ), this wildcard must be the first character in the list.
- \* Matches zero or more occurrences of the previous characters or set of characters.

#### Example 1

The command File.Utilities.Find("section 3", "Test"); prints the line(s) containing the string "section 3" in the specified file.

#### Example 2

The command File.Utilities.Find("y\*", "Test", True); prints the line(s) containing a string of any length that is a legal Ada identifier.

#### Example 3

The command File.Utilities.Find("[^abc]", "Test", True); prints the line(s) that start with characters other than 'a', 'b', or 'c' in the specified file.

### Banner Symbols

The Editor maintains the banner of each window it displays. On the left end of the banner is a symbol. This single character represents the state of the image in the window. The following table provides the typical interpretation of these symbols:

Symbol	Lock	Committed?	Formatted?
=	Read	N.A.	True
(Blank)	Write	True	Can be either
*	Write	False	False
#	Write	False	True
!	(The Editor or a job has temporarily locked the image.)		

Note that, for I/O windows, the \* and # symbols mean that a job that is requesting input is running. Because such images are considered uncommitted, you cannot log off without terminating these jobs or ignoring changed images.

### Library Switch Definitions and Default Values

PROCESSOR	SWITCH	TYPE	VALUE
Ftp	Account	: String	:= ""
Format	Alignment_Threshold	: Line_Range	:= 0
R1000_Cg	Asm_Listing	: Boolean	:= False
Cross_Cg	Asm_Source	: Boolean	:= False
Cross_Cg	Auto_Assemble	: Boolean	:= True
Cross_Cg	Auto_Link	: Boolean	:= True
Ftp	Auto_Login	: Boolean	:= False
Semantics	Closed_Private_Part	: Boolean	:= False
Format	Comment_Column	: Line_Range	:= 1
Parser	Configuration	: Switch_Set	:= ()
Format	Consistent_Breaking	: Integer	:= 1
Directory	Create_Internal_Links	: Boolean	:= True
Directory	Create_Subprogram_Specs	: Boolean	:= True
Cross_Cg	Debugging_Level	: Debug_Level	:= None
R1000_Cg	Elab_Order_Listing	: Boolean	:= False
R1000_Cg	Enable_Deallocation	: Boolean	:= False
Cross_Cg	Flags	: String	:= ""
Format	Id_Case	: Letter_Case	:= Capitalized
Semantics	Ignore_Interface_Fragmas	: Boolean	:= False
Semantics	Ignore_Minor_Errors	: Boolean	:= False
Semantics	Ignore_Unsupported_Rep_Specs	: Boolean	:= False
Format	Keyword_Case	: Letter_Case	:= Lower
Format	Line_Length	: Line_Range	:= 80
Cross_Cg	Linker_Command_File	: String	:= ""
Cross_Cg	Listing	: Boolean	:= False
Cross_Cg	Magic	: String	:= ""
Format	Major_Indentation	: Indent_Range	:= 4
Format	Minor_Indentation	: Indent_Range	:= 4
Format	Number_Case	: Letter_Case	:= Upper
Cross_Cg	Object_Libraries	: String	:= ""
Cross_Cg	Optimization_Level	: Integer	:= 0
R1000_Cg	Page_Limit	: Integer	:= 0
Ftp	Password	: String	:= ""
Ftp	Prompt_For_Account	: Boolean	:= False
Ftp	Prompt_For_Password	: Boolean	:= False
Ftp	Remote_Directory	: String	:= ""
Ftp	Remote_Machine	: String	:= ""
Ftp	Remote_Roof	: String	:= ""
Ftp	Remote_Type	: String	:= ""
Directory	Require_Internal_Links	: Boolean	:= True
R1000_Cg	Seg_Listing	: Boolean	:= False
Ftp	Send_Port_Enabled	: Boolean	:= True
Format	Statement_Indentation	: Indent_Range	:= 3
Format	Statement_Length	: Line_Range	:= 35
Semantics	Subsystem_Interface	: Boolean	:= False
Cross_Cg	Suppress_All_Checks	: Boolean	:= False
R1000_Cg	Terminal_Echo	: Boolean	:= False
Ftp	Transfer_Mode	: Mode_Code	:= Nil
Ftp	Transfer_Structure	: Structure_Code	:= Nil
Ftp	Transfer_Type	: Type_Code	:= Nil
Ftp	Username	: String	:= ""
Format	Wrap_Indentation	: Line_Range	:= 16

Session Switch Definitions and Default Values

PROCESSOR	SWITCH	TYPE	VALUE
Session_Ftp	Account	: String	:: ""
Session_Ftp	Auto_Login	: Boolean	:: False
Queue	Banner	: String	:: "user_id">
Session	Beep_On_Errors	: Boolean	:: True
Session	Beep_On_Interrupt	: Boolean	:: True
Session	Beep_On_Messages	: Boolean	:: False
Session	Cmvc_Break_Long_Lines	: Boolean	:: True
Session	Cmvc_Capitalize	: Boolean	:: True
Session	Cmvc_Comment_Extent	: Integer	:: 4
Session	Cmvc_Configuration_Extent	: Integer	:: 0
Session	Cmvc_Enable_Relocation	: Boolean	:: True
Session	Cmvc_Field_Extent	: Integer	:: 4
Session	Cmvc_Indentation	: Integer	:: 2
Session	Cmvc_Line_Length	: Integer	:: 80
Session	Cmvc_Shorten_Name	: Boolean	:: True
Session	Cmvc_Shorten_Unit_State	: Boolean	:: True
Session	Cmvc_Show_Add_Date	: Boolean	:: True
Session	Cmvc_Show_Add_Time	: Boolean	:: True
Session	Cmvc_Show_All_Default_Lists	: Boolean	:: False
Session	Cmvc_Show_All_Default_Orders	: Boolean	:: False
Session	Cmvc_Show_Deleted_Objects	: Boolean	:: False
Session	Cmvc_Show_Deleted_Versions	: Boolean	:: False
Session	Cmvc_Show_Display_Position	: Boolean	:: False
Session	Cmvc_Show_Edit_Info	: Boolean	:: False
Session	Cmvc_Show_Field_Default	: Boolean	:: True
Session	Cmvc_Show_Field_Max_Index	: Boolean	:: False
Session	Cmvc_Show_Field_Type	: Boolean	:: False
Session	Cmvc_Show_Frozen	: Boolean	:: False
Session	Cmvc_Show_Hidden_Fields	: Boolean	:: False
Session	Cmvc_Show_Retention	: Boolean	:: False
Session	Cmvc_Show_Size	: Boolean	:: True
Session	Cmvc_Show_Unit_State	: Boolean	:: True
Session	Cmvc_Show_Users	: Boolean	:: False
Session	Cmvc_Show_Version_Number	: Boolean	:: False
Session	Cmvc_Uppercase	: Boolean	:: False
Session	Cmvc_Version_Extent	: Integer	:: 0
Session	Cursor_Bottom_Offset	: Integer	:: 33
Session	Cursor_Left_Offset	: Integer	:: 33
Session	Cursor_Right_Offset	: Integer	:: 33
Session	Cursor_Top_Offset	: Integer	:: 33
Session	Cursor_Transpose_Moves	: Boolean	:: False
Session	Debug_Addresses	: Boolean	:: False
Session	Debug_Break_At_Creation	: Boolean	:: True
Session	Debug_Declaration_Display	: Boolean	:: True
Session	Debug_Delete_Temporary_Breaks	: Boolean	:: False
Session	Debug_Display_Count	: Integer	:: 10
Session	Debug_Display_Creation	: Boolean	:: True
Session	Debug_Display_Level	: Integer	:: 3
Session	Debug_Echo_Commands	: Boolean	:: True
Session	Debug_Element_Count	: Integer	:: 25
Session	Debug_First_Element	: Integer	:: 0
Session	Debug_Freeze_Tasks	: Boolean	:: False
Session	Debug_History_Count	: Integer	:: 10
Session	Debug_History_Entries	: Integer	:: 1000

Session	Debug_History_Start	: Integer	:: 10
Session	Debug_Include_Packages	: Boolean	:: False
Session	Debug_Interpreter_Control_Words	: Boolean	:: False
Session	Debug_Kill_Old_Jobs	: Boolean	:: True
Session	Debug_Machine_Level	: Boolean	:: False
Session	Debug_Memory_Count	: Integer	:: 3
Session	Debug_No_History_Timestamps	: Boolean	:: True
Session	Debug_Optimize_Generic_History	: Boolean	:: True
Session	Debug_Persistent_Breakpoints	: Boolean	:: True
Session	Debug_Pointer_Level	: Integer	:: 3
Session	Debug_Put_Locals	: Boolean	:: False
Session	Debug_Qualify_Stack_Names	: Boolean	:: False
Session	Debug_Require_Debug_Off	: Boolean	:: False
Session	Debug_Save_Exceptions	: Boolean	:: False
Session	Debug_Show_Location	: Boolean	:: True
Session	Debug_Stack_Count	: Integer	:: 10
Session	Debug_Stack_Start	: Integer	:: 1
Session	Debug_Timestamps	: Boolean	:: False
Session	Default_Job_Page_Limit	: Integer	:: 8000
Cmvc	Default_Venture	: Integer	:: <>
Telnet	Escape	: String	:: ""
Telnet	Escape_On_Break	: Boolean	:: True
Queue	Footer	: String	:: ""
Queue	Header	: String	:: ""
Session	Image_Fill_Column	: Integer	:: 72
Session	Image_Fill_Extra_Space	: Text	:: "!.?"
Session	Image_Fill_Indent	: Integer	:: -1
Session	Image_Fill_Mode	: Boolean	:: False
Session	Image_Insert_Mode	: Text	:: ""
Session	Image_Tab_Stops	: Boolean	:: True
Session	Job_Context_First	: Text	:: ""
Session	Job_Context_Length	: Integer	:: 3
Session	Job_Name_Length	: Integer	:: 2
Session	Job_Name_Separator	: Text	:: "% % "
Session	Key_Directory	: Text	:: ""
Session	Library_Break_Long_Lines	: Boolean	:: True
Session	Library_Capitalize	: Boolean	:: True
Session	Library_Indentation	: Integer	:: 2
Session	Library_Jazy_Realignment	: Boolean	:: True
Session	Library_Line_Length	: Integer	:: 80
Session	Library_Misc_Show_Edit_Info	: Boolean	:: True
Session	Library_Misc_Show_Frozen	: Boolean	:: True
Session	Library_Misc_Show_Retention	: Boolean	:: True
Session	Library_Misc_Show_Size	: Boolean	:: True
Session	Library_Misc_Show_Subclass	: Boolean	:: False
Session	Library_Misc_Show_Unit_State	: Boolean	:: True
Session	Library_Misc_Show_Volume	: Boolean	:: True
Session	Library_Shorten_Names	: Boolean	:: True
Session	Library_Shorten_Subclass	: Boolean	:: True
Session	Library_Shorten_Unit_State	: Boolean	:: True
Session	Library_Show_Deleted_Objects	: Boolean	:: False
Session	Library_Show_Deleted_Versions	: Boolean	:: False
Session	Library_Show_Miscellaneous	: Boolean	:: False
Session	Library_Show_Standard	: Boolean	:: False
Session	Library_Show_Subunits	: Boolean	:: True
Session	Library_Show_Version_Number	: Boolean	:: False
Session	Library_Std_Show_Class	: Boolean	:: True
Session	Library_Std_Show_Subclass	: Boolean	:: True
Session	Library_Std_Show_Unit_State	: Boolean	:: False

```

Session . Library_Uppercase           := False
Profile . Log_At_Sign_Msgs           := True
Profile . Log_Auxiliary_Msgs        := True
Profile . Log_Diagnostic_Msgs       := False
Profile . Log_Dollar_Msgs           := True
Profile . Log_Error_Msgs             := True
Profile . Log_Exception_Msgs        := True
Profile . Log_File                   := Use_Output
Profile . Log_Line_Width             := 77
Profile . Log_Negative_Msgs          := True
Profile . Log_Note_Msgs              := True
Profile . Log_Position_Msgs         := True
Profile . Log_Positive_Msgs         := True
Profile . Log_Prefix_1              := Yr_Mn_Dy
Profile . Log_Prefix_2              := Hr_Mn_Sc
Profile . Log_Prefix_3              := Symbols
Profile . Log_Sharp_Msgs             := True
Profile . Log_Warning_Msgs           := True
Session . Mail_Multiple_Message_Windows := False
Session . Notify_Warnings           := False
Queue . Options                     := String
Session_Ftp . Password              := "Format=>(Wrap, System_Header)"
Session . Prompt_Delimiters         := String
Session_Ftp . Prompt_For_Account    := ""
Session_Ftp . Prompt_For_Password   := ""
Profile . Reaction                  := Boolean
Session_Ftp . Remote_Directory      := Boolean
Session_Ftp . Remote_Machine        := Boolean
Profile . Remote_Machine             := Boolean
Session_Ftp . Remote_Passwords     := Boolean
Profile . Remote_Root               := Object
Session_Ftp . Remote_Sessions      := Object
Session_Ftp . Remote_Type           := Object
Session . Screen_Dump_File         := Text
Session . Search_Ignore_Case        := Boolean
Session . Search_Preserve_Case     := Boolean
Session . Search_Regular_Expr      := Boolean
Session_Ftp . Send_Port_Enabled     := Boolean
Session . Terminal_Line_Speed      := Integer
Session . Terminal_Padding         := Integer
Session . Text_Bottom_Stripe       := Text
-----
Session . Text_Convert_Tabs        := Boolean
Session . Text_Header              := Boolean
Session . Text_Print_Name          := Boolean
Session . Text_Print_Number        := Boolean
Session . Text_Print_Time          := Boolean
Session . Text_Reuse_Window        := Boolean
Session . Text_Scroll_Output       := Boolean
Session . Text_Top_Stripe          := Text
-----
Session_Ftp . Transfer_Mode        := Nil
Session_Ftp . Transfer_Structure   := Nil
Session_Ftp . Transfer_Type        := Nil
Session_Ftp . Username             := String
Session . Window_Command_Size      := Integer
Session . Window_Frames           := Integer
Session . Window_Frames_Startup    := Integer

```

```

Session . Window_Have_Sides        := Boolean
Session . Window_Is_Staggered     := Boolean
Session . Window_Message_Life     := Integer
Session . Window_Message_Size     := 2
Session . Window_Scroll_Overlap   := Integer
Session . Window_Shift_Overlap    := 20
Session . Word_Breaks              := Text
" " "%%%' () **,-./:;<=>?[]_~{}""

```

This section intentionally left blank.

Please use this section as a repository for specifications of units you use for system programming not included elsewhere in the Reference Summary.





# RATIONAL

## READER'S COMMENTS

**Note:** This form is for documentation comments only. You can also submit problem reports and comments electronically by using the SIMS problem-reporting system. If you use SIMS to submit documentation comments, please indicate the manual name, book name, and page number.

Did you find this book understandable, usable, and well organized? Please comment and list any suggestions for improvement.

---

---

---

---

---

If you found errors in this book, please specify the error and the page number. If you prefer, attach a photocopy with the error marked.

---

---

---

---

Indicate any additions or changes you would like to see in the index.

---

---

---

---

How much experience have you had with the Rational Environment?

6 months or less \_\_\_\_\_ 1 year \_\_\_\_\_ 3 years or more \_\_\_\_\_

How much experience have you had with the Ada programming language?

6 months or less \_\_\_\_\_ 1 year \_\_\_\_\_ 3 years or more \_\_\_\_\_

Name (optional) \_\_\_\_\_ Date \_\_\_\_\_  
Company \_\_\_\_\_  
Address \_\_\_\_\_  
City \_\_\_\_\_ State \_\_\_\_\_ ZIP Code \_\_\_\_\_

Please return this form to:

**Publications Department  
Rational  
1501 Salado Drive  
Mountain View, CA 94043**



**Rational Environment  
Reference Manual**

**Keymap: Facit Terminal**

Copyright © 1985, 1986, 1987 by Rational

Document Control Number: 8001A-51 (803-002324)

Rev. 6.0, November 1985  
Rev. 6.1, March 1986  
Rev. 6.2, July 1986  
Rev. 7.0, July 1987 (Delta)

This document subject to change without notice.

Note the Reader's Comments form on the last page of this book, which requests the user's evaluation to assist Rational in preparing future documentation.

Ada is a registered trademark of the U.S. Government (Ada Joint Program Office).

Rational and R1000 are registered trademarks and Rational Environment and Rational Subsystems are trademarks of Rational.

Rational  
1501 Salado Drive  
Mountain View, California 94043

## Contents

<b>How to Use the Keymap</b> . . . . .	1
Keymap Overview . . . . .	1
Detailed Reference to Key Bindings . . . . .	1
Master Reference to Key Bindings by Command . . . . .	1
Master Reference to Key Bindings by Key . . . . .	2
Environment Key Combinations . . . . .	2
Item-Operation Key Combinations . . . . .	2
Patterns among Item-Operation Combinations . . . . .	2
Modified Key Combinations . . . . .	3
Basic and Accelerated Keystrokes . . . . .	3
Keymap Notation . . . . .	3
Symbols . . . . .	3
Numeric Arguments . . . . .	3
Case Sensitivity of Key Bindings . . . . .	4
<b>Detailed Reference to Key Bindings</b> . . . . .	5
Getting Help and Other Information . . . . .	6
Traversing the Environment . . . . .	6
Logging Off . . . . .	6
Selecting Items . . . . .	7
Executing Commands . . . . .	7
Managing Windows . . . . .	7
Moving between Windows . . . . .	7
Resizing and Repositioning Windows . . . . .	8
Redrawing the Screen . . . . .	8
Retaining Windows . . . . .	8
Removing Windows . . . . .	8
Finding Windows . . . . .	8
Moving within an Image . . . . .	9

By Character . . . . .	9
By Word . . . . .	9
By Underline or Prompt . . . . .	9
By Line . . . . .	9
In a Region . . . . .	10
By Tabs . . . . .	10
By Scrolling . . . . .	10
By Marking Your Place . . . . .	10
General Editing Operations . . . . .	11
Selecting an Arbitrary Region . . . . .	11
Moving and Copying Text . . . . .	11
Deleting Text . . . . .	11
Searching and Replacing Text . . . . .	11
Entering Text . . . . .	12
Transposing Text . . . . .	12
Controlling Case . . . . .	12
Holding and Retrieving Text . . . . .	13
Formatting Text . . . . .	13
Writing Text Files . . . . .	14
Accessing Text Files . . . . .	14
Saving Changes . . . . .	14
Terminating Edit . . . . .	14
Selecting Substructures within Text . . . . .	14
Writing Ada Programs . . . . .	15
Creating Ada Programs . . . . .	15
Accessing Ada Programs . . . . .	15
Saving Changes and Terminating Edit . . . . .	15
Checking for Errors . . . . .	15
Changing the Compilation State . . . . .	16
Changing to a Higher Compilation State . . . . .	16
Changing to a Lower Compilation State . . . . .	16
Selecting Structures within Ada Programs . . . . .	16
Modifying Ada Programs . . . . .	17
Entering Comments and Special Strings . . . . .	17
Browsing Ada Programs . . . . .	17
Checking Using Occurrences . . . . .	17
Debugging Ada Programs . . . . .	18
Stepping and Executing . . . . .	18

Setting and Removing Breakpoints . . . . .	18
Viewing Stacks . . . . .	18
Displaying and Modifying Variables . . . . .	18
Handling Exceptions . . . . .	18
Managing Libraries . . . . .	19
Creating Libraries . . . . .	19
Manipulating Objects in Libraries . . . . .	19
Controlling Library Display . . . . .	19
Using CMVC . . . . .	19
Managing Links . . . . .	20
Accessing Links . . . . .	20
Removing the Link Editor . . . . .	20
Selecting Links . . . . .	20
Modifying Links . . . . .	20
Traversing Linked Ada Units . . . . .	20
Controlling the Display . . . . .	20
Managing Searchlists . . . . .	21
Accessing the Searchlist . . . . .	21
Removing the Searchlist Editor . . . . .	21
Selecting Entries . . . . .	21
Modifying the Searchlist . . . . .	21
Using Keyboard Macros . . . . .	22
Using Environment I/O Resources . . . . .	22
Managing Jobs . . . . .	22
<b>Master Reference to Key Bindings by Command . . . . .</b>	<b>23</b>
<b>Master Reference to Key Bindings by Key . . . . .</b>	<b>29</b>

RATIONAL



## **How to Use the Keymap**

The Rational Environment Keymap is the primary reference guide describing the keys that have been bound to Environment commands. Users have the option of modifying these key bindings for their own use, following procedures described in Rational Environment Basic Operations.

Note that there is a more basic reference to Environment key bindings in the Rational Environment Basic Keymap, in the Rational Environment Basic Operations. It is intended as the primary key reference for new Environment users.

### **Keymap Overview**

The Keymap has been divided into the following three sections. The first and third sections apply to the Facit terminal only. The second section includes key bindings for both the Facit terminal and the Rational Terminal.

#### **Detailed Reference to Key Bindings**

The Detailed Reference provides a nearly complete list of key combinations, organized by topic and subtopic. The Detailed Reference entry for each key combination includes:

- A brief description of what the combination does
- The full name of the command that is bound to it
- Alternative key bindings, including accelerated key combinations (see "Basic and Accelerated Keystrokes," below)

#### **Master Reference to Key Bindings by Command**

This section provides a complete, alphabetic list of the commands that are bound to keys on both the Facit terminal and the Rational Terminal. Each entry includes:

- The full name of an Environment command
- The key combination(s) to which the command is bound on the Facit terminal
- The key combination(s) to which the command is bound on the Rational Terminal

## Master Reference to Key Bindings by Key

This section provides a complete table of the commands that are bound to keys on the Facit terminal. This table is in the form of an Ada procedure with nested case statements describing each of the key combinations. It is the same as the online keymap definition stored in !Machine.Editor\_Data.Facit\_Commands.

## Environment Key Combinations

Environment commands are bound to two types of key combinations:

- Item-operation combinations
- Modified key combinations

These two types of key combinations differ in how they are executed.

## Item-Operation Key Combinations

Each item-operation key combination contains an item key (`Esc`, `Object`, `Region`, `Window`, `Image`, `Line`, `Word`, or `Mark`) followed by an operation key (either alphabetic or nonalphabetic). The item key identifies the item affected by the operation; the operation key identifies the action that applies to the indicated item.

The keystrokes must be sequential in an item-operation key combination. To execute an item-operation key:

1. Press and release the item key.
2. Press and release the operation key.

The notation indicates sequential keystrokes by separating them with a hyphen:

`Item key` - `operation key`.

## Patterns among Item-Operation Combinations

In general, commands that execute similar operations are bound to combinations that contain a common operation key. Some examples include:

`Item` - `C`            Commands that copy items are bound to combinations such as `Line` - `C`, `Region` - `C`, and `Object` - `C`, which share the operation key `C`.

`Item` - `D`            Commands that delete items are bound to combinations such as `Line` - `D`, `Word` - `D`, and `Window` - `D`, which share the operation key `D`.

`Item` - `T`            Commands that transpose items are bound to combinations such as `Word` - `T`, `Line` - `T`, and `Window` - `T`, which share the operation key `T`.

## Modified Key Combinations

Each modified key combination contains one or more modifier keys (`Shift`, `Control`), along with another key (either alphabetic or nonalphabetic). Modifier keys are never used with item keys.

The keystrokes must overlap in a modified key combination. To execute a modified combination:

1. Press and hold the modifier key(s).
2. While holding down the modifier key(s), press the key to be modified.

The notation indicates overlapping keystrokes by naming the keys adjacently:

`modifier key other key`.

## Basic and Accelerated Keystrokes

Certain key combinations (namely, item-operation combinations and modified function keys) are considered *basic* combinations because they involve explicitly labeled keys, such as `Word` or `Definition`. Basic key bindings are recommended if you are new to the Environment, because they are easy to remember.

However, experienced users may find *accelerated* key bindings more convenient. Accelerated bindings generally involve the modifier keys in combination with keys on the main keyboard so that you can use them without moving your hands away from normal typing position.

Many commands are bound to both basic and accelerated key combinations. As an example, you can delete a word using either `Word - D` or the corresponding accelerated key combination, `Esc - D`.

## Keymap Notation

The following notations apply to the "Detailed Reference to Key Bindings" sections of the Keymap.

### Symbols

`key1 - key2` Press and release `key1`; then press `key2`.

`key1 key2` Press and hold `key1` while pressing `key2`.

`numeric 1` Press `1` on the numeric keypad.

### Numeric Arguments

You can give a numeric argument to many of the commands that are bound to keys. Indicate the desired number using the numeric keypad, and then press the key combination bound to the command. For example, `Word - D` deletes one word; the following combination deletes four words: `numeric 4 - Word - D`.

## How to Use the Keymap

Indicate negative numbers by pressing `numeric -` first. For example, the following combination shrinks a window by seven lines ("expands" it by `-7` lines):

`numeric -` - `numeric 7` - `Window` - `I`

### Case Sensitivity of Key Bindings

Although keys are shown as uppercase, the unshifted equivalent also works. This is true for the nonalphabetic characters as well. For example, `Object - d` is equivalent to `Object - D` and `Object - I` is equivalent to `Object - i`.

## **Detailed Reference to Key Bindings**

### Getting Help and Other Information

<i>Description</i>	<i>Basic Keys</i>	<i>Accelerated Keys</i>	<i>Command</i>
Determine what help is available	Help on Help		What.Does
Get help on item	Help		What.Does
Get help on key	Help on Key	Esc - G	Editor.Key.Name
Display Help window	Help Window		Editor.Image.Find
Explain underlined error	Object - ?		Common.Explain
Show time and date	What Time		What.Time
Show system load	What Load		What.Load (True)
Show current users	What Users		What.Users (True)
Show lock information for object in window	What Locks		What.Locks
Show full name of object in window	What Object		What.Object
Show access list for designated object	Show Access List		Access_List.Display

### Traversing the Environment

<i>Description</i>	<i>Basic Keys</i>	<i>Accelerated Keys</i>	<i>Command</i>
Display the Window Directory	Window - Definition	Window - ?	Editor.Window.Directory
Display object cursor is on	Definition		Common.Definition
Display object, same window	Definition In Place		Common.Definition
Display parent object	Enclosing		Common.Enclosing
Display parent object, same window	Enclosing In Place		Common.Enclosing
Display parent library, same window	Enclosing Library		Common.Enclosing
Display your home library	Esc - ↑		What.Home_Library
Set mark at current location	Mark - ↓		Editor.Mark.Push
Cycle through marks in stack	Mark - ←	Esc - M	Editor.Mark.Next
Cycle back through marks in stack	Mark - →		Editor.Mark.Previous
Return to most recent mark	Mark - ↑		Editor.Mark.Top

### Logging Off

<i>Description</i>	<i>Basic Keys</i>	<i>Accelerated Keys</i>	<i>Command</i>
Log off, unless changes aren't saved	-	-	Editor.Quit
Log off, ignoring unsaved changes	-	-	Editor.Quit(True)

### Selecting Items

<i>Description</i>	<i>Basic Keys</i>	<i>Accelerated Keys</i>	<i>Command</i>
Select successively larger structures	<b>Object</b> - <b>--</b>		<b>Common.Object.Parent</b>
Select successively smaller structures	<b>Object</b> - <b>--</b>		<b>Common.Object.Child</b>
Select previous structure, same level	<b>Object</b> - <b>↑</b>		<b>Common.Object.Previous</b>
Select next structure, same level	<b>Object</b> - <b>↓</b>		<b>Common.Object.Next</b>
Select first structure	<b>Object</b> - <b>Begin Of</b>	<b>Object</b> - <b>B</b>	<b>Common.Object.First_Child</b>
Select last structure	<b>Object</b> - <b>End Of</b>	<b>Object</b> - <b>E</b>	<b>Common.Object.Last_Child</b>
Turn off selection cursor is in	<b>Control</b> <b>X</b>		<b>Editor.Set.Designation_Off</b>

### Executing Commands

<i>Description</i>	<i>Basic Keys</i>	<i>Accelerated Keys</i>	<i>Command</i>
Create a Command window	<b>Create Command</b>		<b>Common.Create_Command</b>
Complete command name and parameters	<b>Complete</b>		<b>Common.Complete</b>
Execute a command	<b>Promote</b>		<b>Common.Promote</b>
Execute command in background	<b>Shift</b> <b>Promote</b>		<b>Command.Spawn</b>
Move to the next parameter prompt	<b>Esc</b> - <b>N</b>		<b>Editor.Cursor.Next</b>
Move to the previous parameter prompt	<b>Esc</b> - <b>U</b>		<b>Editor.Cursor.Previous</b>
Turn a prompt into text	<b>Control</b> <b>X</b>		<b>Editor.Set.Designation_Off</b>
Redisplay the previous command (undo)	<b>Object</b> - <b>U</b>		<b>Common.Undo</b>
Redisplay the next command (redo)	<b>Object</b> - <b>R</b>		<b>Common.Redo</b>
Provide prompts for the next key pressed	<b>Esc</b> - <b>Q</b>		<b>Editor.Key.Prompt</b>

### Managing Windows

<i>Description</i>	<i>Basic Keys</i>	<i>Accelerated Keys</i>	<i>Command</i>
<b>Moving between Windows</b>			
Move to the next window	<b>Window</b> - <b>↓</b>	<b>Esc</b> - <b>V</b> , <b>Shift</b> <b>↓</b>	<b>Editor.Window.Next</b>
Move to the previous window	<b>Window</b> - <b>↑</b>	<b>Esc</b> - <b>Z</b> , <b>Shift</b> <b>↑</b>	<b>Editor.Window.Previous</b>
Move to next attached window	<b>Window</b> - <b>--</b>		<b>Editor.Window.Child</b>
Move to previous attached window	<b>Window</b> - <b>--</b>		<b>Editor.Window.Parent</b>

**Managing Windows (Continued)**

Description	Basic Keys	Accelerated Keys	Command
<b>Resizing and Repositioning Windows</b>			
Join with the next window Join with the previous window Expand a window 4 lines Shrink a window 4 lines Transpose current window with previous Realign windows Copy a window	Window - J Window - Delete Window -   Window - . Window - T Window - Format Window - C		Editor.Window.Join (1) Editor.Window.Join (-1) Editor.Window.Expand Editor.Window.Expand (-4) Editor.Window.Transpose Editor.Window.Focus Editor.Window.Copy
<b>Redrawing the Screen</b>			
Redraw the screen Erase the screen, resetting the terminal	Control L Esc - L		Editor.Screen.Redraw Editor.Screen.Clear
<b>Retaining Windows</b>			
Lock a window on the screen Release a locked window	Window - Promote Window - Demote	Window - Edit	Editor.Window.Promote Editor.Window.Demote
<b>Removing Windows</b>			
Remove a window temporarily Release image, discarding changes Release image, saving changes Delete selected Window Directory entry	Window - D, Window - K, Window - X Object - G Object - X Object - D		Editor.Window.Delete Common.Abandon Common.Release Common.Object.Delete
<b>Finding Windows</b>			
Display Window Directory Display Window Directory entry	Window - Definition Definition	Window - ?	Editor.Window.Directory Common.Definition



## Moving within an Image

<i>Description</i>	<i>Basic Keys</i>	<i>Accelerated Keys</i>	<i>Command</i>
<b>By Character</b>			
Move right 1 character	<code>-</code>	<code>Control J</code>	<code>Editor.Cursor.Right</code>
Move right 8 characters	<code>numeric 8 - -</code>	<code>Esc - Control J</code>	<code>Editor.Cursor.Right(8)</code>
Move left 1 character	<code>-</code>	<code>Control H</code>	<code>Editor.Cursor.Left</code>
Move left 8 characters	<code>numeric 8 - -</code>	<code>Esc - Control H</code>	<code>Editor.Cursor.Left(8)</code>
<b>By Word</b>			
Move to next word	<code>Word - -</code>	<code>Esc - J</code>	<code>Editor.Word.Next</code>
Move to previous word	<code>Word - -</code>	<code>Esc - H</code>	<code>Editor.Word.Previous</code>
Move to beginning of word	<code>Word - Begin Of</code>	<code>Esc - A</code> , <code>Esc - B</code>	<code>Editor.Word.Beginning-Of</code>
Move to end of word	<code>Word - End Of</code>	<code>Esc - E</code>	<code>Editor.Word.End-Of</code>
<b>By Underline or Prompt</b>			
Move to next underline or prompt	<code>Esc - N</code>		<code>Editor.Cursor.Next</code>
Move to previous underline or prompt	<code>Esc - U</code>		<code>Editor.Cursor.Previous</code>
<b>By Line</b>			
Move up 1 line	<code>↑</code>	<code>Control U</code>	<code>Editor.Cursor.Up</code>
Move up 8 lines	<code>numeric 8 - ↑</code>	<code>Esc - Control U</code>	<code>Editor.Cursor.Up(8)</code>
Move down 1 line	<code>↓</code>	<code>Control N</code>	<code>Editor.Cursor.Down</code>
Move down 8 lines	<code>numeric 8 - ↓</code>	<code>Esc - Control N</code>	<code>Editor.Cursor.Down(8)</code>
Move to beginning of line	<code>Line - Begin Of</code>	<code>Control B</code>	<code>Editor.Line.Beginning-Of</code>
Move to end of line	<code>Line - End Of</code>	<code>Control E</code>	<code>Editor.Line.End-Of</code>

**Moving within an Image (Continued)**

Description	Basic Keys	Accelerated Keys	Command
<b>In a Region</b>			
Move to beginning of region Move to end of region	Region - Begin Of Region - End Of	Region - B Region - E	Editor.Region.Beginning-Of Editor.Region.End-Of
<b>By Tabs</b>			
Tab forward Tab backward	Control I Esc - Control I		Editor.Char.Tab-Forward Editor.Char.Tab-Backward
<b>By Scrolling</b>			
Scroll up Scroll down Scroll right Scroll left  Scroll to top of image Scroll to end of image Scroll current line to top Scroll current line to bottom	Image - ↑ Image - ↓ Image - → Image - ←  Image - Begin Of Image - End Of Window - Begin Of Window - End Of	Control Z Control V  Image - B Image - E Window - B Window - E	Editor.Image.Up Editor.Image.Down Editor.Image.Right Editor.Image.Left  Editor.Image.Beginning-Of Editor.Image.End-Of Editor.Window.Beginning-Of Editor.Window.End-Of
<b>By Marking Your Place</b>			
Set mark at cursor position Cycle through marks in stack Cycle back through marks in stack Return to most recent mark	Mark - ↓ Mark - → Mark - ← Mark - ↑	Control Ⓞ Esc - M	Editor.Mark.Push Editor.Mark.Next Editor.Mark.Previous Editor.Mark.Top

## General Editing Operations

Description	Basic Keys	Accelerated Keys	Command
<b>Selecting an Arbitrary Region</b>			
Select start of region	<b>Region</b> - <b>[</b>	<b>Esc</b> - <b>[</b>	<b>Editor.Region.Start</b>
Select end of region	<b>Region</b> - <b>]</b>	<b>Esc</b> - <b>]</b>	<b>Editor.Region.Finish</b>
Unselect a region	<b>Region</b> - <b>X</b>		<b>Editor.Region.Off</b>
<b>Moving and Copying Text</b>			
Copy a selected item	<b>Region</b> - <b>C</b>		<b>Editor.Region.Copy</b>
Move a selected item	<b>Region</b> - <b>M</b>		<b>Editor.Region.Move</b>
Duplicate a single line	<b>Line</b> - <b>C</b>	<b>Esc</b> - <b>Control</b> <b>C</b>	<b>Editor.Line.Copy</b>
<b>Deleting Text</b>			
Delete character — forward	<b>Control</b> <b>D</b>		<b>Editor.Char.Delete_Forward</b>
Delete character — backward	<b>Delete</b>		<b>Editor.Char.Delete_Backward</b>
Reduce multiple blanks to one	<b>Control</b> <b>Delete</b>		<b>Editor.Char.Delete_Spaces</b>
Delete word	<b>Word</b> - <b>D</b>	<b>Esc</b> - <b>D</b>	<b>Editor.Word.Delete</b>
Delete to end of word	<b>Word</b> - <b>K</b>	<b>Esc</b> - <b>K</b>	<b>Editor.Word.Delete_Forward</b>
Delete to beginning of word	<b>Word</b> - <b>Delete</b>	<b>Esc</b> - <b>Delete</b>	<b>Editor.Word.Delete_Backward</b>
Delete line	<b>Line</b> - <b>D</b>	<b>Esc</b> - <b>Control</b> <b>D</b>	<b>Editor.Line.Delete</b>
Delete to end of line	<b>Line</b> - <b>K</b>	<b>Control</b> <b>K</b>	<b>Editor.Line.Delete_Forward</b>
Delete to beginning of line	<b>Line</b> - <b>Delete</b>	<b>Esc</b> - <b>Control</b> <b>F</b>	<b>Editor.Line.Delete_Backward</b>
Delete selected item	<b>Region</b> - <b>D</b> , <b>Region</b> - <b>K</b>		<b>Editor.Region.Delete</b>
<b>Searching and Replacing Text</b>			
Search for next occurrence	<b>Control</b> <b>F</b>		<b>Editor.Search.Next</b>
Search for previous occurrence	<b>Control</b> <b>R</b>		<b>Editor.Search.Previous</b>
Replace next occurrence	<b>Esc</b> - <b>F</b>		<b>Editor.Search.Replace_Next</b>
Replace previous occurrence	<b>Esc</b> - <b>R</b>		<b>Editor.Search.Replace_Previous</b>

**General Editing Operations (Continued)**

Description	Basic Keys	Accelerated Keys	Command
<b>Entering Text</b>			
<p>Quote a special character</p> <p>Split line, cursor on new line</p> <p>Split line, cursor on old line</p> <p>Join 2 lines</p> <p>Enter text in insert mode</p> <p>Enter text in overwrite mode</p> <p>Show current line number</p>	<p><b>Esc</b> - <b>'</b></p> <p><b>Line</b> - <b>I</b></p> <p><b>Line</b> - <b>O</b></p> <p><b>Line</b> - <b>J</b></p> <p><b>Image</b> - <b>I</b></p> <p><b>Image</b> - <b>O</b></p> <p><b>Line</b> - <b>?</b></p>	<p><b>Control</b> <b>O</b></p> <p><b>Esc</b> - <b>O</b>,</p> <p><b>Esc</b> - <b>Control</b> <b>O</b></p>	<p><b>Editor.Char.Quote</b></p> <p><b>Editor.Line.Insert</b></p> <p><b>Editor.Line.Open</b></p> <p><b>Editor.Line.Join</b></p> <p><b>Editor.Set.Insert_Mode(True)</b></p> <p><b>Editor.Set.Insert_Mode(False)</b></p> <p><b>What.Line</b></p>
<b>Transposing Text</b>			
<p>Transpose with previous character</p> <p>Transpose with previous word</p> <p>Transpose with previous line</p>	<p><b>Control</b> <b>T</b></p> <p><b>Word</b> - <b>T</b></p> <p><b>Line</b> - <b>T</b></p>	<p><b>Esc</b> - <b>T</b></p> <p><b>Esc</b> - <b>Control</b> <b>T</b></p>	<p><b>Editor.Char.Transpose</b></p> <p><b>Editor.Word.Transpose</b></p> <p><b>Editor.Line.Transpose</b></p>
<b>Controlling Case</b>			
<p>Capitalize to end of word</p> <p>Capitalize words to end of line</p> <p>Capitalize every word in region</p> <p>Make lowercase to end of word</p> <p>Make lowercase to end of line</p> <p>Convert entire region to lowercase</p> <p>Make uppercase to end of word</p> <p>Make uppercase to end of line</p> <p>Convert entire region to uppercase</p>	<p><b>Word</b> - <b>^</b></p> <p><b>Line</b> - <b>^</b></p> <p><b>Region</b> - <b>^</b></p> <p><b>Word</b> - <b>&lt;</b></p> <p><b>Line</b> - <b>&lt;</b></p> <p><b>Region</b> - <b>&lt;</b></p> <p><b>Word</b> - <b>&gt;</b></p> <p><b>Line</b> - <b>&gt;</b></p> <p><b>Region</b> - <b>&gt;</b></p>	<p><b>Esc</b> - <b>^</b></p> <p><b>Esc</b> - <b>&lt;</b></p> <p><b>Esc</b> - <b>&gt;</b></p>	<p><b>Editor.Word.Capitalize</b></p> <p><b>Editor.Line.Capitalize</b></p> <p><b>Editor.Region.Capitalize</b></p> <p><b>Editor.Word.Lower_Case</b></p> <p><b>Editor.Line.Lower_Case</b></p> <p><b>Editor.Region.Lower_Case</b></p> <p><b>Editor.Word.Upper_Case</b></p> <p><b>Editor.Line.Upper_Case</b></p> <p><b>Editor.Region.Upper_Case</b></p>

## General Editing Operations (Continued)

Description	Basic Keys	Accelerated Keys	Command
<b>Holding and Retrieving Text</b>			
Hold selected text Retrieve most recently held text Retrieve previous held text Retrieve next held text	Region - I Region - ↑ Region - ← Region - →	Control - C Control - Y Esc - C, Esc - Y	Editor.Hold_Stack.Push Editor.Hold_Stack.Top Editor.Hold_Stack.Previous Editor.Hold_Stack.Next
<b>Formatting Text</b>			
Center the line cursor is on Fill text in selected region Justify text in selected region Automatically wrap lines Do not wrap lines	Line - § Region - Format Region - Complete Image - F Image - X		Editor.Line.Center Editor.Region.Fill Editor.Region.Justify Editor.Set.Fill_Mode(True) Editor.Set.Fill_Mode(False)

### Writing Text Files

Description	Basic Keys	Accelerated Keys	Command
<b>Accessing Text Files</b>			
Create a new text file Display existing text file Open text file for editing Revert to last saved version	Create Text Definition Edit Object . L		Text.Create Common.Definition Common.Edit Common.Revert
<b>Saving Changes</b>			
Save, leaving open for editing Save, making read only	Enter Promote		Common.Commit Common.Promote
<b>Terminating Edit</b>			
Remove image, discarding changes Remove image, saving changes	Object . G Object . X		Common.Abandon Common.Release
<b>Selecting Substructures within Text</b>			
Select current word Select current sentence Select current paragraph Select smaller structure Select previous structure, same level Select next structure, same level Turn off selection	Object . -- numeric 2 . Object . -- numeric 3 . Object . -- Object . -- Object . ↑ Object . ↓ Control X		Common.Object.Parent Common.Object.Parent Common.Object.Parent Common.Object.Child Common.Object.Previous Common.Object.Next Editor.Set.Designation_Off

## Writing Ada Programs

<i>Description</i>	<i>Basic Keys</i>	<i>Accelerated Keys</i>	<i>Command</i>
<b>Creating Ada Programs</b>			
Create an Ada unit in library Build a body Build a private part Put temporary name in library	Object . I Create Body Create Private -		Common.Object.Insert Ada.Create_Body Ada.Create_Private Ada.Install_Stub
<b>Accessing Ada Programs</b>			
Display Ada unit, read only Demote to source, open for editing	Definition Edit		Common.Definition Common.Edit
<b>Saving Changes and Terminating Edit</b>			
Save, leaving open for editing Release image, discarding changes Release image, saving changes Revert to last version	Enter Object . G Object . X Object . L		Common.Commit Common.Abandon Common.Release Common.Revert
<b>Checking for Errors</b>			
Complete and check syntax Check for semantic errors Explain underlined error Move to next underlined error Move to previous underlined error Remove underline from error Clear all underlined errors Redisplay cleared errors	Format Semanticize Object . ? Esc . N Esc . U Control X Underlines Off Show Errors		Common.Format Common.Semanticize Common.Explain Editor.Cursor.Next Editor.Cursor.Previous Editor.Set.Designation_Off Common.Clear_Underlining Ada.Get_Errors

**Writing Ada Programs (Continued)**

Description	Basic Keys	Accelerated Keys	Command
<b>Changing the Compilation State</b>			
Change unit to source state from any state Change unit to installed state from any state Change unit to coded state from any state	Source Unit Install Unit Code Unit		Ada.Source_Unit Ada.Install_Unit Ada.Code_Unit
<b>Changing to a Higher Compilation State</b>			
Promote unit to next higher state Code unit and those it depends on In this world only Across worlds Install unit and those it depends on In this world only	Promote  Code (This World) Code (All Worlds)  Install (This World)		Common.Promote  Compilation.Make Compilation.Make  Compilation.Promote
<b>Changing to a Lower Compilation State</b>			
Demote unit to next lower state Demote unit and dependents to source In this world only	Demote  Source (This World)		Common.Demote  Compilation.Demote
<b>Selecting Structures within Ada Programs</b>			
Select successively larger structures Select successively smaller structures Select previous structure, same level Select next structure, same level Select first structure Select last structure Turn off selection cursor is in	Object . -- Object . -- Object . ↑ Object . ↓ Object . Begin Of Object . End Of Control X	   Object . B Object . E	Common.Object.Parent Common.Object.Child Common.Object.Previous Common.Object.Next Common.Object.First_Child Common.Object.Last_Child Editor.Set.Designation_Off



## Writing Ada Programs (Continued)

Description	Basic Keys	Accelerated Keys	Command
<b>Modifying Ada Programs</b>			
Edit selected Ada structure Insert Ada structures(s) in program Delete selected Ada structure  Copy selected Ada structure Move selected Ada structure Withdraw Ada unit stub	Edit Object - I Object - D, Object - K Object - C Object - M Withdraw Unit		Common.Edit Common.Object.Insert Common.Object.Delete  Common.Object.Copy Common.Object.Move Ada.Withdraw
<b>Entering Comments and Special Strings</b>			
Comment selected item or region Uncomment selected item or region Tab forward to comment	- - -		Region.Comment Region.Uncomment Editor.Char.Tab.To.Comment
<b>Browsing Ada Programs</b>			
Display other part of Ada unit Display other part, same window Display Ada unit cursor is on Display parent object Set mark at current location Cycle through marks in stack Cycle back through marks in stack Return to most recent mark	Other Part Other Part In Place Definition Enclosing Mark - ↓ Mark - ← Mark - → Mark - ↑	Esc - M	Ada.Other_Part Ada.Other_Part Common.Definition Common.Enclosing Editor.Mark.Push Editor.Mark.Next Editor.Mark.Previous Editor.Mark.Top
<b>Checking Using Occurrences</b>			
Show uses of selected identifier In this unit only In any unit Show unused declarations In this unit only Check other units	Show Usage (Unit) Show Usage  Show Unused (Unit) Show Unused		Ada.Show_Usage Ada.Show_Usage  Ada.Show_Unused Ada.Show_Unused

### Debugging Ada Programs

<i>Description</i>	<i>Basic Keys</i>	<i>Accelerated Keys</i>	<i>Command</i>
Execute program with Debugger on Display Debugger window Show current statement in source	<b>Esc</b> - <b>Promote</b> <b>Debugger Window</b> <b>Show Source</b>		<b>Command.Debug</b> <b>Debug.Current-Debugger</b> <b>Debug.Source</b>
<b>Stepping and Executing</b>			
Continue program execution Step one statement Step one statement at same level Stop task execution Display information about tasks Display task rendezvous info	<b>Execute</b> <b>Run</b> <b>Run Local</b> <b>Stop</b> <b>Task Display</b> -		<b>Debug.Execute</b> <b>Debug.Run</b> <b>Debug.Run (Local)</b> <b>Debug.Stop</b> <b>Debug.Task-Display</b> <b>Debug.Information</b>
<b>Setting and Removing Breakpoints</b>			
Set breakpoints with default lifetime Display breakpoints Reactivate existing breakpoints Remove breakpoints	<b>Break</b> <b>Show Breaks</b> <b>Activate</b> <b>Remove Breaks</b>		<b>Debug.Break</b> <b>Debug.Show</b> <b>Debug.Activate</b> <b>Debug.Remove</b>
<b>Viewing Stacks</b>			
Display calling stack	<b>Stack</b>		<b>Debug.Stack</b>
<b>Displaying and Modifying Variables</b>			
Display values of selected variables Modify value of selected variable	<b>Put</b> <b>Modify</b>		<b>Debug.Put</b> <b>Debug.Modify</b>
<b>Handling Exceptions</b>			
Stop execution when exception raised Do not stop when exception raised Remove handling for this exception	<b>Catch</b> <b>Propagate</b> -		<b>Debug.Catch</b> <b>Debug.Propagate</b> <b>Debug.Forget</b>

## Managing Libraries

<i>Description</i>	<i>Basic Keys</i>	<i>Accelerated Keys</i>	<i>Command</i>
<b>Creating Libraries</b>			
Create a directory Create a world	<div style="border: 1px solid black; padding: 2px; display: inline-block;">Create Directory</div> <div style="border: 1px solid black; padding: 2px; display: inline-block;">Create World</div>		Library.Create_Directory Library.Create_World
<b>Manipulating Objects in Libraries</b>			
Create an Ada unit in library Create a text file in library Delete selected object from library  Undelete selected object from library Print selected object Show access list for designated object	<div style="border: 1px solid black; padding: 2px; display: inline-block;">Object - I</div> <div style="border: 1px solid black; padding: 2px; display: inline-block;">Create Text</div> <div style="border: 1px solid black; padding: 2px; display: inline-block;">Object - D,</div> <div style="border: 1px solid black; padding: 2px; display: inline-block;">Object - K</div> <div style="border: 1px solid black; padding: 2px; display: inline-block;">Object - U</div> <div style="border: 1px solid black; padding: 2px; display: inline-block;">Print</div> <div style="border: 1px solid black; padding: 2px; display: inline-block;">Show Access List</div>		Common.Object.Insert Text.Create Common.Object.Delete  Common.Object.Undo Queue.Print Access_List.Display
<b>Controlling Library Display</b>			
Toggle information on library objects Show more detail Show less detail	<div style="border: 1px solid black; padding: 2px; display: inline-block;">Object - ?</div> <div style="border: 1px solid black; padding: 2px; display: inline-block;">Object - I</div> <div style="border: 1px solid black; padding: 2px; display: inline-block;">Object - .</div>		Common.Explain Common.Expand Common.Elide

## Using CMVC

<i>Description</i>	<i>Basic Keys</i>	<i>Accelerated Keys</i>	<i>Command</i>
Check out designated object	-		Cmvc.Check_Out
Check in designated object	-		Cmvc.Check_In
Accept changes for designated object	-		Cmvc.Accept_Changes
Show objects that are checked out			
In this view	-		Cmvc.Show_Checked_Out_In_View
By you, any view	-		Cmvc.Show_Checked_Out_By_User
Show info about designated object	-		Cmvc.Show
Show out-of-date objects in this view	-		Cmvc.Show_Out_Of_Date_Objects

### Managing Links

<i>Description</i>	<i>Basic Keys</i>	<i>Accelerated Keys</i>	<i>Command</i>
<b>Accessing Links</b>			
List links Edit links display Refresh link image	- - Object - L		Links.Display Links.Edit Common.Revert
<b>Removing the Link Editor</b>			
Remove window temporarily Release image permanently	Window - D Object - X		Editor.Window.Delete Common.Release
<b>Selecting Links</b>			
Select link cursor is on Select all links Select previous link Select next link Select first link in image Select last link in image	Object - -- Object - -- Object - ↑ Object - ↓ Object - Begin Of Object - End Of	Object - B Object - E	Common.Object.Child Common.Object.Parent Common.Object.Previous Common.Object.Next Common.Object.First_Child Common.Object.Last_Child
<b>Modifying Links</b>			
Add a new link—simple method Add a new link Give selected link another source Delete selected link	- Object - I Edit Object - D, Object - K		Links.Add Common.Object.Insert Common.Edit Common.Object.Delete
<b>Traversing Linked Ada Units</b>			
Go to source unit of current link Go to world associated with links List Ada units that use current link	Definition Enclosing Object - ?		Common.Definition Common.Enclosing Common.Explain
<b>Controlling the Display</b>			
Toggle order of kind of link Toggle classes of source of link	Object - I Object - -		Common.Expand Common.Elide

## Managing Searchlists

<i>Description</i>	<i>Basic Keys</i>	<i>Accelerated Keys</i>	<i>Command</i>
<b>Accessing the Searchlist</b>			
Edit or view searchlist Refresh searchlist image	- Object - L		Search-List.Edit Common.Revert
<b>Removing the Searchlist Editor</b>			
Remove window temporarily Release image permanently	Window - D Object - X		Editor.Window.Delete Common.Release
<b>Selecting Entries</b>			
Select entry cursor is on Select all entries Select next entry Select previous entry Select first entry on list Select last entry on list Go to world named by current entry	Object - -- Object - -- Object - ↓ Object - ↑ Object - Begin Of Object - End Of Definition	Object - B Object - E	Common.Object.Child Common.Object.Parent Common.Object.Next Common.Object.Previous Common.Object.First-Child Common.Object.Last-Child Common.Definition
<b>Modifying the Searchlist</b>			
Add a new entry Delete selected entry  Move selected entry	Object - I Object - D Object - K Object - M		Common.Object.Insert Common.Object.Delete  Common.Object.Move

## Using Keyboard Macros

<i>Description</i>	<i>Basic Keys</i>	<i>Accelerated Keys</i>	<i>Command</i>
Start macro definition	Mark . Begin Of	Mark . [F]	Editor.Macro.Start
End macro definition	Mark . End Of	Mark . [J]	Editor.Macro.Finish
Execute macro	Mark . Promote	Esc . X	Editor.Macro.Execute
Bind macro to key	Mark . Definition		Editor.Macro.Bind

## Using Environment I/O Resources

<i>Description</i>	<i>Basic Keys</i>	<i>Accelerated Keys</i>	<i>Command</i>
Indicate end of input to program	numeric .		Text.End.Of.Input
Commit interactive input	Promote	Enter	Common.Commit

## Managing Jobs

<i>Description</i>	<i>Basic Keys</i>	<i>Accelerated Keys</i>	<i>Command</i>
Disconnect job from terminal	Control   G		Job.Interrupt
Kill job	Job Kill	Esc . G	Job.Kill(0)
Stop running jobs	Job Disable		Job.Disable(0)
Resume stopped jobs	Job Enable		Job.Enable(0)
Reconnect job	Job Connect		Job.Connect(0)

# Master Reference to Key Bindings by Command

Legend	Control	Object	Region	Window	Image	Line	Word	Mark	Facit	Rational
C	ESC_C	ESC_C	ESC_C	ESC_C	ESC_C	ESC_C	ESC_C	ESC_C	ESC_C	ESC_C
S	ESC_S	ESC_S	ESC_S	ESC_S	ESC_S	ESC_S	ESC_S	ESC_S	ESC_S	ESC_S
X4	ESC_X4	ESC_X4	ESC_X4	ESC_X4	ESC_X4	ESC_X4	ESC_X4	ESC_X4	ESC_X4	ESC_X4
X5	ESC_X5	ESC_X5	ESC_X5	ESC_X5	ESC_X5	ESC_X5	ESC_X5	ESC_X5	ESC_X5	ESC_X5
X6	ESC_X6	ESC_X6	ESC_X6	ESC_X6	ESC_X6	ESC_X6	ESC_X6	ESC_X6	ESC_X6	ESC_X6
Command	ESC_CMD	ESC_CMD	ESC_CMD	ESC_CMD	ESC_CMD	ESC_CMD	ESC_CMD	ESC_CMD	ESC_CMD	ESC_CMD
Access_List.Display	ESC_ALD	ESC_ALD	ESC_ALD	ESC_ALD	ESC_ALD	ESC_ALD	ESC_ALD	ESC_ALD	ESC_ALD	ESC_ALD
Ada.Code_Unit	ESC_ACU	ESC_ACU	ESC_ACU	ESC_ACU	ESC_ACU	ESC_ACU	ESC_ACU	ESC_ACU	ESC_ACU	ESC_ACU
Ada.Create_Body	ESC_CBO	ESC_CBO	ESC_CBO	ESC_CBO	ESC_CBO	ESC_CBO	ESC_CBO	ESC_CBO	ESC_CBO	ESC_CBO
Ada.Create_Private	ESC_CPR	ESC_CPR	ESC_CPR	ESC_CPR	ESC_CPR	ESC_CPR	ESC_CPR	ESC_CPR	ESC_CPR	ESC_CPR
Ada.Delete_Blank_Line	ESC_DBL	ESC_DBL	ESC_DBL	ESC_DBL	ESC_DBL	ESC_DBL	ESC_DBL	ESC_DBL	ESC_DBL	ESC_DBL
Ada.Get_Errors	ESC_GER	ESC_GER	ESC_GER	ESC_GER	ESC_GER	ESC_GER	ESC_GER	ESC_GER	ESC_GER	ESC_GER
Ada.Insert_Blank_Line	ESC_IBL	ESC_IBL	ESC_IBL	ESC_IBL	ESC_IBL	ESC_IBL	ESC_IBL	ESC_IBL	ESC_IBL	ESC_IBL
Ada.Install_Stub	ESC_IST	ESC_IST	ESC_IST	ESC_IST	ESC_IST	ESC_IST	ESC_IST	ESC_IST	ESC_IST	ESC_IST
Ada.Install_Unit	ESC_IU	ESC_IU	ESC_IU	ESC_IU	ESC_IU	ESC_IU	ESC_IU	ESC_IU	ESC_IU	ESC_IU
Ada.Make_Inline	ESC_MIL	ESC_MIL	ESC_MIL	ESC_MIL	ESC_MIL	ESC_MIL	ESC_MIL	ESC_MIL	ESC_MIL	ESC_MIL
Ada.Make_Separate	ESC_MSP	ESC_MSP	ESC_MSP	ESC_MSP	ESC_MSP	ESC_MSP	ESC_MSP	ESC_MSP	ESC_MSP	ESC_MSP
Ada.Other_Part ( Name => <Image...> )	ESC_OPP	ESC_OPP	ESC_OPP	ESC_OPP	ESC_OPP	ESC_OPP	ESC_OPP	ESC_OPP	ESC_OPP	ESC_OPP
Ada.Other_Part ( Name => <Image...> )	ESC_OPP	ESC_OPP	ESC_OPP	ESC_OPP	ESC_OPP	ESC_OPP	ESC_OPP	ESC_OPP	ESC_OPP	ESC_OPP
Ada.Show_Unused ( In_Unit => <I...> )	ESC_SU	ESC_SU	ESC_SU	ESC_SU	ESC_SU	ESC_SU	ESC_SU	ESC_SU	ESC_SU	ESC_SU
Ada.Show_Unused ( In_Unit => <I...> )	ESC_SU	ESC_SU	ESC_SU	ESC_SU	ESC_SU	ESC_SU	ESC_SU	ESC_SU	ESC_SU	ESC_SU
Ada.Show_Usage ( Name => <Cursor...> )	ESC_SU	ESC_SU	ESC_SU	ESC_SU	ESC_SU	ESC_SU	ESC_SU	ESC_SU	ESC_SU	ESC_SU
Ada.Show_Usage ( Name => <Cursor...> )	ESC_SU	ESC_SU	ESC_SU	ESC_SU	ESC_SU	ESC_SU	ESC_SU	ESC_SU	ESC_SU	ESC_SU
Ada.Show_Usage ( Name => <Cursor...> )	ESC_SU	ESC_SU	ESC_SU	ESC_SU	ESC_SU	ESC_SU	ESC_SU	ESC_SU	ESC_SU	ESC_SU
Ada.Source_Unit	ESC_SU	ESC_SU	ESC_SU	ESC_SU	ESC_SU	ESC_SU	ESC_SU	ESC_SU	ESC_SU	ESC_SU
Ada.Withdraw	ESC_WD	ESC_WD	ESC_WD	ESC_WD	ESC_WD	ESC_WD	ESC_WD	ESC_WD	ESC_WD	ESC_WD
Conv.Accept_Changes ( Destination... )	ESC_ACH	ESC_ACH	ESC_ACH	ESC_ACH	ESC_ACH	ESC_ACH	ESC_ACH	ESC_ACH	ESC_ACH	ESC_ACH
Conv.Check_In ( What_Object => <I...> )	ESC_CHI	ESC_CHI	ESC_CHI	ESC_CHI	ESC_CHI	ESC_CHI	ESC_CHI	ESC_CHI	ESC_CHI	ESC_CHI
Conv.Check_Out ( What_Object => <I...> )	ESC_CHO	ESC_CHO	ESC_CHO	ESC_CHO	ESC_CHO	ESC_CHO	ESC_CHO	ESC_CHO	ESC_CHO	ESC_CHO
Conv.Show ( Objects => <Cursor...> )	ESC_SHO	ESC_SHO	ESC_SHO	ESC_SHO	ESC_SHO	ESC_SHO	ESC_SHO	ESC_SHO	ESC_SHO	ESC_SHO
Conv.Show_Checked_Out_By_User ( ... )	ESC_SCH	ESC_SCH	ESC_SCH	ESC_SCH	ESC_SCH	ESC_SCH	ESC_SCH	ESC_SCH	ESC_SCH	ESC_SCH
Conv.Show_Checked_Out_In_View ( ... )	ESC_SCI	ESC_SCI	ESC_SCI	ESC_SCI	ESC_SCI	ESC_SCI	ESC_SCI	ESC_SCI	ESC_SCI	ESC_SCI
Conv.Show_Out_Of_Date_Objects ( ... )	ESC_SCO	ESC_SCO	ESC_SCO	ESC_SCO	ESC_SCO	ESC_SCO	ESC_SCO	ESC_SCO	ESC_SCO	ESC_SCO
Command.Debug	ESC_CMD	ESC_CMD	ESC_CMD	ESC_CMD	ESC_CMD	ESC_CMD	ESC_CMD	ESC_CMD	ESC_CMD	ESC_CMD
Command.Spawn	ESC_CMD	ESC_CMD	ESC_CMD	ESC_CMD	ESC_CMD	ESC_CMD	ESC_CMD	ESC_CMD	ESC_CMD	ESC_CMD
Common.Abandon	ESC_CMD	ESC_CMD	ESC_CMD	ESC_CMD	ESC_CMD	ESC_CMD	ESC_CMD	ESC_CMD	ESC_CMD	ESC_CMD

Common.Clear_Underlining	ESC_F10	ENTER	C.F16
Common.Commit	ESC_F10	ENTER	S.ENTER
Common.Complete	X5		C.CARRIAGE_RETURN
Common.Create_Command	F8		ENTER
Common.Definition	X1.F4		CS.CARRIAGE_RETURN
Common.Definition ( In_Place => ... )			COMPLI
Common.Definition ( Name => * <Cu...> )	F4		F15
Common.Definition ( Name => * <Cu...> )	S.F4		OBJECT.F10
Common.Definition ( Name => * <Cu...> )	S.F7		CH.RIGHT
Common.Demote	F7		S.RIGHT
Common.Edit	X1..''		F10
Common.Elise	C.F4		
Common.Enclosing	ESC_F4		S.F10
Common.Enclosing ( In_Place => F... )	ESC_F4		S.F14
Common.Enclosing ( In_Place => I... )	ESC_C.F4		F14
Common.Enclosing ( In_Place => I... )	X1..''		OBJECT..''
Common.Expand	X1..''		OBJECT..'>
Common.Explain	ESC_SLASH		M.F10
Common.Explain	ESC_PLUS		CM.LEFT
Common.Explain	X1..''		S.LEFT
Common.Explain	ESC_QUERY		MS.F10
Common.Explain	X1..''		CMS.F10
Common.Explain	X1..''		C.EXCLAM
Common.Explain	X1..''		OBJECT..''
Common.Explain	X1..''		C.J
Common.Explain	ESC_SLASH		OBJECT..''
Common.Explain	ESC_PLUS		OBJECT..'p'
Common.Explain	X1..''		F17
Common.Explain	ESC_QUERY		C.QUERY
Common.Explain	X1..''		OBJECT..'/'
Common.Explain	X1..''		C.SLASH
Common.Explain	X6		FORMAT
Common.Explain	X1.RIGHT		OBJECT.RIGHT
Common.Explain	X1..'c'		C.RIGHT
Common.Explain	X1..'C'		OBJECT..'c'
Common.Explain	X1..'k'		OBJECT..'C'
Common.Explain	X1..'D'		OBJECT..'d'
Common.Explain	X1..'k'		OBJECT..'D'
Common.Explain	X1..'d'		OBJECT..'k'
Common.Explain	X1..'B'		OBJECT..'k'
Common.Explain	X1..'b'		OBJECT..'B'
Common.Explain	X1..'b'		OBJECT..'b'
Common.Explain	X1..'i'		OBJECT..'i'
Common.Explain	X1..'I'		C.F15
Common.Explain	X1..'i'		OBJECT..'I'
Common.Explain	X1..'e'		C.END_OF
Common.Explain	X1..'E'		OBJECT.END_OF
Common.Explain	X1..'M'		OBJECT..'M'
Common.Explain	X1..'m'		OBJECT..'m'
Common.Explain	X1.DOWN		C.DOWN
Common.Explain	X1.LEFT		OBJECT.DOWN
Common.Explain	X1.UP		OBJECT.LEFT
Common.Explain	X4		C.LEFT
Common.Explain			OBJECT.UP
Common.Explain			C.UP
Common.Explain			PROMOT

**Common.Redo**  
**Common.Release**  
**Common.Revert**  
**Common.Semanticize**  
**Common.Sort\_Image**  
**Common.Undo**  
 Compilation.Demote ( Unit => "<S...  
 Compilation.Demote ( Unit => "<S...  
 Compilation.Demote ( Unit => "<S...  
 Compilation.Demote ( Unit => "<S...  
 Compilation.Make ( Unit => "<Ima...  
 Compilation.Make ( Unit => "<Ima...  
 Compilation.Promote ( Unit => "<...  
 Compilation.Promote ( Unit => "<...  
 Debug.Activate ( Breakpoint => 0 )  
 Debug.Break  
 Debug.Break ( Default\_Lifetime = ...  
 Debug.Catch  
 Debug.Current.Debugger ( "" )  
 Debug.Execute  
 Debug.Forget  
 Debug.Information ( Debug.Rendez...  
 Debug.Modify ( New\_Value => "", ...  
 Debug.Propagate  
 Debug.Put  
 Debug.Remove ( Breakpoint => 0 )  
 Debug.Run  
 Debug.Run ( Debug.Returned )  
 Debug.Run ( Stop\_At => Debug.Loc...  
 Debug.Set\_Value ( Variable => De...  
 Debug.Set\_Value ( Variable => De...  
 Debug.Set\_Value ( Variable => De...  
 Debug.Show ( Debug.Exceptions )  
 Debug.Source  
 Debug.Source ( Location => "", S...  
 Debug.Stack  
 Debug.Stop  
 Debug.Stop ( Name => "" )  
 Debug.Task\_Display  
 Editor.Char.Capitalize  
 Editor.Char.Delete\_Backward  
 Editor.Char.Delete\_Forward  
 Editor.Char.Delete\_Spaces  
 Editor.Char.Insert\_Character ( 1...  
 Editor.Char.Insert\_String ( "" ) )

Editor.Char.Insert\_String ( "" ) )  
 Editor.Char.Insert\_String ( " :=" )  
 Editor.Char.Insert\_String ( " =>" )  
 Editor.Char.Lower\_Case  
 Editor.Char.Quote  
 Editor.Char.Tab\_Backward  
 Editor.Char.Tab\_Forward  
 Editor.Char.Tab\_To\_Comment  
 Editor.Char.Transpose  
 Editor.Char.Upper\_Case  
 Editor.Cursor.Backward  
 Editor.Cursor.Down  
 Editor.Cursor.Down ( 8 )  
 Editor.Cursor.Forward  
 Editor.Cursor.Left  
 Editor.Cursor.Left ( 8 )  
 Editor.Cursor.Next  
 Editor.Cursor.Next ( Prompt => F...  
 Editor.Cursor.Next ( Prompt => I...  
 Editor.Cursor.Next ( Prompt => I...  
 Editor.Cursor.Previous  
 Editor.Cursor.Previous ( Prompt ...  
 Editor.Cursor.Previous ( Prompt ...  
 Editor.Cursor.Previous ( Prompt ...  
 Editor.Cursor.Right ( 8 )  
 Editor.Cursor.Up  
 Editor.Cursor.Up

C\_0  
 C\_9  
 C\_LEFT\_PAREN  
 C\_COLON  
 C\_SEMICOLON  
 C\_EQUAL  
 C\_PLUS  
 C\_LESS\_THAN  
 C\_COMMA  
 C\_TICK  
 C\_QUOTE  
 CS\_TICK  
 CS\_TAB  
 C\_TAB  
 S\_TAB  
 TAB  
 MS\_TAB  
 OBJECT\_TAB  
 M\_TAB  
 C\_J  
 CS\_J  
 C\_PERIOD  
 C\_GREATER\_THAN  
 CS\_B  
 C\_B  
 C\_N  
 C\_N  
 LINE\_DOWN  
 CS\_N  
 DOWN  
 CM\_N  
 CM\_N  
 CS\_F  
 CS\_F  
 CS\_H  
 C\_H  
 LEFT  
 LEFT  
 CM\_H  
 CM\_H  
 M\_DOWN  
 S\_F10  
 M\_F10  
 F10  
 M\_N  
 M\_N  
 MS\_N  
 MS\_N  
 M\_U  
 M\_U  
 M\_UP  
 CS\_F18  
 CM\_F18  
 CM\_F18  
 C\_J  
 C\_J  
 CS\_J  
 RIGHT  
 RIGHT  
 CM\_F  
 CM\_F  
 CM\_J  
 CM\_J  
 C\_U  
 C\_U  
 CS\_U  
 CS\_U

OBJECT.'r'  
 OBJECT.'R'  
 OBJECT.'x'  
 OBJECT.'X'  
 OBJECT.'l'  
 OBJECT.'L'  
 F16  
 OBJECT.'S'  
 OBJECT.'s'  
 OBJECT.'u'  
 OBJECT.'U'  
 MS.F14  
 CS.F14  
 CMS.F14  
 CM.F14  
 CM.F14  
 CMS.F13  
 MS.F13  
 M.F13  
 M.F7  
 C.F7  
 S.F7  
 S.F7  
 CMS.F9  
 S.F6  
 S.F8  
 CMS.F8  
 CM.F9  
 M.F8  
 F9  
 CM.F7  
 F6  
 M.F6  
 C.F6  
 C.F6  
 S.F9  
 M.F9  
 CMS.F7  
 CM.F8  
 F7  
 F8  
 CM.F6  
 CMS.F6  
 C.6  
 C\_CIRCUMFLEX  
 DELETE  
 C.D  
 CS.D  
 CS\_DELETE  
 CS\_DELETE  
 MS\_SPACE  
 CS\_SPACE  
 CM\_SPACE  
 M\_SPACE  
 CMS\_SPACE  
 C\_SPACE  
 S\_SPACE  
 C\_RIGHT\_PAREN



UP	LINE.UP		
CM_S_U	CM_S_U		
REGION.'p'	REGION.'p'		
REGION.'p'	REGION.'p'		
REGION.DELETE	REGION.DELETE		
M_Y	M_Y		
MS_Y	MS_Y		
M_C	REGION.RIGHT		
MS_C	REGION.LEFT		
C_C	REGION.DOWN		
CS_C	REGION.'r'		
REGION.'r'	REGION.'b'		
REGION.'B'	REGION.'B'		
REGION.'t'	REGION.'t'		
REGION.UP	REGION.UP		
CS_Y	CS_Y		
S_BEGIN_OF	S_BEGIN_OF		
IMAGE_BEGIN_OF	IMAGE_BEGIN_OF		
C_V	C_V		
S_DOWN	S_DOWN		
IMAGE.DOWN	IMAGE.DOWN		
CS_V	CS_V		
IMAGE_END_OF	IMAGE_END_OF		
S_END_OF	S_END_OF		
IMAGE.'?'	IMAGE.'?'		
IMAGE.'/'	IMAGE.'/'		
C.F11	C.F11		
MS.LEFT	MS.LEFT		
IMAGE.LEFT	IMAGE.LEFT		
MS.RIGHT	MS.RIGHT		
IMAGE.RIGHT	IMAGE.RIGHT		
IMAGE.UP	IMAGE.UP		
CS_Z	CS_Z		
S_UP	S_UP		
C_Z	C_Z		
C.O	C.O		
CS_Q	CS_Q		
M.F11	M.F11		
M.Q	M.Q		
MS.Q	MS.Q		
F12	F12		
CM_S_B	CM_S_B		
CM_L_B	CM_L_B		
CM_A	CM_A		
CS_A	CS_A		
C_A	C_A		
ESC_C_U	ESC_C_U		
ESC_Y	ESC_Y		
ESC_C	ESC_C		
X2.RIGHT	X2.RIGHT		
ESC_S_Y	ESC_S_Y		
ESC_S_C	ESC_S_C		
X2.LEFT	X2.LEFT		
C_C	C_C		
X2.DOWN	X2.DOWN		
C_Y	C_Y		
X2.UP	X2.UP		
PF1.'b'	PF1.'b'		
PF1.'B'	PF1.'B'		
C_V	C_V		
PF1.DOWN	PF1.DOWN		
PF1.END_OF	PF1.END_OF		
PF1.'e'	PF1.'e'		
PF1.'e'	PF1.'e'		
PF1.TAB	PF1.TAB		
ESC.F5	ESC.F5		
PF1.'+'	PF1.'+'		
PF1.'/'	PF1.'/'		
PF1.'?'	PF1.'?'		
PF1.LEFT	PF1.LEFT		
PF1.RIGHT	PF1.RIGHT		
PF1.UP	PF1.UP		
C_Z	C_Z		
C.F5	C.F5		
ESC.AT_SIGN	ESC.AT_SIGN		
ESC.QUOTATION	ESC.QUOTATION		
ESC.Z	ESC.Z		
ESC_S_Q	ESC_S_Q		
ESC.Q	ESC.Q		
ESC_C_B	ESC_C_B		
PF2.'b'	PF2.'b'		
C_B	C_B		
C_A	C_A		
ESC_C_A	ESC_C_A		
Editor.Cursor.Up ( 8)	Editor.Cursor.Up ( 8)		
Editor.Hold_Stack.Copy_Top	Editor.Hold_Stack.Copy_Top		
Editor.Hold_Stack.Delete_Top	Editor.Hold_Stack.Delete_Top		
Editor.Hold_Stack.Next	Editor.Hold_Stack.Next		
Editor.Hold_Stack.Previous	Editor.Hold_Stack.Previous		
Editor.Hold_Stack.Push	Editor.Hold_Stack.Push		
Editor.Hold_Stack.Rotate	Editor.Hold_Stack.Rotate		
Editor.Hold_Stack.Swap	Editor.Hold_Stack.Swap		
Editor.Hold_Stack.Top	Editor.Hold_Stack.Top		
Editor.Image.Beginning_Of	Editor.Image.Beginning_Of		
Editor.Image.Down	Editor.Image.Down		
Editor.Image.End_Of	Editor.Image.End_Of		
Editor.Image.Find ( "")	Editor.Image.Find ( "")		
Editor.Image.Find ( "Help Window")	Editor.Image.Find ( "Help Window")		
Editor.Image.Find ( Name => "Nam...	Editor.Image.Find ( Name => "Nam...		
Editor.Image.Left	Editor.Image.Left		
Editor.Image.Right	Editor.Image.Right		
Editor.Image.Up	Editor.Image.Up		
Editor.Key.Name	Editor.Key.Name		
Editor.Key.Prompt	Editor.Key.Prompt		
Editor.Key.Prompt ( Key_Code => "")	Editor.Key.Prompt ( Key_Code => "")		
Editor.Line.Beginning_Of	Editor.Line.Beginning_Of		

Editor.Line.Capitalize	PF2.'B'	LINE.BEGIN_OF
Editor.Line.Center	PF2.'6'	CM_S_A
Editor.Line.Copy	PF2.'4'	CM_CIRCUMFLEX
Editor.Line.Delete	PF2.'c'	CM_S
Editor.Line.Delete_Backward	ESC.C.F	LINE.'1'
Editor.Line.Delete_Forward	PF2.'k'	LINE.'6'
Editor.Line.End_Of	PF2.'TAB	LINE.'4'
Editor.Line.Insert	PF2.'e'	LINE.'C'
Editor.Line.Indent	ESC.C.F	CM_C
Editor.Line.Join	PF2.'j'	LINE.'c'
Editor.Line.Lower_Case	PF2.'<'	CM_S_C
Editor.Line.Open	PF2.'o'	CM_S_D
Editor.Line.Transpose	ESC.C.T	LINE.'d'
Editor.Line.Upper_Case	PF2.'t'	LINE.DELETE
Editor.Macro.Blind	PF4.F4	CM_DELETE
Editor.Macro.Execute	PF4.ENTER	CM_DELETE
		CS_X
		LINE.'k'
		LINE.'k'
		C_X
		CM_F
		CM_S_F
		CM_S_F
		END_OF
		C_F
		ESC.C.F
		CM_J
		PF2.'I'
		LINE.'I'
		LINE.'I'
		C_J
		CS_J
		LINE.'j'
		M_O
		CM_O
		MS_O
		CM_S_O
		LINE.'j'
		LINE.'<'
		LINE.'<'
		CM_LESS_THAN
		CM_LESS_THAN
		C_O
		CS_O
		LINE.'o'
		LINE.'O'
		CM_T
		LINE.'t'
		LINE.'T'
		CM_S_T
		CM_GREATER_THAN
		LINE.'>'
		LINE.'>'
		CM_PERIOD
		LINE.'>'
		MARK.F10
		M_X

REGION. '{'  
 C\_LEFT\_BRACE  
 REGION. '='  
 REGION. '+'  
 REGION. '.'  
 REGION. '>'  
 M\_L  
 MS\_L  
 CS\_DOWN  
 CS\_LEFT  
 CMS\_RIGHT  
 CMS\_LEFT  
 CMS\_DOWN  
 CS\_L  
 CS\_RIGHT  
 CMS\_UP  
 CMS\_BEGIN\_OF  
 CS\_UP  
 CS\_S  
 CS\_S  
 CS\_S  
 CS\_R  
 C\_J  
 M\_S  
 MS\_S  
 M\_S  
 MS\_R  
 M\_NUMERIC\_0  
 M\_NUMERIC\_1  
 M\_NUMERIC\_2  
 M\_NUMERIC\_3  
 M\_NUMERIC\_4  
 M\_NUMERIC\_5  
 M\_NUMERIC\_6  
 M\_NUMERIC\_7  
 M\_NUMERIC\_8  
 C\_NUMERIC\_0  
 C\_NUMERIC\_1  
 C\_NUMERIC\_2  
 C\_NUMERIC\_3  
 C\_NUMERIC\_4  
 C\_NUMERIC\_5  
 C\_NUMERIC\_6  
 C\_NUMERIC\_7  
 C\_NUMERIC\_8

MARK.ENTER  
 MARK.PROMOT  
 MARK.CARRIAGE\_RETURN  
 MS\_X  
 MARK. '}'  
 M\_RIGHT\_BRACE  
 MARK. '{'  
 M\_LEFT\_BRACKET  
 MARK.END\_OF  
 M\_RIGHT\_BRACKET  
 M\_LEFT\_BRACKET  
 MARK. '['  
 M\_LEFT\_BRACE  
 MARK. '{'  
 MARK.BEGIN\_OF  
 MARK. 'p'  
 MARK. 'P'  
 MARK.DELETE  
 MS\_M  
 M\_M  
 MARK.RIGHT  
 MARK.LEFT  
 MARK.DOWN  
 CS\_M  
 C\_M  
 MARK. 'r'  
 MARK. 't'  
 MARK. 'i'  
 MARK.UP  
 REGION.BEGIN\_OF  
 REGION. '6'  
 REGION. '-'  
 REGION. ' '  
 REGION. 'c'  
 REGION. 'C'  
 REGION. 'k'  
 REGION. 'K'  
 REGION. 'd'  
 REGION. 'D'  
 REGION.END\_OF  
 REGION.FORMAT  
 REGION. ']'  
 C\_RIGHT\_BRACKET  
 C\_RIGHT\_BRACE  
 REGION. '}'  
 REGION.COMPLT  
 REGION. '<'  
 REGION. 'M'  
 REGION. 'm'  
 REGION. 'x'  
 REGION. 'X'  
 C\_LEFT\_BRACKET  
 REGION. '['  
 PFA.X4  
 ESC.S\_X  
 ESC.X  
 PFA. 'e'  
 PFA. 'E'  
 PFA. '}'  
 PFA.END\_OF  
 PFA. ']'  
 PFA.BEGIN\_OF  
 PFA. 'b'  
 PFA. 'B'  
 PFA. '{'  
 PFA. 'P'  
 ESC.M  
 PFA.RIGHT  
 ESC.S\_M  
 PFA.LEFT  
 NUL  
 PFA.DOWN  
 PFA.UP  
 X2. 'b'  
 X2. 'B'  
 X2.BEGIN\_OF  
 X2. '6'  
 X2. '-'  
 X2. ' '  
 X2. 'c'  
 X2. 'C'  
 X2. 'd'  
 X2. 'D'  
 X2. 'k'  
 X2. 'K'  
 X2. 'e'  
 X2. 'E'  
 X2.END\_OF  
 X2. 'e'  
 X2.X6  
 X2. '}'  
 X2. ']'  
 ESC.RIGHT\_BRACE  
 X2.X5  
 X2. '<'  
 X2. 'm'  
 X2. 'M'  
 X2. 'x'  
 X2. 'X'  
 ESC.LEFT\_BRACE  
 X2. '['

Editor.Macro.Finish  
 Editor.Macro.Start  
 Editor.Mark.Copy\_Top  
 Editor.Mark.Delete\_Top  
 Editor.Mark.Next  
 Editor.Mark.Previous  
 Editor.Mark.Push  
 Editor.Mark.Rotate  
 Editor.Mark.Swap  
 Editor.Mark.Top  
 Editor.Region.Beginning\_Of  
 Editor.Region.Capitalize  
 Editor.Region.Comment  
 Editor.Region.Copy  
 Editor.Region.Delete  
 Editor.Region.End\_Of  
 Editor.Region.Fill  
 Editor.Region.Finish  
 Editor.Region.Justify  
 Editor.Region.Lower\_Case  
 Editor.Region.Move  
 Editor.Region.Off  
 Editor.Region.Start

S\_NUMERIC\_8  
 M\_NUMERIC\_8  
 NUMERIC\_8  
 M\_NUMERIC\_9  
 S\_NUMERIC\_9  
 NUMERIC\_9  
 C\_NUMERIC\_9  
 DASH  
 S\_DASH  
 M\_DASH  
 NUMERIC\_COMMA  
 M\_NUMERIC\_COMMA  
 C\_NUMERIC\_COMMA  
 S\_NUMERIC\_COMMA  
 CS\_X  
 C\_F17  
 C\_X  
 IMAGE.'X'  
 IMAGE.'x'  
 IMAGE.'f'  
 IMAGE.'F'  
 IMAGE.'o'  
 IMAGE.'O'  
 IMAGE.'i'  
 IMAGE.'I'  
 WINDOW\_BEGIN\_OF  
 CM\_BEGIN\_OF  
 X3.'b'  
 X3.'B'  
 X3.RIGHT  
 X3.'c'  
 X3.'C'  
 X3.'d'  
 X3.'D'  
 X3.'x'  
 X3.'X'  
 X3.'k'  
 X3.'K'  
 X3.'d'  
 X3.'D'  
 X3.'x'  
 X3.'X'  
 X3.F7  
 X3.S\_F7  
 X3.'/'  
 X3.F4  
 X3.'+'  
 X3.'?'  
 X3.'e'  
 X3.END\_OF  
 X3.'e'  
 X3.'l'  
 X3.'L'  
 X3.'l'  
 X3.'l'  
 X3.'l'  
 X3.X6  
 X3.DELETE  
 X3.'J'  
 X3.'j'  
 X3.'j'  
 ESC.V  
 S\_DOWN  
 X3.DOWN  
 ESC.S.V  
 X3.LEFT  
 Editor.Set.Argument\_Digit ( 9)  
 Editor.Set.Argument\_Minus  
 Editor.Set.Argument\_Prefix  
 Editor.Set.Designation\_Off  
 Editor.Set.Fill\_Mode ( False)  
 Editor.Set.Fill\_Mode ( True)  
 Editor.Set.Insert\_Mode ( False)  
 Editor.Set.Insert\_Mode ( True)  
 Editor.Window.Beginning\_Of  
 Editor.Window.Child  
 Editor.Window.Copy  
 Editor.Window.Delete  
 Editor.Window.Demote  
 Editor.Window.Directory  
 Editor.Window.End\_Of  
 Editor.Window.Expand  
 Editor.Window.Expand ( - ( 4))  
 Editor.Window.Focus  
 Editor.Window.Join ( - ( 1))  
 Editor.Window.Join ( 1)  
 Editor.Window.Next  
 Editor.Window.Parent  
 NUMERIC\_9  
 DASH  
 NUMERIC\_COMMA  
 C\_X  
 PF1.'x'  
 PF1.'x'  
 PF1.'f'  
 PF1.'F'  
 PF1.'o'  
 PF1.'O'  
 PF1.'i'  
 PF1.'I'  
 X3.BEGIN\_OF  
 X3.'B'  
 X3.RIGHT  
 X3.'c'  
 X3.'C'  
 X3.'d'  
 X3.'D'  
 X3.'x'  
 X3.'X'  
 X3.'k'  
 X3.'K'  
 X3.'d'  
 X3.'D'  
 X3.'x'  
 X3.'X'  
 X3.F7  
 X3.S\_F7  
 X3.'/'  
 X3.F4  
 X3.'+'  
 X3.'?'  
 X3.'e'  
 X3.END\_OF  
 X3.'e'  
 X3.'l'  
 X3.'L'  
 X3.'l'  
 X3.'l'  
 X3.X6  
 X3.DELETE  
 X3.'J'  
 X3.'j'  
 X3.'j'  
 ESC.V  
 S\_DOWN  
 X3.DOWN  
 ESC.S.V  
 X3.LEFT

Editor.Window.Previous  
 Editor.Window.Promote  
 Editor.Window.Transpose  
 Editor.Word.Beginning\_Of  
 Editor.Word.Capitalize  
 Editor.Word.Delete  
 Editor.Word.Delete\_Backward  
 Editor.Word.Delete\_Forward  
 Editor.Word.End\_Of  
 Editor.Word.Lower\_Case  
 Editor.Word.Next  
 Editor.Word.Previous  
 Editor.Word.Transpose  
 Editor.Word.Upper\_Case  
 Job.Connect ( 0)  
 Job.Disable ( 0)  
 Job.Enable ( 0)  
 ESC.S.Z  
 ESC.Z  
 S\_UP  
 X3.UP  
 X3.X4  
 X3.'t'  
 X3.'t'  
 PF3.BEGIN\_OF  
 ESC.A  
 PF3.'b'  
 PF3.'b'  
 ESC.S.B  
 ESC.S.A  
 ESC.B  
 PF3.'6'  
 PF3.'-'  
 PF3.'-'  
 ESC.TILDE  
 ESC.6  
 ESC.CIRCUMFLEX  
 PF3.'d'  
 ESC.S.D  
 PF3.'d'  
 ESC.D  
 PF3.DELETE  
 ESC.DEL  
 ESC.K  
 PF3.'k'  
 ESC.S.K  
 PF3.'K'  
 PF3.TAB  
 ESC.S.I  
 PF3.'e'  
 ESC.F  
 PF3.'e'  
 PF3.END\_OF  
 PF3.'<'  
 ESC.LESS\_THAN  
 PF3.RIGHT  
 ESC.S.J  
 ESC.J  
 PF3.LEFT  
 ESC.S.H  
 ESC.H  
 ESC.S.T  
 WORD.'t'  
 PF3.'t'  
 ESC.T  
 PF3.'>'  
 ESC.GREATER\_THAN  
 ESC.F11  
 ESC.C.F11  
 S\_F11  
 WINDOW.UP  
 M\_Z  
 MS\_Z  
 CM\_UP  
 WINDOW.PROMOT  
 WINDOW.'t'  
 WINDOW.'t'  
 M\_A  
 WORD.BEGIN\_OF  
 MS\_A  
 M\_BEGIN\_OF  
 MS\_B  
 M\_B  
 WORD.'6'  
 M\_6  
 M\_CIRCUMFLEX  
 WORD.'-'  
 MS\_D  
 M\_D  
 WORD.'d'  
 WORD.'D'  
 M\_DELETE  
 WORD.DELETE  
 MS\_DELETE  
 MS\_K  
 WORD.'k'  
 M\_K  
 WORD.'K'  
 M\_E  
 WORD.END\_OF  
 MS\_E  
 M\_END\_OF  
 M\_LESS\_THAN  
 WORD.'<'  
 M\_COMMA  
 WORD.'.'  
 M\_J  
 M\_RIGHT  
 WORD.RIGHT  
 MS\_J  
 M\_LEFT  
 WORD.LEFT  
 M\_H  
 MS\_H  
 WORD.'t'  
 WORD.'T'  
 MS\_T  
 M\_T  
 WORD.'.'  
 WORD.'>'  
 M\_GREATER\_THAN  
 M\_PERIOD  
 M\_F19  
 F19  
 S\_F19

```

Job.Interrupt
Job.Kill ( 0)

Library.Create_Directory ( Name => "" )
Library.Create_World ( Name => "" )
Queue.Print
Text.Create ( Image_Name => "" )
Text.End_Of_Input

What.Does ( "" )
What.Does ( "Help_On_Help" )
What.Home_Library
What.Line

What.Load ( Verbose => True )
What.Locks ( Name => "<image>" )
What.Object
What.Tabs

What.Time

C_G
CS_G
C_F19
M_G
MS_G
M_F15
CS_F15
CM_F11
CM_F15
DOT
CM_F19
F11
S_F11
CM_F10
LINE.'/'
LINE.'?'
S_F20
M_F20
CM_F20
CMS_TAB
CM_TAB
F20

C_G
ESC_G
C_F11
ESC_S_G
ESC_G
ESC_F8
ESC_C_F8
F11
C_F8
DOT
S_F5
F5
ESC_UP
PF2.'?'
PF2.'+'
PF2.'/'
S_F12
ESC_C_F12
C_F12
F12

```

## Master Reference to Key Bindings by Key

```

with Access_List;
with Ada;
with Command;
with Common;
with Compilation;
with Daemon;
with Debug;
with Editor;
with File_Utillities;
with Job;
with Library;
with Operator;
with Profile;
with Queue;
with Text;
with Visible_Key_Names;
with What;

procedure Facit_Commands is

```

```

-- Updated for Delta under A.9.5.0 by KR, 3/28/87
-- Updated for Delta under A.9.16.0 for LIB, 6/24/87

```

```

use Visible_Key_Names;

```

```

type Intent is (Prompt, Execute, Interrupt);

```

```

Action : Intent;

```

```

Key_1 : Facit_Key_Names;

```

```

Key_2 : Facit_Key_Names;

```

```

-- Interrupt and Prompt are first because they are shorter and can be moved
-- around very easily. Each of the keys is replicated in comments in the
-- main Execute case statement.

```

```

begin
case Action is
when Interrupt =>
case Key_1 is
when C_G =>
Job.Interrupt;
when Esc_D =>
Job.Kill (0);
when C_F1 =>
Debug.Stop;
when Esc_C_F11 =>
Job.Disable (0);
when C_F11 =>
Job.Kill (0);
when others =>
null;
end case;

```

```

when Prompt =>
case Key_1 is
when Esc_C_F3 =>
Debug.Modify (New_Value => "",
Variable => "<SELECTION>",
Stack_Frame => 0);
when S_F5 =>
What.Does ("");
when Esc_F8 =>
Library.Create_Directory (Name => "");
when C_F8 =>
Text.Create (Image_Name => "");
when Esc_C_F8 =>
Library.Create_World (Name => "");
when P_f1 =>
case Key_2 is
when '?' | '/' | '+' =>
Editor.Image.Find (Name => "name or name fragment");
when others =>
null;
end case;
when others =>
null;
end case;
when Execute =>
case Key_1 is
when X4 =>
Common.Promote;
when S_X4 =>
Command.Spawn;
when X5 =>
Common.Complete;
when X6 =>
Common.Format;
when C_A =>
Editor.Line.Beginning_Of;
when C_B =>
Editor.Line.Beginning_Of;
when C_C =>
Editor.Hold_Stack.Push;
when C_D =>
Editor.Char.Delete_Forward;
when C_E =>
Editor.Line.End_Of;
when C_F =>
Editor.Search.Next;
when C_G =>
Job.Interrupt;
when C_H =>
Editor.Cursor.Left;
when C_I =>
Editor.Char.Tab_Forward;

```

```

when C_J =>
  Editor.Mark.Next;
when Esc_N | Esc_S_N =>
  Editor.Cursor.Next;
when Esc_O | Esc_S_O =>
  Editor.Line.Join;
when Esc_Q | Esc_S_Q =>
  Editor.Key.Prompt;
when Esc_R | Esc_S_R =>
  Editor.Search.Replace.Previous;
when Esc_T | Esc_S_T =>
  Editor.Word.Transpose;
when Esc_U | Esc_S_U =>
  Editor.Cursor.Previous;
when Esc_V | Esc_S_V =>
  Editor.Window.Next;
when Esc_X | Esc_S_X =>
  Editor.Macro.Execute;
when Esc_Y | Esc_S_Y =>
  Editor.Hold_Stack.Next;
when Esc_Z | Esc_S_Z =>
  Editor.Window.Previous;

when Esc_Circumflex | Esc_6 | Esc_Filde =>
  Editor.Word.Capitalize;
when Esc_Greater_Than =>
  Editor.Word.Upper_Case;
when Esc_Less_Than =>
  Editor.Word.Lower_Case;

when Esc_2 | Esc_At_Sign | Esc_Quotation =>
  Editor.Key.Name;
when Esc_C_A | Esc_C_B =>
  Editor.Line.Beginning_Of;
when Esc_C_C =>
  Editor.Line.Copy;
when Esc_C_D =>
  Editor.Line.Delete;
when Esc_C_E =>
  Editor.Line.End_Of;
when Esc_C_H =>
  Editor.Cursor.Left (8);
when Esc_C_I =>
  Editor.Char.Tab_Backward;
when Esc_C_J =>
  Editor.Cursor.Right (8);
when Esc_C_K =>
  Editor.Line.Delete_Forward;
when Esc_C_M =>
  Command.Debug;
when Esc_C_N =>
  Editor.Cursor.Down (8);
when Esc_C_O =>
  Editor.Line.Join;
when Esc_C_T =>
  Editor.Line.Transpose;
when Esc_C_U =>
  Editor.Cursor.Up (8);
when Esc_Up =>
  What_Home_Library;

```

```

when C_J =>
  Editor.Cursor.Right;
when C_K =>
  Editor.Line.Delete_Forward;
when C_L =>
  Editor.Screen.Redraw;
when C_M =>
  Editor.Line.Indent;
when C_N =>
  Editor.Cursor.Down;
when C_O =>
  Editor.Line.Open;
when C_R =>
  Editor.Search.Previous;
when C_T =>
  Editor.Char.Transpose;
when C_U =>
  Editor.Cursor.Up;
when C_V =>
  Editor.Image.Down;
when C_X =>
  Editor.Set.Designation_Off;
when C_Y =>
  Editor.Hold_Stack.Top;
when C_Z =>
  Editor.Image.Up;
when Null =>
  Editor.Mark.Push;

when Esc_Left_Brace =>
  Editor.Region.Start;
when Esc_Right_Brace =>
  Editor.Region.Finish;

when Esc_Query | Esc_Slash | Esc_Plus =>
  Common.Explain;
when Esc_Tick | Esc_Star =>
  Editor.Char.Quote;

when Esc_A | Esc_B | Esc_C | Esc_D | Esc_E | Esc_F | Esc_G | Esc_H | Esc_I | Esc_J | Esc_K | Esc_L | Esc_M | Esc_N | Esc_O | Esc_P | Esc_Q | Esc_R | Esc_S | Esc_T | Esc_U | Esc_V | Esc_W | Esc_X | Esc_Y | Esc_Z =>
  Editor.Word.Beginning_Of;
when Esc_C | Esc_S_C =>
  Editor.Hold_Stack.Next;
when Esc_D | Esc_S_D =>
  Editor.Word.Delete;
when Esc_E | Esc_S_E =>
  Editor.Word.End_Of;
when Esc_F | Esc_S_F =>
  Editor.Search.Replace.Next;
when Esc_G | Esc_S_G =>
  Job.Kill (0);
when Esc_H | Esc_S_H =>
  Editor.Word.Previous;
when Esc_J | Esc_S_J =>
  Editor.Word.Next;
when Esc_K | Esc_S_K =>
  Editor.Word.Delete_Forward;
when Esc_L | Esc_S_L =>
  Editor.Screen.Clear;
when Esc_M | Esc_S_M =>

```

```

when Delete =>
  Editor.Char.Delete_Backward;
when Esc_Del =>
  Editor.Word.Delete_Backward;
when Esc_Backslash =>
  Editor.Char.Delete_Spaces;
when Esc_C_F =>
  Editor.Line.Delete_Backward;

when Enter =>
  Common.Commit;

-- F1 - F3 Debugger Control
-- F1 Debugger Execution Control
when F1 =>
  Debug.Run;
when S_F1 =>
  Debug.Execute;
when Esc_F1 =>
  Debug.Run (Stop_At => Debug.Local_Statement);
-- when C_F1 => Debug.Stop; -- Interrupts
when Esc_C_F1 =>
  Debug.Task_Display;

-- F2 Debugger source and breakpoints
when F2 =>
  Debug.Source;
when S_F2 =>
  Debug.Break;
when Esc_F2 =>
  Debug.Activate (Breakpoint => 0);
when C_F2 =>
  Debug.Remove (Breakpoint => 0);
when Esc_C_F2 =>
  Debug.Show;

-- F3 Debugger stack, exceptions and values
when F3 =>
  Debug.Put;
when S_F3 =>
  Debug.Catch;
when Esc_F3 =>
  Debug.Propagate;
when C_F3 =>
  Debug.Stack;
-- when Esc_C_F3 => Debug.Modify; -- Prompt

-- F4 Definition and traversal
when F4 =>
  Common.Definition
    (Name => "<CURSOR>", In_Place => False, Visible => True);
when S_F4 =>
  Common.Definition
    (Name => "<CURSOR>", In_Place => True, Visible => False);

```

```

when Esc_F4 =>
  Common.Enclosing (In_Place => False, Library => True);
when C_F4 =>
  Common.Enclosing;
when Esc_C_F4 =>
  Common.Enclosing (In_Place => True, Library => False);

-- F5 Help and access list display
when F5 =>
  What.Does ("Help_On_Help");
when S_F5 => What.Does (""); -- Prompt
when Esc_F5 =>
  Editor.Image.Find ("Help Window");
when C_F5 =>
  Editor.Key.Name;
when Esc_C_F5 =>
  Access_List.Display;

-- F6 Ada promotion
when F6 =>
  Ada.Install_Unit;
when S_F6 =>
  Ada.Code_Unit;
when Esc_F6 =>
  Compilation.Promote (Unit => "<Image>",
    Scope => Compilation.All_Parts,
    Goal => Compilation.Installed,
    Limit => "<WORLDS>",
    Effort_Only => False,
    Response => "<PROFILE>");
when C_F6 =>
  Compilation.Make (Unit => "<IMAGE>",
    Scope => Compilation.All_Parts,
    Goal => Compilation.Coded,
    Limit => "<WORLDS>",
    Effort_Only => False,
    Response => "<PROFILE>");

when Esc_C_F6 =>
  Compilation.Make (Unit => "<IMAGE>",
    Scope => Compilation.Load_Views,
    Goal => Compilation.Coded,
    Limit => "<ALL_WORLDS>",
    Effort_Only => False,
    Response => "<PROFILE>");

-- F7 Ada edit/demotion
when F7 =>
  Common.Edit;
when S_F7 =>
  Common.Demote;
when Esc_F7 =>
  Compilation.Demote (Unit => "<SELECTION>",
    Goal => Compilation.Source,
    Limit => "<WORLDS>",
    Effort_Only => False,

```

```

-- when Esc_C_F11 => Job.Disable (0); -- Interrupt
-- when C_F11 => Job.Kill (0); -- Interrupt

-- -- F12 Information

when F12 =>
  What.Time;
when S_F12 =>
  What.Load (Verbose => True);
when Esc_F12 =>
  What.Users (All_Users => True);
when C_F12 =>
  What.Object;
when Esc_C_F12 =>
  What.Locks (Name => "<Image>");

when Up =>
  Editor.Cursor.Up;
when S_Up =>
  Editor.Window.Previous;

when Down =>
  Editor.Cursor.Down;
when S_Down =>
  Editor.Window.Next;

when Right =>
  Editor.Cursor.Right;

when Left =>
  Editor.Cursor.Left;

-- PF3 => Word
when PF3 =>
  case Key_2 is
    when '<' =>
      Editor.Word.Lower_Case;
    when '>' =>
      Editor.Word.Upper_Case;
    when 'd' | 'D' =>
      Editor.Word.Delete;
    when 'k' | 'K' =>
      Editor.Word.Delete_Forward;
    when 't' | 'T' =>
      Editor.Word.Transpose;
    when '-' | '6' | '.' =>
      Editor.Word.Capitalize;
    when Delete =>
      Editor.Word.Delete_Backward;
    when End_Of | 'e' | 'Z' | Tab =>
      Editor.Word.End_Of;
    when Begin_Of | 'b' | 'B' =>
      Editor.Word.Beginning_Of;
    when Left =>
      Editor.Word.Previous;
    when Right =>
      Editor.Word.Next;
    when others =>
      null;
  end case;

```

```

Response => "<PROFILE>";

when C_F7 =>
  Ada.Source_Unit;
when Esc_C_F7 =>
  Ada.Withdraw;

-- -- F8 Creations

when F8 =>
  Common.Create_Command;
when S_F8 =>
  Ada.Create_Body;

-- -- F9 Ada traversal and show usage

when Esc_F8 => Library.Create_Directory; -- Prompt
when Esc_C_F8 => Library.Create_World; -- Prompt
when C_F8 => Text.Create; -- Prompt

when F9 =>
  Ada.Other_Part (Name => "<IMAGE>", In_Place => False);
when S_F9 =>
  Ada.Other_Part (Name => "<IMAGE>", In_Place => True);

when Esc_F9 =>
  Ada.Show_Usage (Name => "<CURSOR>",
    Global => False,
    Limit => "<WORLDS>",
    Closure => False);

when Esc_C_F9 =>
  Ada.Create_Private;
when C_F9 =>
  Ada.Show_Usage (Name => "<CURSOR>",
    Global => True,
    Limit => "<ALL_WORLDS>",
    Closure => False);

-- -- F10 Ada semanticize, underlinings, and show unused

when F10 =>
  Common.Semanticize;
when S_F10 =>
  Ada.Get_Errors;
when Esc_F10 =>
  Common.Clear_Underlining;
when C_F10 =>
  Ada.Show_Unused (In_Unit => "<IMAGE>",
    Check_Other_Units => False);

when Esc_C_F10 =>
  Ada.Show_Unused (In_Unit => "<IMAGE>",
    Check_Other_Units => True);

-- -- F11 Jobs and print

when F11 =>
  Queue.Print;
when S_F11 =>
  Job.Enable (0);
when Esc_F11 =>
  Job.Connect (0);

```



```

end case;
-- PF2 => Line
  when Pfl2 =>
    case Key_2 is
      when '<' =>
        Editor.Line.Lower_Case;
      when '>' =>
        Editor.Line.Upper_Case;
      when '?' | '/' | ',' =>
        What.Line;
      when Begin_Of | 'b' | 'B' =>
        Editor.Line.Beginning_Of;
      when 'c' | 'C' =>
        Editor.Line.Copy;
      when 'd' | 'D' =>
        Editor.Line.Delete;
      when End_Of | 'e' | 'E' | Tab =>
        Editor.Line.End_Of;
      when 'i' | 'I' =>
        Editor.Line.Insert;
      when 'j' | 'J' =>
        Editor.Line.Join;
      when 'k' | 'K' =>
        Editor.Line.Delete_Forward;
      when 'o' | 'O' =>
        Editor.Line.Open;
      when 't' | 'T' =>
        Editor.Line.Transpose;
      when '-' | '6' | '!' =>
        Editor.Line.Capitalize;
      when Delete =>
        Editor.Line.Delete_Backward;
      when 'g' | '4' =>
        Editor.Line.Center;
      when others =>
        null;
    end case;
-- X2 => Region/Hold_Stack
  when X2 =>
    case Key_2 is
      when 'x' | 'X' =>
        Editor.Region.Off;
      when '<' =>
        Editor.Region.Lower_Case;
      when '>' =>
        Editor.Region.Upper_Case;
      when 'c' | 'C' =>
        Editor.Region.Copy;
      when 'd' | 'D' | 'k' | 'K' =>
        Editor.Region.Delete;
      when 'm' | 'M' =>
        Editor.Region.Move;
      when '[' | '{' =>
        Editor.Region.Start;
      when ']' | '}' =>
        Editor.Region.Finish;
      when '-' | '6' | '!' =>
        Editor.Region.Capitalize;
    end case;

```

```

when End_Of | 'e' | 'E' =>
  Editor.Region.End_Of;
when Begin_Of | 'b' | 'B' =>
  Editor.Region.Beginning_Of;
when Left =>
  Editor.Hold_Stack.Previous;
when Right =>
  Editor.Hold_Stack.Next;
when Up =>
  Editor.Hold_Stack.Top;
when Down =>
  Editor.Hold_Stack.Push;
when X6 =>
  Editor.Region.Fill;
when X5 =>
  Editor.Region.Justify;
when others =>
  null;
end case;
-- X3 => Window
  when X3 =>
    case Key_2 is
      when '|' | '1' =>
        Editor.Window.Expand;
      when '.' =>
        Editor.Window.Expand (-4);
      when '?' | '/' | F4 | '+' =>
        Editor.Window.Directory;
      when 'c' | 'C' =>
        Editor.Window.Copy;
      when 'd' | 'D' | 'k' | 'g' =>
        Editor.Window.Delete;
      when 'j' | 'J' =>
        Editor.Window.Join (1);
      when 't' | 'T' =>
        Editor.Window.Transpose;
      when 'x' | 'X' =>
        Editor.Window.Delete;
      when Delete =>
        Editor.Window.Join (-1);
      when End_Of | 'e' | 'E' =>
        Editor.Window.End_Of;
      when Down =>
        Editor.Window.Next;
      when X6 =>
        Editor.Window.Focus;
      when X4 =>
        Editor.Window.Promote;
      when S_F7 =>
        Editor.Window.Demote;
      when F7 =>
        Editor.Window.Demote;
      when Begin_Of | 'b' | 'B' =>
        Editor.Window.Beginning_Of;
      when Left =>
        Editor.Window.Parent;
      when Right =>
        Editor.Window.Child;
      when Up =>

```

```

Editor.Window.Previous;
when others =>
  null;
end case;

--- PF1 => Image
when Pf1 =>
  case Key_2 is
    when End_Of | 'e' | 'p' | Tab =>
      Editor.Image.End_Of;
    when Down =>
      Editor.Image.Down;
    when Begin_Of | 'b' | 'B' =>
      Editor.Image.Beginning_Of;
    when Left =>
      Editor.Image.Left;
    when Right =>
      Editor.Image.Right;
    when Up =>
      Editor.Image.Up;
    when 'i' | 'I' =>
      Editor.Set.Insert_Mode (True);
    when 'o' | 'O' =>
      Editor.Set.Insert_Mode (False);
    when 'f' | 'F' =>
      Editor.Set.Fill_Mode (True);
    when 'X' | 'x' =>
      Editor.Set.Fill_Mode (False);
    when others =>
      null;
  end case;

--- PF4 => Mark
when Pf4 =>
  case Key_2 is
    when Down =>
      Editor.Mark.Push;
    when Right =>
      Editor.Mark.Next;
    when Left =>
      Editor.Mark.Previous;
    when Up =>
      Editor.Mark.Top;
    when Begin_Of | 'b' | 'B' | '[' | '{' =>
      Editor.Macro.Start;
    when End_Of | 'e' | 'E' | ']' | '}' =>
      Editor.Macro.Finish;
    when X4 | Enter =>
      Editor.Macro.Execute;
    when F4 =>
      Editor.Macro.Bind;
    when others =>
      null;
  end case;

when Numeric_0 =>
  Editor.Set.Argument_Digit (0);
when Numeric_1 =>
  Editor.Set.Argument_Digit (1);
when Numeric_2 =>

```

```

Editor.Set.Argument_Digit (2);
when Numeric_3 =>
  Editor.Set.Argument_Digit (3);
when Numeric_4 =>
  Editor.Set.Argument_Digit (4);
when Numeric_5 =>
  Editor.Set.Argument_Digit (5);
when Numeric_6 =>
  Editor.Set.Argument_Digit (6);
when Numeric_7 =>
  Editor.Set.Argument_Digit (7);
when Numeric_8 =>
  Editor.Set.Argument_Digit (8);
when Numeric_9 =>
  Editor.Set.Argument_Digit (9);
when Numeric_Comma =>
  Editor.Set.Argument_Prefix;
when Dash =>
  Editor.Set.Argument_Minus;
when Dot =>
  Text.End_Of_Input;

--- X1 => Object
when X1 =>
  case Key_2 is
    when 'i' | 'I' =>
      Common.Expand;
    when '.' =>
      Common.Elide;
    when '/' | '?' | '+' =>
      Common.Explain;
    when 'c' | 'C' =>
      Common.Object.Copy;
    when 'd' | 'D' | 'k' | 'K' =>
      Common.Object.Delete;
    when 'g' | 'G' =>
      Common.Abandon;
    when 'l' | 'L' =>
      Common.Object.Insert;
    when 'm' | 'M' =>
      Common.Revert;
    when 'x' | 'X' =>
      Common.Object.Move;
    when 'r' | 'R' =>
      Common.Redo;
    when 'u' | 'U' =>
      Common.Undo;
    when Down =>
      Common.Object.Next;
    when Left =>
      Common.Object.Parent;
    when Right =>
      Common.Object.Child;
    when Up =>
      Common.Object.Previous;
    when Begin_Of | 'b' | 'B' =>
      Common.Object.First_Child;

```

```
when End_Of | 'e' | 'z' =>  
  Common.Object.Last_Child;  
when F4 =>  
  Common.Definition;  
when others =>  
  null;  
end case;  
  
when others =>  
  null;  
end case;  
  
end case;  
end Facit_Commands;
```



# RATIONAL

## READER'S COMMENTS

**Note:** This form is for documentation comments only. You can also submit problem reports and comments electronically by using the SIMS problem-reporting system. If you use SIMS to submit documentation comments, please indicate the manual name, book name, and page number.

Did you find this book understandable, usable, and well organized? Please comment and list any suggestions for improvement.

---

---

---

---

---

If you found errors in this book, please specify the error and the page number. If you prefer, attach a photocopy with the error marked.

---

---

---

---

Indicate any additions or changes you would like to see in the index.

---

---

---

---

How much experience have you had with the Rational Environment?

6 months or less \_\_\_\_\_ 1 year \_\_\_\_\_ 3 years or more \_\_\_\_\_

How much experience have you had with the Ada programming language?

6 months or less \_\_\_\_\_ 1 year \_\_\_\_\_ 3 years or more \_\_\_\_\_

Name (optional) \_\_\_\_\_ Date \_\_\_\_\_  
Company \_\_\_\_\_  
Address \_\_\_\_\_  
City \_\_\_\_\_ State \_\_\_\_\_ ZIP Code \_\_\_\_\_

**Please return this form to:**

**Publications Department  
Rational  
1501 Salado Drive  
Mountain View, CA 94043**

*Rational Environment Reference Manual, Keymap: Facit Terminal, 8001A-51*



Rational Environment  
Reference Manual

Master Index: Environment

Copyright © 1987, 1988 by Rational

Document Control Number: 8001A-21 (803-002304)

Rev. 1.0, July 1987

Rev. 2.0, September 1988 (Software Rev. D\_10\_20\_0)

This document is subject to change without notice.

Note the Reader's Comments forms at the end of this book, which request the user's evaluation to assist Rational in preparing future documentation.

Rational and R1000 are registered trademarks and Rational Environment and Rational Subsystems are trademarks of Rational.

Rational  
3320 Scott Boulevard  
Santa Clara, California 95054-3197



# Master Index

The Master Index combines the indexes for Volumes 2–11 in the *Rational Environment Reference Manual*. Thus, the prefix to the page number is the abbreviation for the book in which the item can be found. This index contains information on key concepts as well as entries for each unit and its declarations, exceptions, errors, enumerations, pragmas, switches, and so on. The entries for each unit are arranged alphabetically by simple name. The first indented item under the simple name is the pathname; second-level indentions, if any, list units in which cross-references appear. An italic page number indicates the primary reference for an entry. Units that are not documented separately as reference entries, but are cross-referenced, appear under their full pathnames (see the ! section at the beginning of the index).

!(exclamation mark)	
special character . . . . .	DEB-18, LM-10, LM-11, PM-131, SJM-8, SMU-5
!Io.Device_Independent_Io package . . . . .	DIO-3, TIO-3
Io.Convert function . . . . .	TIO-17
!Io.Device_Independent_Io.File_Type type	
Io.Convert function . . . . .	TIO-17
Io.Convert procedure . . . . .	TIO-18
!Io.Terminal_Specific package . . . . .	DIO-3, TIO-3
!Machine.Archive_Mappings file . . . . .	LM-106
!Machine.Devices library . . . . .	SMU-227, SMU-250, SMU-269
!Machine.Devices.Terminal_n device . . . . .	DIO-3, SMU-250, SMU-269, TIO-3
!Machine.Editor_Data directory	
Editor.Key package . . . . .	EI-28
!Machine.Editor_Data.Daily_Message file	
What.Message procedure . . . . .	SJM-267
!Machine.Editor_Data.Rational_Commands procedure	
Editor.Key package . . . . .	EI-28
!Machine.Editor_Data.Rational_User_Commands procedure	
Editor.Key package . . . . .	EI-27, EI-28
!Machine.Editor_Data.Terminal_Recognition file . . . . .	SMU-300
!Machine.Editor_Data.Terminal_Types file . . . . .	SMU-299
!Machine.Editor_Data.Visible_Key_Names package . . . . .	SMU-300
Window_Io package . . . . .	DIO-85
Window_Io.Image function . . . . .	DIO-179
Window_Io.Raw.Value function . . . . .	DIO-188
Window_Io.Raw.Value procedure . . . . .	DIO-190
!Machine.Error_Logs world . . . . .	SMU-12, SMU-16
!Machine.Initialize procedure . . . . .	SMU-13, SMU-131
What.Users procedure . . . . .	SJM-273
!Machine.Operator_Capability file . . . . .	LM-20, LM-74, SMU-54

!Machine.Release.Current.Activity . . . . .	PM-80, PM-83
!Machine.Release.Current.Commands.Login . . . . .	PM-83
!Machine.Search_Lists.Default file	
Search_List package . . . . .	SJM-209
Search_List.Reset_To_System_Default procedure . . . . .	SJM-220
!Machine.User_Acl_Suffix file . . . . .	SMU-55, SMU-64
!Machine.User_Default_Acl_Suffix file . . . . .	SMU-55, SMU-64
!Machine.Users username . . . . .	SMU-64
!Machine.Users world . . . . .	SMU-67
!Model.R1000 . . . . .	PM-22
!Model.R1000 world . . . . .	SMU-64
!Model.R1000_Implementation . . . . .	PM-22
!Model.R1000_Portable . . . . .	PM-22
!Tools.Disk_Daemon package	
Daemon package . . . . .	SMU-11
!Tools.Disk_Daemon.Set_Backup_Killing procedure	
System_Backup.Backup procedure . . . . .	SMU-192
# (pound sign)	
library wildcard . . . . .	LM-8, LM-109, LM-113, LM-297, LM-301, PM-129, SJM-6, SMU-3
substitution character . . . . .	LM-10, PM-130, SJM-7, SMU-4
symbol in window banner . . . . .	PM-135, PM-404
\$ (dollar sign)	
file utilities wildcard . . . . .	LM-172, LM-181, LM-184, LM-187
metacharacter . . . . .	EI-56
special character . . . . .	DEB-18, LM-10, LM-11, PM-131, SJM-8, SJM-9, SJM-209, SMU-6
\$\$ (double dollar sign)	
special character . . . . .	DEB-18, DEB-19, LM-10, LM-11, PM-132, SJM-8, SJM-9, SMU-6
% (percent)	
file utilities wildcard . . . . .	LM-172, LM-181, LM-184, LM-187
metacharacter . . . . .	EI-56
special character . . . . .	DEB-18, DEB-19, LM-10, LM-11, SJM-8, SJM-9, SMU-6
`body attribute . . . . .	LM-14, SJM-11
`L attributes, <i>see</i> link attributes	
`N attributes, <i>see</i> nickname attributes	
`S attributes, <i>see</i> state attributes	
`spec attribute . . . . .	LM-14, SJM-11
* (asterisk)	
file utilities wildcard . . . . .	LM-172, LM-181, LM-184, LM-187
metacharacter . . . . .	EI-56
symbol in window banner . . . . .	PM-210, PM-232, PM-244, PM-292
+ (plus) symbol . . . . .	PM-404
+ function	
Calendar.+ . . . . .	PT-8
Time_Uilities.+ . . . . .	PT-176
, (comma)	
in set notation . . . . .	LM-13, PM-133, SJM-11, SMU-8
separator . . . . .	LM-18, SJM-16, SMU-8

- (hyphen)		
indicating nondefault versions	.....	LM-198
- function		
Calendar.-	.....	PT-8
Time_Uilities.-	.....	PT-177
-n version attribute	.....	LM-14, SJM-12
.(period)		
special character	.....	DEB-18, DEB-19, LM-10, LM-12, PM-132, SJM-8, SJM-10, SMU-7
.. (double dot) symbol	.....	LM-18, SJM-16, SMU-9
:= (colon equals) value delimiter	.....	LM-18, SJM-15, SMU-8
;(semicolon)		
in set notation	.....	LM-13, PM-133, SJM-11, SMU-8
separator	.....	LM-18, PM-133, SJM-16, SMU-8
< (less than), <i>see also</i> Less_Than		
< function		
Calendar.<	.....	PT-8
Io.<	.....	TIO-10
< generic formal function		
Table_Sort_Generic.<	.....	PT-170
<= function		
Calendar.<=	.....	PT-8
= (equals)		
symbol in window banner	.....	PM-135
<i>see also</i> equal, equals		
= function		
Io.=	.....	TIO-9
= value delimiter	.....	LM-18, SJM-15, SMU-8
=> value delimiter	.....	LM-18, SJM-15, SMU-8
> (greater than), <i>see also</i> Greater_Than		
> function		
Calendar.>	.....	PT-8
Io.>	.....	TIO-11
>= function		
Calendar.>=	.....	PT-8
? (question mark)		
file utilities wildcard	.....	LM-172, LM-181, LM-184, LM-187
library wildcard	LM-8, LM-9, LM-109, LM-113, LM-297, LM-301, PM-129, SJM-6, SJM-7, SMU-3, SMU-4	
metacharacter	.....	EI-56
substitution character	.....	LM-10, PM-130, SJM-8, SMU-5
?? (double question mark)		
library wildcard	.....	LM-8, LM-9, PM-129, SJM-6, SJM-7, SMU-3, SMU-4
@ (at sign)		
indicating frozen window state	.....	EI-63, EI-65, EI-67
library wildcard	LM-8, LM-9, LM-109, LM-113, LM-297, LM-301, PM-129, SJM-6, SMU-3	
substitution character	.....	LM-10, PM-130, SJM-8, SMU-4
[ ] (brackets)		
file utilities wildcards	.....	LM-172, LM-181, LM-184, LM-187
metacharacters	.....	EI-57
special characters	LM-13, LM-109, LM-113, LM-297, LM-301, PM-131, PM-133,	
.....	SJM-8, SJM-11, SMU-5, SMU-8	

\ (backslash)	
file utilities wildcard . . . . .	LM-172, LM-181, LM-184, LM-187
metacharacter . . . . .	EI-57
special character . . . . .	DEB-20, LM-10, LM-12, PM-132, SJM-8, SJM-10, SJM-210, SMU-7
^ (caret)	
file utilities wildcard . . . . .	LM-172, LM-181, LM-184, LM-187
metacharacter . . . . .	EI-56
special character . . . . .	DEB-18, LM-10, LM-11, PM-131, SJM-8, SMU-6
_ (underscore)	
identifier character . . . . .	LM-9, SJM-6
special character . . . . .	DEB-18, DEB-19, LM-10, LM-12, PM-132, SJM-8, SJM-9, SMU-7
` (grave)	
special character . . . . .	DEB-20, LM-10, LM-12, PM-132, SJM-4, SJM-8, SJM-10, SJM-209, SMU-7
" " execution message (Debug.Task_Display) . . . . .	DEB-136
{ } (braces)	
file utilities wildcards . . . . .	LM-172, LM-181, LM-184, LM-187
indicating deleted objects . . . . .	LM-198
metacharacters . . . . .	EI-56
special characters . . . . .	LM-13, PM-131, PM-133, SJM-8, SJM-11, SMU-5, SMU-8
(bar) symbol . . . . .	LM-18, SJM-16, SMU-9
~ (tilde)	
indicating replace window state . . . . .	EI-63, EI-65, EI-67
symbol . . . . .	LM-13, LM-18, LM-109, LM-113, LM-297, LM-301, PM-133, SJM-16, SMU-8, SMU-9

A

A.M.	
Time_Uilities.Sun_Positions type . . . . .	PT-202
abandon . . . . .	PM-27
Abandon procedure	
Common.Abandon . . . . .	EST-62, PM-135
Ada images . . . . .	EST-9
Cmvc.Append_Notes procedure . . . . .	PM-210
Cmvc.Get_Notes procedure . . . . .	PM-244
Cmvc.Put_Notes procedure . . . . .	PM-292
command images . . . . .	EST-47
Debugger . . . . .	DEB-5
Help . . . . .	EST-126
Library package . . . . .	LM-203
Links package . . . . .	LM-276
menu images . . . . .	EST-133
Search_List package . . . . .	SJM-210
session switches . . . . .	SJM-248
Switches package . . . . .	LM-315
text images . . . . .	EST-140
What package . . . . .	SJM-253
windows images . . . . .	EST-156
xref images . . . . .	EST-161
Abandon_Reservation procedure	
Cmvc.Abandon_Reservation . . . . .	PM-28, PM-202
Cmvc.Check_Out procedure . . . . .	PM-218
abort, <i>see</i> Cancel, Delete_Group, Delete_User, Force_Logoff, kill	
aborting task execution message (Debug.Task_Display) . . . . .	DEB-136

About_To_Return enumeration	
Debug.Stop_Event type	DEB-130
accept changes	PM-14, PM-40, PM-41, PM-202, PM-205
Accept_Changes procedure	
Cmvc.Accept_Changes	PM-41, PM-42, PM-46, PM-205
Cmvc.Merge_Changes procedure	PM-288
Cmvc.Revert procedure	PM-306
access	
controlled objects, concurrently	PM-43
access control	DIO-5, LM-19, SMU-53, TIO-5
affect on procedures in packages in !Commands	LM-23
archiving	LM-23
change identity	
Program.Change_Identity procedure	SJM-174
classes	LM-20, LM-21
command execution	LM-22
compilation	LM-22, LM-129
default access list	LM-21
groups	LM-20, SMU-54
identity	LM-19
job	LM-19
library commands	LM-195
links	LM-22
networking	LM-22
objects	LM-21
searchlists	LM-23
subsystems	LM-23
user	LM-19
access list	LM-1, LM-19, LM-195, SMU-53
add	
Access_List.Add procedure	LM-32
add default	
Access_List.Add_Default procedure	LM-33
amended	
Access_List_Tools.Amend function	LM-57
change	
Access_List.Add procedure	LM-32
Access_List.Add_Default procedure	LM-33
Access_List.Set procedure	LM-44
Access_List.Set_Default procedure	LM-46
Access_List_Tools.Set procedure	LM-81
Access_List_Tools.Set_Default procedure	LM-83
check	
Access_List_Tools.Check function	LM-59
class	
Access_List_Tools.Access_Class subtype	LM-54
classes	
Access_List.Create constant	LM-34
Access_List.Delete constant	LM-37
Access_List.Owner constant	LM-42
Access_List.Read constant	LM-43
Access_List.Write constant	LM-48
Access_List_Tools.Create constant	LM-64
Access_List_Tools.Delete constant	LM-65
Access_List_Tools.Owner constant	LM-79
Access_List_Tools.Read constant	LM-80
Access_List_Tools.Write constant	LM-85

access list (continued)

create	
Access_List.Add procedure . . . . .	LM-32
Access_List.Set procedure . . . . .	LM-44
Access_List.Set_Default procedure . . . . .	LM-46
Access_List_Tools.Set procedure . . . . .	LM-81
Access_List_Tools.Set_Default procedure . . . . .	LM-83
Daemon.Get_Access_List_Compaction function . . . . .	SMU-17
Daemon.Set_Access_List_Compaction procedure . . . . .	SMU-35
default display	
Access_List.Default_Display procedure . . . . .	LM-35
display	
Access_List.Display procedure . . . . .	LM-38
edit	
Access_List.Add procedure . . . . .	LM-32
Access_List.Add_Default procedure . . . . .	LM-33
error	
Access_List_Tools.Access_Tools_Error exception . . . . .	LM-55
get	
Access_List_Tools.Get function . . . . .	LM-66
Access_List_Tools.Get procedure . . . . .	LM-68
get default	
Access_List_Tools.Get_Default function . . . . .	LM-70
Access_List_Tools.Get_Default procedure . . . . .	LM-72
maximum length	
Access_List_Tools.Max_Acl_Length constant . . . . .	LM-75
name	
Access_List.Name subtype . . . . .	LM-41
Access_List_Tools.Name subtype . . . . .	LM-76
normalize	
Access_List_Tools.Normalize function . . . . .	LM-77
operator capability	
Access_List_Tools.Has_Operator_Capability function . . . . .	LM-74
remove old entries	
Access_List_Tools.Normalize function . . . . .	LM-77
set	
Access_List.Set procedure . . . . .	LM-44
Access_List_Tools.Set procedure . . . . .	LM-81
set default	
Access_List.Set_Default procedure . . . . .	LM-46
Access_List_Tools.Set_Default procedure . . . . .	LM-83
validity	
Access_List_Tools.Check_Validity procedure . . . . .	LM-62
Access_Class subtype	
Access_List_Tools.Access_Class . . . . .	LM-54
Access_Error	
Io_Exceptions.Use_Error exception . . . . .	DIO-37, TIO-163
Access_List package . . . . .	LM-25
Access_List_Tools package . . . . .	LM-49
Access_Tools_Error exception	
Access_List_Tools.Access_Tools_Error . . . . .	LM-55
Amend function . . . . .	LM-57
Check function . . . . .	LM-60
Get function . . . . .	LM-66
Get_Default function . . . . .	LM-70
Account library switch . . . . .	LM-309

Account session switch . . . . .	SJM-78, SJM-230
ACL, <i>see</i> access list	
Acl subtype	
Access_List.Acl . . . . .	LM-31
Access_List_Tools.Acl . . . . .	LM-56
Actions	
client . . . . .	SMU-13, SMU-27
object manager . . . . .	SMU-11, SMU-12, SMU-58
[Activate] key	
Debug.Activate procedure . . . . .	DEB-30
Activate procedure	
Debug.Activate . . . . .	DEB-11, DEB-30
Remove procedure . . . . .	DEB-107
activating child packages execution message (Debug.Task_Display) . . . . .	DEB-136
activating child tasks execution message (Debug.Task_Display) . . . . .	DEB-136
active breakpoint . . . . .	DEB-11
activity . . . . .	PM-12, PM-29, PM-52, PM-65, PM-136, PM-137, PM-139
adding entries . . . . .	PM-66
creating an empty activity . . . . .	PM-66
default	
Profile.Default_Activity function . . . . .	SJM-84
defined . . . . .	PM-1
differential entries . . . . .	PM-82
editing . . . . .	PM-135, PM-150
images . . . . .	EST-1
modes for creating entries . . . . .	PM-82
release . . . . .	PM-16
set	
Profile.Set_Activity procedure . . . . .	SJM-140
Profile.Set_Default_Activity procedure . . . . .	SJM-142
setting the default . . . . .	PM-67
specifying compatible load views in . . . . .	PM-94
type	
Profile.Activity_Type subtype . . . . .	SJM-81
using for execution . . . . .	PM-65
window . . . . .	PM-175
Activity function	
Profile.Activity . . . . .	SJM-80
Activity package . . . . .	PM-1, PM-135
Activity procedure	
Check.Activity . . . . .	PM-178
Activity profile attribute . . . . .	SJM-73
<ACTIVITY> special name . . . . .	EST-58, EST-131, EST-155, LM-8, LM-130, LM-170, LM-199, LM-308, PM-128, SJM-5, SMU-2
Activity subclass . . . . .	LM-17, SJM-15
Activity_File session switch . . . . .	SJM-230
Activity.Set_Default procedure . . . . .	PM-161
Activity_Name subtype	
Activity.Activity_Name . . . . .	PM-139
Activity_Type subtype	
Profile.Activity_Type . . . . .	SJM-81

Ada	
class	LM-14, LM-16, SJM-12, SJM-14
client	SMU-13, SMU-17, SMU-20, SMU-27, SMU-35
images	EST-1, EST-3
commands from package Common	EST-9
committing images	EST-8
cursor designation	EST-4
designation	EST-3
image structure	EST-3
incremental compilation	EST-7, EST-8
insertion points	EST-7
key concepts	EST-3
library switches	EST-9
locks	EST-9
selection	EST-4
special names	EST-5
unit states	EST-5
versions	EST-8
library, <i>see</i> directory, library, world	
name	LM-7, PM-127, SJM-5, SMU-1
name resolution mode	LM-12, PM-132, SJM-10, SMU-7
object manager	SMU-11, SMU-58
subclass attributes	LM-16, SJM-14
Ada enumeration	
Time_Uilities.Date_Format type	PT-180
Time_Uilities.Time_Format type	PT-204
Ada package	EST-6, EST-19
Ada unit	
access classes	LM-21
Access_List.Read constant	LM-43
Access_List.Write constant	LM-48
Access_List_Tools.Read constant	LM-80
Access_List_Tools.Write constant	LM-85
define coded	
Xref.Uses procedure	LM-347
define installed	
Xref.Uses procedure	LM-347
library switches	LM-309
list coded	
Xref.Used_By procedure	LM-342
list installed	
Xref.Used_By procedure	LM-342
pretty-printing	
Library.Reformat_Image procedure	LM-255
restore	LM-89
session switches	SJM-228
Ada_Format constant	
Library.Ada_Format	LM-208
Ada_List renamed procedure	
Library.Ada_List	LM-3, LM-209
Ada_Format constant	LM-208
Ada_Location function	
Debug_Tools.Ada_Location	DEB-17, DEB-152
add	
comment	
Editor.Region.Comment procedure	EI-46



add ( <i>continued</i> )	
cross-library link	
Links.Add procedure . . . . .	LM-279
entry, <i>see</i> Create, Create_Group, Create_Session, Create_User, Define, Insert, Push	
to end, <i>see</i> Append	
Add procedure	
Access_List.Add . . . . .	LM-32
Activity.Add . . . . .	PM-82, PM-140
Links.Add . . . . .	LM-279
Cmvc_Maintenance.Check_Consistency procedure . . . . .	PM-341
Insert procedure . . . . .	LM-293
Link_Name subtype . . . . .	LM-297
Replace procedure . . . . .	LM-298
Queue.Add . . . . .	SMU-93
Register procedure . . . . .	SMU-123
Queue_Generic.Add . . . . .	PT-96
Search_List.Add . . . . .	SJM-213
Set_Generic.Add . . . . .	PT-112
Work_Order.List_Editor.Add . . . . .	PM-414
Add_Child procedure	
Cmvc_Hierarchy.Add_Child . . . . .	PM-326, PM-328
Add_Comment procedure	
Work_Order.Editor.Add_Comment . . . . .	PM-405
Add_Configuration procedure	
Work_Order.Editor.Add_Configuration . . . . .	PM-406
Add_Default procedure	
Access_List.Add_Default . . . . .	LM-33
Add_To_Group procedure	
Operator.Add_To_Group . . . . .	SMU-56
Create_Group procedure . . . . .	SMU-61
Add_To_List procedure	
Work_Order.Add_To_List . . . . .	PM-365
Add_User procedure	
Work_Order.Editor.Add_User . . . . .	PM-407
Add_Version procedure	
Work_Order.Editor.Add_Version . . . . .	PM-408
address	
Debug.Location_To_Address procedure . . . . .	DEB-77
System.Null_Address constant . . . . .	PT-166
Address type	
System.Address . . . . .	PT-164
Address_To_Location procedure	
Debug.Address_To_Location . . . . .	DEB-31
Debug_Tools.Get_Exception_Name function . . . . .	DEB-157
Debug_Tools.Get_Raise_Location function . . . . .	DEB-159
Location_To_Address procedure . . . . .	DEB-77
Addresses enumeration	
Debug.Option type . . . . .	DEB-86
Adjust type	
Table_Formatter.Adjust . . . . .	ST-93
Alert procedure	
Editor.Alert . . . . .	EI-10

Alignment_Threshold library switch . . . . .	LM-309
All version attribute . . . . .	LM-14, SJM-12
all, send	
Message.Send_All procedure . . . . .	SMU-51
All_Bad_Blocks constant	
System_Uilities.All_Bad_Blocks . . . . .	SMU-198
All_Classes constant	
Queue.All_Classes . . . . .	SMU-96
All_Events enumeration	
Debug.Trace_Event type . . . . .	DEB-146
All_Fields constant	
Library.All_Fields . . . . .	LM-211
All_Parts enumeration	
Compilation.Promote_Scope type . . . . .	LM-160
All_Spooler_Devices constant	
Queue.All_Spooler_Devices . . . . .	SMU-97
All_State enumeration	
Debug.State_Type type . . . . .	DEB-126
All_Tasks enumeration	
Debug.Task_Category type . . . . .	DEB-135
All_Worlds constant	
Compilation.All_Worlds . . . . .	LM-132
<ALL_WORLDS> special value . . . . .	LM-129, LM-134, LM-199
Allocate function	
String_Table.Allocate . . . . .	ST-50
Allow_Edit_Of_Work_Orders enumeration	
Work_Order.Venture_Policy_Switch . . . . .	PM-400
Work_Order.Create_Field procedure . . . . .	PM-370
Allows_Deallocation function	
Allows_Deallocation.Allows_Deallocation . . . . .	PT-2
Allows_Deallocation generic function . . . . .	PT-1
Unchecked_Deallocation generic procedure . . . . .	PT-241
Unchecked_Deallocation procedure . . . . .	PT-246
alphanumeric character set . . . . .	DIO-82, DIO-157
Already_Open_Error	
Io_Exceptions.Status_Error exception . . . . .	DIO-36, TIO-162
Alt_List subclass . . . . .	LM-16, SJM-14
Ambiguous_Name_Error	
Io_Exceptions.Name_Error exception . . . . .	DIO-35, TIO-161
Amend function	
Access_List_Tools.Amend . . . . .	LM-57
ANSI format . . . . .	SMU-283
Any constant	
Links.Any . . . . .	LM-281
Any version attribute . . . . .	LM-14, SJM-12
Append procedure	
Bounded_String.Append . . . . .	ST-2
File_Uilities.Append . . . . .	LM-171

Append procedure ( <i>continued</i> )	
Io.Append . . . . .	TIO-12
Polymorphic_Sequential_Io.Append . . . . .	DIO-40
Unbounded_String.Append . . . . .	ST-104
Append_Notes procedure	
Cmvc.Append_Notes . . . . .	PM-210
Cmvc.Create_Empty_Note_Window procedure . . . . .	PM-232
Cmvc.Get_Notes procedure . . . . .	PM-244
Cmvc.Put_Notes procedure . . . . .	PM-292
application	
execution . . . . .	PM-84
single library . . . . .	PM-15
testing . . . . .	PM-85
archive	
access control . . . . .	LM-23
copy . . . . .	LM-87, LM-90
Archive.Copy procedure . . . . .	LM-100
hints . . . . .	LM-98
list . . . . .	LM-87
Archive.List procedure . . . . .	LM-109
Operator.Get_Archive_On_Shutdown function . . . . .	SMU-76
restore . . . . .	LM-87, LM-89
Archive.Restore procedure . . . . .	LM-112
save . . . . .	LM-87, LM-88
Archive.Save procedure . . . . .	LM-122
<i>see also</i> Backup	
Archive package . . . . .	LM-87
Archive_On_Shutdown procedure	
Operator.Archive_On_Shutdown . . . . .	SMU-58
Get_Archive_On_Shutdown function . . . . .	SMU-76
Show_Shutdown_Settings procedure . . . . .	SMU-87
Archived enumeration	
Compilation.Unit_State type . . . . .	LM-166
archived unit state . . . . .	EST-5
Archived_Code class . . . . .	LM-14, SJM-12
Archived_Code object manager . . . . .	SMU-11, SMU-58
argument prefixes . . . . .	DEB-4
Argument_Digit procedure	
Editor.Set.Argument_Digit . . . . .	EI-60
Argument_Minus procedure	
Editor.Set.Argument_Minus . . . . .	EI-60
Argument_Prefix procedure	
Editor.Set.Argument_Prefix . . . . .	EI-60
array	
Table_Sort_Generic.Element_Array generic formal type . . . . .	PT-172
ASCII characters . . . . .	DIO-1, DIO-85, TIO-1
Ascii package	
Standard.Ascii . . . . .	PT-161
Ascii.Ff . . . . .	DIO-6, TIO-5

Ascii.Lf . . . . .	DIO-6, TIO-5
Asm_Listing library switch . . . . .	LM-309
Assertion_Error exception	
System.Assertion_Error . . . . .	PT-164
Unchecked_Conversion.Unchecked_Conversion function . . . . .	PT-213
Unchecked_Conversions.Convert_From_Byte_String function . . . . .	PT-233
Unchecked_Conversions.Unchecked_Conversion_Package.Convert function . . . . .	PT-227
Associate procedure	
Switches.Associate . . . . .	LM-318
Dissociate procedure . . . . .	LM-328
Associated function	
Switches.Associated . . . . .	LM-320
asterisk (*)	
file utilities wildcard . . . . .	LM-172, LM-181, LM-184, LM-187
metacharacter . . . . .	EI-56
symbol in window banner . . . . .	PM-210, PM-232, PM-244, PM-292
at sign (@)	
indicating frozen window state . . . . .	EI-63, EI-65, EI-67
library wildcard . . . . .	LM-8, LM-9, LM-109, LM-113, LM-297, LM-301, PM-129, SJM-6, SMU-3
substitution character . . . . .	LM-10, PM-130, SJM-8, SMU-4
At_Msg enumeration	
Profile.Msg_Kind type . . . . .	SJM-117
Atomic_Destroy procedure	
Compilation.Atomic_Destroy . . . . .	LM-133
attach, <i>see</i> Connect	
Attached enumeration	
Scheduler.Job_Kind type . . . . .	SMU-165
attached job . . . . .	SMU-133
attempting entry call execution message (Debug.Task_Display) . . . . .	DEB-136
attribute	
job	
Scheduler.Get_Job_Attribute function . . . . .	SMU-153
Scheduler.Set_Job_Attribute procedure . . . . .	SMU-179
Attribute type	
Window_Io.Attribute . . . . .	DIO-102
attributes . . . . .	LM-7, LM-13, PM-127, SJM-5, SJM-11, SJM-73, SMU-1
class . . . . .	LM-14, SJM-12
link . . . . .	LM-15, SJM-13
nickname . . . . .	LM-15, SJM-13
state . . . . .	LM-17, SJM-13
version . . . . .	LM-14, SJM-12
visible parts and bodies . . . . .	LM-13, SJM-11
Auto_Login library switch . . . . .	LM-310
Auto_Login session switch . . . . .	SJM-230
automated compilation	
Compilation.Make renamed procedure . . . . .	LM-151
Compilation.Promote procedure . . . . .	LM-157
Auxiliary_Msg enumeration	
Profile.Msg_Kind type . . . . .	SJM-117

**B**

background, <i>see</i> Disconnect	
background job . . . . .	SMU-132, SMU-136
streams . . . . .	SMU-137
parameters . . . . .	SMU-172, SMU-177
Scheduler.Display procedure . . . . .	SMU-144
backoff . . . . .	SMU-15
backslash (\)	
file utilities wildcard . . . . .	LM-172, LM-181, LM-184, LM-187
metacharacter . . . . .	EI-57
special character . . . . .	DEB-20, LM-10, LM-12, PM-132, SJM-8, SJM-10, SJM-210, SMU-7
backup	
System_Backup package . . . . .	SMU-191
Backup procedure	
System_Backup.Backup . . . . .	SMU-192
Kind type . . . . .	SMU-196
backward	
deletion	
Editor.Char.Delete_Backward procedure . . . . .	EI-12
Editor.Line.Delete_Backward procedure . . . . .	EI-33
Editor.Word.Delete_Backward procedure . . . . .	EI-70
movement	
Editor.Cursor.Previous procedure . . . . .	EI-19
Editor.Line.Previous procedure . . . . .	EI-34
Editor.Window.Previous procedure . . . . .	EI-67
Editor.Word.Previous procedure . . . . .	EI-71
search . . . . .	EI-4
Editor.Search.Previous procedure . . . . .	EI-57
search and replace . . . . .	EI-4
Editor.Search.Replace_Previous procedure . . . . .	EI-58
tab	
Editor.Char.Tab_Backward procedure . . . . .	EI-14
Backward procedure	
Editor.Cursor.Backward . . . . .	EI-17, EI-18
bad blocks	
System_Uilities.All_Bad_Blocks constant . . . . .	SMU-198
System_Uilities.Manufacturers_Bad_Blocks constant . . . . .	SMU-240
Bad_Block_Kinds type	
System_Uilities.Bad_Block_Kinds . . . . .	SMU-199
Bad_Block_List function	
System_Uilities.Bad_Block_List . . . . .	SMU-200
banner	
Window_Io.Read_Banner function . . . . .	DIO-161
Window_Io.Set_Banner procedure . . . . .	DIO-168
Banner session switch . . . . .	SJM-230
bar ( ) symbol . . . . .	LM-18, SJM-16, SMU-9
base	
default	
Io.Integer_Io.Default_Base constant . . . . .	TIO-144
Text_Io.Integer_Io.Default_Base constant . . . . .	TIO-252
number	
Io.Number_Base subtype . . . . .	TIO-62
Text_Io.Number_Base subtype . . . . .	TIO-192

batch, <i>see</i> Disconnect	
batch job, <i>see</i> background job	
baud rates	
System_Uilities.Input_Rate function . . . . .	SMU-230
System_Uilities.Output_Rate function . . . . .	SMU-247
Terminal.Set_Input_Rate procedure . . . . .	SMU-312
Terminal.Set_Output_Rate procedure . . . . .	SMU-316
Beep_On_Errors session switch . . . . .	SJM-230
Beep_On_Interrupt session switch . . . . .	SJM-230
Beep_On_Messages session switch . . . . .	SJM-231
begin, <i>see</i> Run	
begin at scheduled time	
Daemon.Schedule procedure . . . . .	SMU-33
Begin_Rendezvous enumeration	
Debug.Stop_Event type . . . . .	DEB-130
beginning, <i>see</i> First	
Beginning_Of procedure	
Editor.Image.Beginning_Of . . . . .	EI-25
Editor.Line.Beginning_Of . . . . .	EI-31, EI-32
Editor.Region.Beginning_Of . . . . .	EI-45
Editor.Window.Beginning_Of . . . . .	EI-64
Editor.Word.Beginning_Of . . . . .	EI-69
being aborted execution message (Debug.Task_Display) . . . . .	DEB-137
bell	
Editor.Alert procedure . . . . .	EI-10
Bell procedure	
Window_Io.Bell . . . . .	DIO-104
binary files, create	
Direct_Io.Create procedure . . . . .	DIO-10
Polymorphic_Sequential_Io.Create procedure . . . . .	DIO-42
Sequential_Io.Create procedure . . . . .	DIO-63
binary objects, controlling . . . . .	PM-25
Binary subclass . . . . .	LM-17, SJM-15
Bind procedure	
Editor.Macro.Bind . . . . .	EI-37, EI-38
binding keys	
Editor.Key package . . . . .	EI-27
Bit constant	
System.Bit . . . . .	PT-164
bits	
System_Uilities.Character_Bits_Range subtype . . . . .	SMU-203
System_Uilities.Stop_Bits function . . . . .	SMU-263
System_Uilities.Stop_Bits_Range subtype . . . . .	SMU-264
Terminal.Character_Bits_Range subtype . . . . .	SMU-302
Terminal.Set_Stop_Bits procedure . . . . .	SMU-322
Terminal.Stop_Bits_Range subtype . . . . .	SMU-327
blank line	
Ada.Delete_Blank_Line procedure . . . . .	EST-26
Ada.Insert_Blank_Line procedure . . . . .	EST-29

block	
System_Uilities.All_Bad_Blocks constant	SMU-198
System_Uilities.Bad_Block_Kinds type	SMU-199
System_Uilities.Bad_Block_List function	SMU-200
System_Uilities.Manufacturers_Bad_Blocks constant	SMU-240
System_Uilities.Retargeted_Blocks constant	SMU-256
<i>see also</i> Disable, Hold, Quiesce	
Block procedure	
Text.Block	EST-146
Block_List type	
System_Uilities.Block_List	SMU-201
Blocked enumeration	
Debug.Task_Category type	DEB-135
blue tapes	SMU-191
board information	
System_Uilities.Get_Board_Info function	SMU-218
body	
Ada.Create_Body procedure	EST-22
Bold character attribute	DIO-102
Boolean options	LM-18, SJM-16, SMU-9
Boolean type	
Standard.Boolean	PT-161
Boolean value	
read from file	
Io.Get procedure	TIO-49
write to current error file (Message window)	
Io.Echo procedure	TIO-31
write to file	
Io.Put procedure	TIO-78
boot	
configuration	
System_Uilities.System_Boot_Configuration function	SMU-265
last	
System_Uilities.System_Up_Time function	SMU-266
Both enumeration	
Time_Uilities.Image_Contents type	PT-192
bottom	
image	
Editor.Image.End_Of procedure	EI-26
of selection	
Editor.Region.End_Of procedure	EI-46
of window	
Editor.Window.End_Of procedure	EI-66
Bounded_String package	ST-1
braces ( {} )	
file utilities wildcards	LM-172, LM-181, LM-184, LM-187
indicating deleted objects	LM-198
metacharacters	EI-56
special characters	LM-13, PM-131, PM-133, SJM-8, SJM-11, SMU-5, SMU-8
brackets ( [ ] )	
file utilities wildcards	LM-172, LM-181, LM-184, LM-187
metacharacters	EI-57

brackets ([ ]) ( <i>continued</i> )	
special characters . . . . .	LM-13, LM-109, LM-113, LM-297, LM-301, PM-131, PM-133, SJM-8, SJM-11, SMU-5, SMU-8
[Break ~Default] key	
Debug.Break(False) procedure . . . . .	DEB-32
break characters . . . . .	EI-70
[Break] key	
Debug.Break procedure . . . . .	DEB-32
Break procedure	
Debug.Break . . . . .	DEB-11, DEB-18, <i>DEB-32</i>
Context procedure . . . . .	DEB-44, DEB-45
Display procedure . . . . .	DEB-53
Editor.Word.Break	
Word_Breaks session switch . . . . .	SJM-248
Break_At_Creation enumeration	
Debug.Option type . . . . .	DEB-86
breakpoint . . . . .	DEB-10
active/inactive . . . . .	DEB-11
cancel	
Debug.Forget procedure . . . . .	DEB-65
Debug.Remove procedure . . . . .	DEB-107
cancel exception	
Debug.Propagate procedure . . . . .	DEB-96
create	
Debug.Break procedure . . . . .	DEB-32
programmatic	
Debug_Tools.User_Break procedure . . . . .	DEB-185
reactivate	
Debug.Activate procedure . . . . .	DEB-30
set	
Debug.Break procedure . . . . .	DEB-32
show	
Debug.Information procedure . . . . .	DEB-73
stop execution	
Debug.Catch procedure . . . . .	DEB-36
temporary/permanent . . . . .	DEB-11
Breakpoints enumeration	
Debug.State_Type type . . . . .	DEB-126
Breaks procedure	
Editor.Word.Breaks . . . . .	EI-69, <i>EI-70</i>
broadcast bulletin, <i>see</i> Send, Send_All	
brother	
Common.Object.Next procedure . . . . .	EST-113
buffer	
create	
Text.Create procedure . . . . .	EST-148
force characters to file	
Io.Flush procedure . . . . .	TIO-39
Build procedure	
Cmvc.Build . . . . .	PM-33, PM-49, PM-50, <i>PM-212</i>
Cmvc.Destroy_View procedure . . . . .	PM-239
Cmvc.Release procedure . . . . .	PM-294
Build_Activity procedure	
Cmvc_Hierarchy.Build_Activity . . . . .	PM-326, <i>PM-329</i>



byte	
convert from byte string	
Unchecked_Conversions.Convert_From_Byte_String function	PT-232
Unchecked_Conversions.Convert_From_Byte_String generic function	PT-231
convert to byte string	
Unchecked_Conversions.Convert_To_Byte_String function	PT-238
Unchecked_Conversions.Convert_To_Byte_String generic function	PT-237
receive Xon/Xoff bytes	
System_Uilities.Receive_Xon_Xoff_Bytes function	SMU-254
set receive Xon/Xoff bytes	
Terminal.Set_Receive_Xon_Xoff_Bytes procedure	SMU-320
set Xon/Xoff bytes	
Terminal.Set_Xon_Xoff_Bytes procedure	SMU-324
Xon/Xoff	
System_Uilities.Xon_Xoff_Bytes function	SMU-280
Byte type	
System.Byte	PT-164
Byte_Size constant	
System.Byte_Size	PT-164
Byte_String subtype	
System_Uilities.Byte_String	SMU-202
Byte_String type	
System.Byte_String	PT-164

C

Cache_Stack_Frames debugger flag	DEB-63
cached	
Profile.Get_Cached_Resolution procedure	SJM-102
Cached_Selected_Text function	
Profile.Cached_Selected_Text	SJM-82
Calendar package	PT-5
Call enumeration	
Debug.Trace_Event type	DEB-146
call stacks	DEB-8
cancel break, <i>see</i> Remove	
cancel exception break, <i>see</i> Forget, Propagate	
Cancel procedure	
Queue.Cancel	SJM-198, SMU-98
Display procedure	SJM-201, SMU-109
Cancel_Shutdown procedure	
Operator.Cancel_Shutdown	SMU-59
Capability_Error exception	
System.Capability_Error	PT-164
Unchecked_Conversion generic function	PT-209
Unchecked_Conversion.Target generic formal type	PT-211
Unchecked_Conversion.Unchecked_Conversion function	PT-212, PT-213
Unchecked_Conversions.Convert_From_Byte_String function	PT-232, PT-233
Unchecked_Conversions.Target generic formal type	PT-235
Unchecked_Conversions.Unchecked_Conversion_Package generic package	PT-225
Unchecked_Conversions.Unchecked_Conversion_Package.Convert function	PT-226, PT-227
Unchecked_Conversions.Unchecked_Conversion_Package.Target generic formal type	PT-230

Capacity_Error	
Io_Exceptions.Use_Error exception . . . . .	DIO-37, TIO-163
capitalize, <i>see also</i> Upper_Case	
Capitalize function	
String_Utilities.Capitalize . . . . .	ST-70
Capitalize procedure	
Editor.Char.Capitalize . . . . .	EI-12
Editor.Line.Capitalize . . . . .	EI-32
Editor.Region.Capitalize . . . . .	EI-46
Editor.Word.Capitalize . . . . .	EI-70
String_Utilities.Capitalize . . . . .	ST-71
car, <i>see</i> First	
Cardinality function	
Concurrent_Map_Generic.Cardinality . . . . .	PT-10
Map_Generic.Cardinality . . . . .	PT-68
String_Map_Generic.Cardinality . . . . .	ST-26
caret (^)	
file utilities wildcard . . . . .	LM-172, LM-181, LM-184, LM-187
metacharacter . . . . .	EI-56
special character . . . . .	DEB-18, LM-10, LM-11, PM-131, SJM-8, SMU-6
case	
capitalize	
Editor.Char.Capitalize procedure . . . . .	EI-12
Editor.Line.Capitalize procedure . . . . .	EI-32
Editor.Word.Capitalize procedure . . . . .	EI-70
String_Utilities.Capitalize function . . . . .	ST-70
String_Utilities.Capitalize procedure . . . . .	ST-71
conversion of strings	
String_Map_Generic generic package . . . . .	ST-25
String_Utilities package . . . . .	ST-69
ignore	
String_Map_Generic.Ignore_Case generic formal object . . . . .	ST-32
lowercase	
Editor.Char.Lower_Case procedure . . . . .	EI-14
Editor.Line.Lower_Case procedure . . . . .	EI-34
Editor.Region.Lower_Case procedure . . . . .	EI-48
Editor.Word.Lower_Case procedure . . . . .	EI-71
Io.Lower_Case constant . . . . .	TIO-57
String_Utilities.Lower_Case function . . . . .	ST-78
String_Utilities.Lower_Case procedure . . . . .	ST-79
uppercase	
Editor.Char.Upper_Case procedure . . . . .	EI-15
Editor.Line.Upper_Case procedure . . . . .	EI-35
Editor.Region.Upper_Case procedure . . . . .	EI-49
Editor.Word.Upper_Case procedure . . . . .	EI-72
Io.Upper_Case constant . . . . .	TIO-104
String_Utilities.Upper_Case function . . . . .	ST-89
String_Utilities.Upper_Case procedure . . . . .	ST-90
[Catch] key	
Debug.Catch procedure . . . . .	DEB-36
Catch procedure	
Debug.Catch . . . . .	DEB-12, DEB-13, DEB-36
Context procedure . . . . .	DEB-44, DEB-45
Debug_Save_Exceptions session switch . . . . .	SJM-235
Option type . . . . .	DEB-88
Propagate procedure . . . . .	DEB-96

catch request . . . . .	DEB-12
category	
Debug.Task_Category type . . . . .	DEB-135
CDB (compatibility database) . . . . .	PM-105, PM-108
CDFs (Cross-Development Facilities)	
using with subsystems . . . . .	PM-111
cdr, <i>see</i> Rest	
Ce enumeration	
Scheduler.Job_Kind type . . . . .	SMU-165
Center procedure	
Editor.Line.Center . . . . .	EI-31, EI-32
Centered enumeration	
Table_Formatter.Adjust type . . . . .	ST-93
change	
name	
Library.Rename procedure . . . . .	LM-256
session switches	
Switches.Edit_Session_Attributes procedure . . . . .	LM-330
Change procedure	
Activity.Change . . . . .	PM-85, PM-142
Switches.Change . . . . .	LM-321
Change_Identity procedure	
Program.Change_Identity . . . . .	LM-19, SJM-174
Condition subtype . . . . .	SJM-177
Create_Job procedure . . . . .	SJM-179, SJM-180
Run_Job procedure . . . . .	SJM-190
Change_Limit subtype	
Compilation.Change_Limit . . . . .	LM-129, LM-134
All_Worlds constant . . . . .	LM-132
Current_Destroy constant . . . . .	LM-138
Same_Directories constant . . . . .	LM-162
Same_World constant . . . . .	LM-163
Same_Worlds constant . . . . .	LM-164
Change_Password procedure	
Operator.Change_Password . . . . .	SJM-66, SMU-53, SMU-60
Create_User procedure . . . . .	SMU-64
Char package	
Editor.Char . . . . .	EI-5, EI-11
Char_At function	
Bounded_String.Char_At . . . . .	ST-4
String_Table.Char_At . . . . .	ST-51
Unbounded_String.Char_At . . . . .	ST-106
Window_Io.Char_At . . . . .	DIO-106
Character type	
Standard.Character . . . . .	PT-161
Character_Bits_Range subtype	
System_Uilities.Character_Bits_Range . . . . .	SMU-203
Terminal.Character_Bits_Range . . . . .	SMU-302
Character_Set type	
Window_Io.Character_Set . . . . .	DIO-105
Character_Size function	
System_Uilities.Character_Size . . . . .	SMU-204

characters	
attributes	DIO-102
case conversion	
Bounded_String.Char_At function	ST-4
Editor.Char.Capitalize procedure	EI-12
Editor.Char.Lower_Case procedure	EI-14
Editor.Char.Upper_Case procedure	EI-15
Unbounded_String.Char_At function	ST-106
character pairs ([ ] and { })	LM-10, PM-131, SJM-8, SMU-5
deletion	
Editor.Char.Delete_Backward procedure	EI-12
Editor.Char.Delete_Forward procedure	EI-12
Editor.Char.Delete_Next procedure	EI-13
Editor.Char.Delete_Previous procedure	EI-13
Editor.Char.Delete_Spaces procedure	EI-13
deletion of	
Window_Io.Delete procedure	DIO-114
editing operations	
Editor.Char package	EI-11
extraction of	
Bounded_String.Extract function	ST-8
Unbounded_String.Extract function	ST-111
force from buffer to file	
Io.Flush procedure	TIO-39
insert	
Editor.Char.Insert_Character procedure	EI-13
read	
System_Uilities.Input_Count function	SMU-229
read current line except terminator	
Io.Get_Line function	TIO-50
read from file	
Io.Get procedure	TIO-41
Text_Io.Get procedure	TIO-181
receive Xon/Xoff characters	
System_Uilities.Receive_Xon_Xoff_Characters function	SMU-255
set receive Xon/Xoff characters	
Terminal.Set_Receive_Xon_Xoff_Characters procedure	SMU-321
set Xon/Xoff characters	
Terminal.Set_Xon_Xoff_Characters procedure	SMU-325
sets	DIO-80, DIO-82
Window_Io.Graphics constant	DIO-137
Window_Io.Plain constant	DIO-157
size	
Terminal.Set_Character_Size procedure	SMU-306
special	LM-7, LM-10, PM-127, SJM-5, SMU-1
strip leading	
String_Uilities.Strip function	ST-86
String_Uilities.Strip_Leading function	ST-87
strip trailing	
String_Uilities.Strip function	ST-86
String_Uilities.Strip_Trailing function	ST-88
transpose	
Editor.Char.Transpose procedure	EI-15
write to current error file (Message window)	
Io.Echo procedure	TIO-26
write to file	
Io.Put procedure	TIO-72
Io.Text_Io.Put procedure	TIO-198
written	
System_Uilities.Output_Count function	SMU-245
Xon/Xoff	
System_Uilities.Xon_Xoff_Characters function	SMU-281

Check function	
Access_List_Tools.Check . . . . .	LM-59
Check package . . . . .	PM-1, PM-177
check syntax	
Common.Format procedure . . . . .	EST-88
Check_Consistency procedure	
Cmvc_Maintenance.Check_Consistency . . . . .	PM-50, PM-340
Check_In procedure	
Cmvc.Check_In . . . . .	PM-26, PM-29, PM-216
Cmvc.Check_Out procedure . . . . .	PM-218
Cmvc.Make_Controlled procedure . . . . .	PM-264
Check_Out procedure	
Cmvc.Check_Out . . . . .	PM-26, PM-29, PM-39, PM-41, PM-42, PM-218
Cmvc.Make_Controlled procedure . . . . .	PM-264
Check_Out_Error	
Io_Exceptions.Use_Error exception . . . . .	DIO-37, TIO-163
Check_Validity procedure	
Access_List_Tools.Check_Validity . . . . .	LM-62
checkin . . . . .	PM-6, PM-26, PM-202, PM-216, PM-317
checkout . . . . .	PM-6, PM-25, PM-26, PM-202, PM-218, PM-310, PM-312, PM-315, PM-316, PM-317
canceling . . . . .	PM-27
retrieving latest generation . . . . .	PM-41
Child procedure	
Common.Object.Child . . . . .	EST-102, PM-137
Ada images . . . . .	EST-4, EST-16
command images . . . . .	EST-50
Debugger . . . . .	DEB-6
Help . . . . .	EST-126
Library package . . . . .	LM-206
Links package . . . . .	LM-277
menu images . . . . .	EST-135
Search_List package . . . . .	SJM-211
session switches . . . . .	SJM-250
Switches package . . . . .	LM-316
text images . . . . .	EST-142
What package . . . . .	SJM-254
windows images . . . . .	EST-158
xref images . . . . .	EST-163
Editor.Window.Child . . . . .	EI-64
child subsystem . . . . .	PM-16
children . . . . .	PM-16
Children function	
Cmvc_Hierarchy.Children . . . . .	PM-332
chmod, <i>see</i> Set	
circuit board	
System_Uilities.Get_Board_Info function . . . . .	SMU-218
circular importing . . . . .	PM-53, PM-79
class . . . . .	SMU-91
access	
Access_List_Tools.Access_Class subtype . . . . .	LM-54
Ada . . . . .	SJM-14
all	
Queue.All_Classes constant . . . . .	SMU-96

class ( <i>continued</i> )	
attributes . . . . .	LM-14, SJM-12
Ada . . . . .	LM-14, SJM-12
Archived_Code . . . . .	LM-14, SJM-12
File . . . . .	LM-14, SJM-12
Group . . . . .	LM-14, SJM-12
Library . . . . .	LM-14, SJM-12
Null_Device . . . . .	LM-14, SJM-12
Pipe . . . . .	LM-14, SJM-12
Session . . . . .	LM-14, SJM-12
Tape . . . . .	LM-14, SJM-12
Terminal . . . . .	LM-15, SJM-12
User . . . . .	LM-15, SJM-12
condition	
Daemon.Condition_Class type . . . . .	SMU-16
create	
Queue.Create procedure . . . . .	SMU-101
destroy	
Queue.Destroy procedure . . . . .	SMU-104
file . . . . .	SJM-15
library . . . . .	SJM-13
Simple_Status.Condition_Class type . . . . .	PT-130
Class enumeration	
Library.Field type . . . . .	LM-239
Class_Error	
Io_Exceptions.Use_Error exception . . . . .	DIO-37, TIO-163
Class_Name subtype	
Queue.Class_Name . . . . .	SMU-100
classes of access . . . . .	LM-20
Classes procedure	
Queue.Classes . . . . .	SJM-199, SMU-99
clear, <i>see</i> Make_Empty	
Clear procedure	
Editor.Screen.Clear . . . . .	EI-51, EI-52
Redraw procedure . . . . .	EI-54
Clear_Stepping procedure	
Debug.Clear_Stepping . . . . .	DEB-14, DEB-42
Context procedure . . . . .	DEB-44
Run procedure . . . . .	DEB-109
Show procedure . . . . .	DEB-119
Clear_Underlining procedure	
Common.Clear_Underlining . . . . .	EST-64
Ada images . . . . .	EST-10
Ada.Get_Errors procedure . . . . .	EST-28
command images . . . . .	EST-47
client views (subsystem imports) . . . . .	PM-10, PM-51
clients . . . . .	SMU-11
Actions . . . . .	SMU-27
Ada . . . . .	SMU-17, SMU-20, SMU-27, SMU-35
Daily . . . . .	SMU-12, SMU-22, SMU-48
DDB . . . . .	SMU-18, SMU-20, SMU-27
Directory . . . . .	SMU-17, SMU-20, SMU-27, SMU-35
Disk . . . . .	SMU-12, SMU-20, SMU-27, SMU-31, SMU-38
Error_Log . . . . .	SMU-12
File . . . . .	SMU-17, SMU-20, SMU-27, SMU-35

clients (continued)

major	
Daemon.Major_Clients constant	SMU-27
Snapshot	SMU-12, SMU-27, SMU-31
Weekly	SMU-12
when last run	
Daemon.Last_Run function	SMU-25
when next run	
Daemon.Next_Scheduled function	SMU-28
Clock function	
Calendar.Clock	PT-6
Close procedure	
Direct_Io.Close	DIO-8
Io.Close	TIO-14
Polymorphic_Sequential_Io.Close	DIO-41
Sequential_Io.Close	DIO-62
Text_Io.Close	TIO-166
Window_Io.Close	DIO-107
Window_Io.Raw.Close	DIO-172
Work_Order.Close	PM-366
closed private part	PM-11, PM-57, PM-87, PM-89, PM-113
Closed_Private_Part library switch	LM-310
CMVC (configuration management and version control)	PM-3
controlling binary objects	PM-25
controlling objects	PM-25
database	PM-6, PM-9, PM-25, PM-188, PM-212, PM-216, PM-232, PM-244, PM-285, PM-321, PM-340, PM-350
defined	PM-1
editing controlled objects	PM-26
managing CMVC information interactively	PM-188
overview	PM-3
Cmvc package	PM-1, PM-185
Cmvc_Break_Long_Lines session switch	PM-362
Cmvc_Capitalize session switch	PM-362
Cmvc_Comment_Extent session switch	PM-362
Cmvc_Configuration_Extent session switch	PM-362
Cmvc_Db subclass	LM-17, SJM-15
Cmvc_Enable_Relocation session switch	PM-196
Cmvc_Field_Extent session switch	PM-362
Cmvc_Hierarchy package	PM-2, PM-325
Cmvc_Indentation session switch	PM-362
Cmvc_Line_Length session switch	PM-362
Cmvc_Maintenance package	PM-2, PM-339
Cmvc_Shorten_Name session switch	PM-362
Cmvc_Shorten_Unit_State session switch	PM-363
Cmvc_Show_Add_Date session switch	PM-363
Cmvc_Show_Add_Time session switch	PM-363
Cmvc_Show_All_Default_Lists session switch	PM-363

Cmvc_Show_All_Default_Orders session switch . . . . .	PM-363
Cmvc_Show_Boolean session switch . . . . .	PM-364
Cmvc_Show_Deleted_Objects session switch . . . . .	PM-363
Cmvc_Show_Deleted_Versions session switch . . . . .	PM-363
Cmvc_Show_Display_Position session switch . . . . .	PM-363
Cmvc_Show_Edit_Info session switch . . . . .	PM-363
Cmvc_Show_Field_Default session switch . . . . .	PM-363
Cmvc_Show_Field_Max_Index session switch . . . . .	PM-363
Cmvc_Show_Field_Type session switch . . . . .	PM-363
Cmvc_Show_Frozen session switch . . . . .	PM-364
Cmvc_Show_Hidden_Fields session switch . . . . .	PM-364
Cmvc_Show_Retention session switch . . . . .	PM-364
Cmvc_Show_Unit_State session switch . . . . .	PM-364
Cmvc_Show_Users session switch . . . . .	PM-364
Cmvc_Show_Version_Number session switch . . . . .	PM-364
Cmvc_Uppercase session switch . . . . .	PM-364
Cmvc_Version_Extent session switch . . . . .	PM-364
code	
Compilation.Make renamed procedure . . . . .	LM-151
[Code (All Worlds)] key	
Compilation.Make renamed procedure . . . . .	LM-151
[Code (This World)] key	
Compilation.Make renamed procedure . . . . .	LM-151
Code format (Debug.Memory_Display) . . . . .	DEB-79
[Code Unit] key	
Ada.Code_Unit procedure . . . . .	EST-20
code view . . . . .	PM-16, PM-30, PM-102, PM-262, PM-348
copying in multihost development . . . . .	PM-104
Code_Db subclass . . . . .	LM-17, SJM-15
Code_Segment object manager . . . . .	SMU-11, SMU-58
Code_Unit procedure	
Ada.Code_Unit . . . . .	EST-20
Coded enumeration	
Compilation.Unit_State type . . . . .	LM-166
coded unit state . . . . .	EST-5
incremental compilation . . . . .	EST-8
Col function	
Io.Col . . . . .	TIO-15
Text_Io.Col . . . . .	TIO-167
collating sequence	
Table_Sort_Generic.< generic formal function . . . . .	PT-170
Collect procedure	
Daemon.Collect . . . . .	SMU-14



Collection_Priority subtype	
Daemon.Collection_Priority . . . . .	SMU-15
collections, <i>see</i> Allows_Deallocation, Unchecked_Deallocation	
colon equals (:=) value delimiter . . . . .	LM-18, SJM-15, SMU-8
column . . . . .	DIO-79
current number	
Io.Col function . . . . .	TIO-15
Text_Io.Col function . . . . .	TIO-167
entries	
Table_Formatter.Adjust type . . . . .	ST-93
fill	
Editor.Set.Fill_Column procedure . . . . .	EI-61
formatting	
Table_Formatter package . . . . .	ST-91
number . . . . .	TIO-7
What.Line procedure . . . . .	SJM-263
number of	
Table_Formatter.Number_Of_Columns generic formal object . . . . .	ST-99
set number	
Io.Set_Col procedure . . . . .	TIO-85
Text_Io.Set_Col procedure . . . . .	TIO-203
terminal device characteristic . . . . .	SMU-299
Column_Error	
Io.Exceptions.Layout_Error exception . . . . .	DIO-33, TIO-159
Column_Number subtype	
Window_Io.Column_Number . . . . .	DIO-108
Comb_Ss subclass . . . . .	LM-15, SJM-13
Comb_View subclass . . . . .	LM-15, SJM-13
combined	
subsystems . . . . .	PM-79, PM-97, PM-117, PM-187
views . . . . .	PM-53, PM-79, PM-113, PM-116, PM-187
Combined_Subsystem enumeration	
Cmvc.System_Object_Enum type . . . . .	PM-324
comma (,)	
in set notation . . . . .	LM-13, PM-133, SJM-11, SMU-8
separator . . . . .	LM-18, SJM-16, SMU-8
commands	
contexts . . . . .	DEB-7
execution, access control . . . . .	LM-22
from package !Commands.Common . . . . .	PM-135
from package Cmvc, grouped by topic . . . . .	PM-186
images . . . . .	EST-1, EST-45
commands from package Common . . . . .	EST-47
designation . . . . .	EST-46
executing Command windows . . . . .	EST-46
histories . . . . .	EST-46
key concepts . . . . .	EST-46
library switches . . . . .	EST-47
structure . . . . .	EST-45
unit states . . . . .	EST-46
versions . . . . .	EST-46
input	
Window_Io.Raw.Key_String type . . . . .	DIO-182

Command package . . . . .	EST-53
Command procedure	
What.Command . . . . .	SJM-256
Command windows	
executing . . . . .	EST-46
getting help on . . . . .	EST-125
<i>see also</i> Create_Command	
comment . . . . .	PM-15, PM-405
Editor.Char.Tab_To_Comment procedure . . . . .	EI-15
Editor.Region.Comment procedure . . . . .	EI-46
Editor.Region.Uncomment procedure . . . . .	EI-49
Comment procedure	
Debug.Comment . . . . .	DEB-43
Editor.Region.Comment . . . . .	EI-46
Comment_Column library switch . . . . .	LM-310
Editor.Char.Tab_To_Comment procedure . . . . .	EI-15
commentary messages . . . . .	LM-5, SJM-3
commit . . . . .	PM-135
[Commit] key	
Common.Commit procedure . . . . .	EST-65
Commit procedure	
Common.Commit . . . . .	EST-59, EST-65, PM-27, PM-66, PM-135
Ada images . . . . .	EST-10
command images . . . . .	EST-47
Links package . . . . .	LM-276
Search_List package . . . . .	SJM-210
session switches . . . . .	SJM-248
Switches package . . . . .	LM-315
text images . . . . .	EST-140
windows images . . . . .	EST-156
Commit_Disk enumeration	
Daemon.Log_Threshold type . . . . .	SMU-26
committing images . . . . .	EST-59
Ada images . . . . .	EST-8
text . . . . .	EST-139
Window Directory . . . . .	EST-155
windows images . . . . .	EST-155
Common package . . . . .	EST-3, EST-61, PM-135
Library package . . . . .	LM-203
Links package . . . . .	LM-276
Search_List package . . . . .	SJM-210
session switches . . . . .	SJM-248
Switches package . . . . .	LM-315
What package . . . . .	SJM-253
Comp_Unit subclass . . . . .	LM-16, SJM-14
Compact_Library procedure	
Library.Compact_Library . . . . .	LM-212
compaction	
Daemon.Get_Access_List_Compaction function . . . . .	SMU-17
Daemon.Set_Access_List_Compaction procedure . . . . .	SMU-35
compare, <i>see also</i> Difference, Equal, Merge	

Compare procedure	
File_Uilities.Compare . . . . .	LM-169, LM-172
comparison	
equal	
Io.= function . . . . .	TIO-9
Simple_Status.Equal function . . . . .	PT-135
String_Table.Equal function . . . . .	ST-53
String_Uilities.Equal function . . . . .	ST-72
greater than	
Calendar.> function . . . . .	PT-8
Io.> function . . . . .	TIO-11
String_Uilities.Greater_Than function . . . . .	ST-73
greater than/equal to	
Calendar.>= function . . . . .	PT-8
less than	
Calendar.< function . . . . .	PT-8
Io.< function . . . . .	TIO-10
String_Uilities.Less_Than function . . . . .	ST-75
Table_Sort_Generic.< generic formal function . . . . .	PT-170
less than/equal to	
Calendar.<= function . . . . .	PT-8
of strings	
String_Uilities package . . . . .	ST-69
Compat_Db subclass . . . . .	LM-17, SJM-15
compatibility . . . . .	PM-88
database . . . . .	LM-90, LM-104, PM-105, PM-108, PM-344, PM-346, PM-351, PM-354, PM-356, PM-358
compatible . . . . .	PM-1, PM-11
Compatible enumeration	
Check.Status . . . . .	PM-179
compilation	
access control . . . . .	LM-22, LM-129
incremental . . . . .	EST-7
coded units . . . . .	EST-8
installed units . . . . .	EST-7
management . . . . .	LM-4
multiple subsystems . . . . .	PM-51
states, <i>see</i> unit states	
subsystems . . . . .	LM-130
Compilation package . . . . .	EST-6, LM-129
compile	
Ada.Code_Unit procedure . . . . .	EST-20
Ada.Install_Unit procedure . . . . .	EST-31
Common.Promote procedure . . . . .	EST-91
Compilation.Make renamed procedure . . . . .	LM-151
Compilation.Promote procedure . . . . .	LM-157
Compile procedure	
Compilation.Compile . . . . .	LM-136
complete, <i>see</i> Done	
[Complete] key	
Common.Complete procedure . . . . .	EST-67
Complete procedure	
Common.Complete . . . . .	EST-67, PM-192
Ada images . . . . .	EST-10
command images . . . . .	EST-47

Complete procedure ( <i>continued</i> )	
Common.Complete ( <i>continued</i> )	
Library package . . . . .	LM-203
Redo procedure . . . . .	EST-49
Composite_Name subtype	
Switches.Composite_Name . . . . .	LM-322
compressed output . . . . .	LM-178
concurrency . . . . .	DIO-5, TIO-5
Concurrent_Map_Generic generic package . . . . .	PT-9
condition	
Log.Put_Condition procedure . . . . .	SJM-47
Simple_Status.Create_Condition procedure . . . . .	PT-132
Simple_Status.Create_Condition_Name function . . . . .	PT-133
condition handling	
Simple_Status package . . . . .	PT-127
Condition subtype	
Program.Condition . . . . .	SJM-177
Condition type	
Simple_Status.Condition . . . . .	PT-127, PT-129
Condition_Class type	
Daemon.Condition_Class . . . . .	SMU-16
Log_Threshold type . . . . .	SMU-26
Set_Log_Threshold procedure . . . . .	SMU-37
Simple_Status.Condition_Class . . . . .	PT-130
Condition_Name type	
Simple_Status.Condition_Name . . . . .	PT-131
Config subclass . . . . .	LM-17, SJM-15
configuration . . . . .	PM-3, PM-9, PM-30, PM-406
defined . . . . .	PM-7
images . . . . .	PM-188, PM-189
management . . . . .	PM-3, PM-9
defined . . . . .	PM-1
configuration object . . . . .	PM-32, PM-212
building a view from . . . . .	PM-50
deleting . . . . .	PM-49
releasing configurations . . . . .	PM-30
System_Uilities.System_Boot_Configuration function . . . . .	SMU-265
Configuration library switch . . . . .	LM-310
Configuration object manager . . . . .	SMU-11, SMU-58
Connect procedure	
Job.Connect . . . . .	SJM-20
cons, <i>see</i> Make	
consistency	
checking	
Daemon.Get_Consistency_Checking function . . . . .	SMU-18
Daemon.Set_Consistency_Checking procedure . . . . .	SMU-36
in imports . . . . .	PM-78
Consistent_Breaking library switch . . . . .	LM-310
Console_Print enumeration	
Daemon.Log_Threshold type . . . . .	SMU-26

Constraint_Error exception	
Bounded_String package	ST-1
Append procedure	ST-3
Char_At function	ST-4
Copy procedure	ST-5
Delete procedure	ST-7
Extract function	ST-8
Insert procedure	ST-13
Move procedure	ST-16
Replace procedure	ST-18
Set_Length procedure	ST-19
Value function	ST-22
Concurrent_Map_Generic generic package	PT-9
Cardinality function	PT-10
Copy procedure	PT-11
Define procedure	PT-12
Eval function	PT-15
Find procedure	PT-17
Init procedure	PT-19
Is_Empty function	PT-22
Make_Empty procedure	PT-25
Undefine procedure	PT-33
Debug package	DEB-57
List_Generic generic package	PT-49
First function	PT-52
Rest function	PT-63
Set_First procedure	PT-64
Set_Rest procedure	PT-65
Map_Generic generic package	PT-67
Cardinality function	PT-68
Copy procedure	PT-69
Define procedure	PT-70
Eval function	PT-73
Find procedure	PT-75
Init procedure	PT-77
Is_Empty function	PT-80
Make_Empty procedure	PT-83
Undefine procedure	PT-91
Queue_Generic generic package	PT-95
Delete procedure	PT-98
First function	PT-101
Set_Generic generic package	PT-111
Stack_Generic generic package	
Next procedure	PT-153
Value function	PT-159
Standard.Constraint_Error	PT-161
String_Map_Generic generic package	ST-25
Cardinality function	ST-26
Copy procedure	ST-27
Define procedure	ST-28
Eval function	ST-30
Find procedure	ST-31
Init procedure	ST-33
Is_Empty function	ST-36
Make_Empty procedure	ST-39
Undefine procedure	ST-46
String_Table package	ST-49
Char_At function	ST-51
Length function	ST-60
System_Uilities package	SMU-197
Job_Name function	SMU-233

Constraint_Error exception ( <i>continued</i> )	
Time_Uilities package . . . . .	PT-175
Value function . . . . .	PT-205
Unbounded_String package . . . . .	ST-103
Char_At function . . . . .	ST-106
Delete procedure . . . . .	ST-110
Extract function . . . . .	ST-111
Unchecked_Conversion generic function	
Unchecked_Conversion function . . . . .	PT-213
Unchecked_Conversions package	
Convert_From_Byte_String function . . . . .	PT-233
Unchecked_Conversions.Unchecked_Conversion_Package generic package	
Convert function . . . . .	PT-227
contents	
Time_Uilities.Image_Contents type . . . . .	PT-192
Contents function	
Cmvc_Hierarchy.Contents . . . . .	PM-333
context . . . . .	DEB-7
characters, <i>see</i> special characters	
control . . . . .	DEB-7
Debug.Context procedure . . . . .	DEB-44
evaluation . . . . .	DEB-7, DEB-15, DEB-99
Debug.Catch procedure . . . . .	DEB-39
Debug.Context procedure . . . . .	DEB-45
Context procedure	
Debug.Context . . . . .	DEB-44
Context_Type procedure . . . . .	DEB-48
Library.Context . . . . .	LM-214
Context subclass . . . . .	LM-16, SJM-14
Context_Name subtype	
Library.Context_Name . . . . .	LM-215
Context_Type type	
Debug.Context_Type . . . . .	DEB-48
Contexts enumeration	
Debug.State_Type type . . . . .	DEB-126
Continue procedure	
Text.Continue . . . . .	EST-147
Block procedure . . . . .	EST-146
control	
characters, insert	
Editor.Char.Quote procedure . . . . .	EI-12, EI-14
context . . . . .	DEB-7, DEB-11, DEB-44
System_Uilities.Flow_Control function . . . . .	SMU-216
System_Uilities.Receive_Flow_Control function . . . . .	SMU-252
Terminal.Set_Flow_Control procedure . . . . .	SMU-311
Terminal.Set_Receive_Flow_Control procedure . . . . .	SMU-318
Control enumeration	
Debug.Context_Type type . . . . .	DEB-48
Control format (Debug.Memory_Display) . . . . .	DEB-79
controlled . . . . .	PM-6
objects	
accessing concurrently . . . . .	PM-43
deleting . . . . .	PM-35
editing . . . . .	PM-26

controlled ( <i>continued</i> )	
objects ( <i>continued</i> )	
library-management operations . . . . .	PM-35
moving . . . . .	PM-35
withdrawing . . . . .	PM-35
conversion	
between different data types	
Unchecked_Conversion generic function . . . . .	PT-209
Unchecked_Conversion.Unchecked_Conversion function . . . . .	PT-212
Unchecked_Conversions package . . . . .	PT-219
Unchecked_Conversions.Unchecked_Conversion_Package generic package . . . . .	PT-225
Unchecked_Conversions.Unchecked_Conversion_Package.Convert function . . . . .	PT-226
from byte string	
Unchecked_Conversions.Convert_From_Byte_String function . . . . .	PT-232
Unchecked_Conversions.Convert_From_Byte_String generic function . . . . .	PT-231
from text to Ada object	
Compilation.Parse procedure . . . . .	LM-155
numeric . . . . .	DEB-16
Debug.Convert procedure . . . . .	DEB-49
of strings, case	
String_Uilities package . . . . .	ST-69
to byte string	
Unchecked_Conversions.Convert_To_Byte_String function . . . . .	PT-238
Unchecked_Conversions.Convert_To_Byte_String generic function . . . . .	PT-237
<i>see also</i> Image functions and Value functions for types of particular interest	
Convert function	
Io.Convert . . . . .	TIO-16, TIO-17
Time_Uilities.Convert . . . . .	PT-178
Unchecked_Conversions.Unchecked_Conversion_Package.Convert . . . . .	PT-226
Window_Io.Raw.Convert . . . . .	DIO-174, DIO-175
Convert procedure	
Debug.Convert . . . . .	DEB-16, DEB-49
Io.Convert . . . . .	TIO-18
Profile.Convert . . . . .	SJM-83
Value function . . . . .	SJM-167
Convert_From_Byte_String function	
Unchecked_Conversions.Convert_From_Byte_String . . . . .	PT-232
Convert_To_Byte_String function . . . . .	PT-238
Source generic formal type . . . . .	PT-240
Target generic formal type . . . . .	PT-235
Convert_From_Byte_String generic function	
Unchecked_Conversions.Convert_From_Byte_String . . . . .	PT-231
Convert_Old_Subsystem procedure	
Cmvc_Maintenance.Convert_Old_Subsystem . . . . .	PM-342
Convert_Time function	
Time_Uilities.Convert_Time . . . . .	PT-179
Convert_To_Byte_String function	
Unchecked_Conversions.Convert_To_Byte_String . . . . .	PT-238
Source generic formal type . . . . .	PT-240
Target generic formal type . . . . .	PT-235
Convert_To_Byte_String generic function	
Unchecked_Conversions.Convert_To_Byte_String . . . . .	PT-237
Convert_From_Byte_String function . . . . .	PT-232
coordinating development in a subsystem . . . . .	PM-37

copy	
Archive package . . . . .	LM-87
Library.Move renamed procedure . . . . .	LM-250
Copy procedure	
Archive.Copy . . . . .	LM-87, LM-90, <i>LM-100</i> , PM-103, PM-109
Cmvc_Maintenance.Make_Primary procedure . . . . .	PM-351
Cmvc_Maintenance.Make_Secondary procedure . . . . .	PM-354
Cmvc_Maintenance.Update_Cdb procedure . . . . .	PM-358
Bounded_String.Copy . . . . .	ST-5
Cmvc.Copy . . . . .	PM-222
Common.Object.Copy . . . . .	EST-104
Ada images . . . . .	EST-16
Library package . . . . .	LM-206
Links package . . . . .	LM-278
session switches . . . . .	SJM-250
Switches package . . . . .	LM-316
text images . . . . .	EST-142
Concurrent_Map_Generic.Copy . . . . .	PT-11
Editor.Line.Copy . . . . .	EI-31, EI-32
Editor.Region.Copy . . . . .	EI-46
Editor.Window.Copy . . . . .	EI-65
Library.Copy . . . . .	<i>LM-216</i> , PM-23, PM-43, PM-93
Links.Copy . . . . .	LM-282
Log.Copy . . . . .	SJM-35
Map_Generic.Copy . . . . .	PT-69
Queue_Generic.Copy . . . . .	PT-97
Set_Generic.Copy . . . . .	PT-113
Stack_Generic.Copy . . . . .	PT-144
String_Map_Generic.Copy . . . . .	ST-27
Unbounded_String.Copy . . . . .	ST-107
copy then delete, <i>see</i> Move	
Copy_Top procedure	
Editor.Hold_Stack.Copy_Top . . . . .	EI-21
Editor.Mark.Copy_Top . . . . .	EI-4, EI-41
Editor.Screen.Copy_Top . . . . .	EI-7, EI-51, EI-52
core editor (Ce) job . . . . .	SMU-132
count	
Direct_Io.Positive_Count subtype . . . . .	DIO-23
input	
System_Uilities.Input_Count function . . . . .	SMU-229
Io.Positive_Count subtype . . . . .	TIO-71
output	
System_Uilities.Output_Count function . . . . .	SMU-245
page	
System_Uilities.Get_Page_Counts procedure . . . . .	SMU-220
set retention	
Library.Set_Retention_Count procedure . . . . .	LM-259
Text_Io.Positive_Count subtype . . . . .	TIO-197
Time_Uilities.Day_Count type . . . . .	PT-182
Window_Io.Positive_Count subtype . . . . .	DIO-160
<i>see also</i> Cardinality	
Count subtype	
Io.Count . . . . .	TIO-7, TIO-19
Window_Io.Count . . . . .	DIO-109
Count type	
Direct_Io.Count . . . . .	DIO-9
Text_Io.Count . . . . .	TIO-165, TIO-168



Cpu	
Scheduler.Get_Cpu_Priority function	SMU-150
Scheduler.Get_Cpu_Time_Used function	SMU-151
Cpu function	
System_Uilities.Cpu	SMU-205
CPU scheduling parameters	SMU-171, SMU-173
CPU time	
System_Uilities.Elapsed function	SMU-213
Cpu_Priority subtype	
Scheduler.Cpu_Priority	SMU-141
crash	
Operator.Explain_Crash procedure	SMU-74
create	
access	LM-21
directories	
Library.Create procedure	LM-220
Library.Create_Directory renamed procedure	LM-222
entry, <i>see</i> Define	
libraries	
Library.Create procedure	LM-220
Library.Create_Directory renamed procedure	LM-222
Library.Create_World renamed procedure	LM-226
new joined objects	PM-42
one, <i>see</i> Allocate	
path	PM-47
space for, <i>see</i> Allocate	
spec view	PM-58
subpath	PM-37
units	
Library.Create_Unit renamed procedure	LM-224
worlds	
Library.Create procedure	LM-220
Library.Create_World renamed procedure	LM-226
[Create Ada] key	
Common.Object.Insert procedure	EST-108
<i>see also</i> insertion points	
[Create Body] key	
Ada.Create_Body procedure	EST-22
[Create Command] key	
Common.Create_Command procedure	EST-68
Create constant	
Access_List.Create	LM-34
Access_List_Tools.Create	LM-64
[Create Directory] key	
Library.Create_Directory renamed procedure	LM-222
[Create Private Part] key	
Ada.Create_Private procedure	EST-24
Create procedure	
Activity.Create	PM-66, PM-82, PM-144
Direct_Io.Create	DIO-10
Form function	DIO-17
Io.Create	TIO-20
Form function	TIO-40

Create procedure ( <i>continued</i> )	
Library.Create . . . . .	LM-220
Create_Directory renamed procedure . . . . .	LM-222
Create_Unit renamed procedure . . . . .	LM-224
Create_World renamed procedure . . . . .	LM-226
Polymorphic_Sequential_Io.Create . . . . .	DIO-42
Form function . . . . .	DIO-48
Queue.Create . . . . .	SMU-101
Default procedure . . . . .	SMU-102
Register procedure . . . . .	SMU-123
Sequential_Io.Create . . . . .	DIO-63
Form function . . . . .	DIO-70
Switches.Create . . . . .	LM-323
Text.Create . . . . .	EST-148, PM-71, PM-72
Image_Kind type . . . . .	EST-150
Text_Io.Create . . . . .	TIO-169
Form function . . . . .	TIO-180
Window_Io.Create . . . . .	DIO-110
Form function . . . . .	DIO-124
Work_Order.Create . . . . .	PM-367
[Create Text] key	
Text.Create procedure . . . . .	EST-148
[Create World] key	
Library.Create_World renamed procedure . . . . .	LM-226
Create_Body procedure	
Ada.Create_Body . . . . .	EST-22
Create_Command procedure	
Common.Create_Command . . . . .	EST-45, EST-68, PM-136
Ada images . . . . .	EST-10
command images . . . . .	EST-48
Debugger . . . . .	DEB-5
Help . . . . .	EST-126
Library package . . . . .	LM-204
Links package . . . . .	LM-276
menu images . . . . .	EST-133
Search_List package . . . . .	SJM-210
session switches . . . . .	SJM-248
Switches package . . . . .	LM-315
text images . . . . .	EST-141
What package . . . . .	SJM-253
windows images . . . . .	EST-156
xref images . . . . .	EST-161
Create_Condition procedure	
Simple_Status.Create_Condition . . . . .	PT-132
Create_Condition_Name function	
Simple_Status.Create_Condition_Name . . . . .	PT-133
Create_Directory renamed procedure	
Library.Create_Directory . . . . .	LM-222
Create_Empty_Note_Window procedure	
Cmvc.Create_Empty_Note_Window . . . . .	PM-232
Cmvc.Append_Notes procedure . . . . .	PM-210
Cmvc.Put_Notes procedure . . . . .	PM-292
Create_Field procedure	
Work_Order.Create_Field . . . . .	PM-369
Create_Group procedure	
Operator.Create_Group . . . . .	SMU-61

Create_Internal_Links library switch . . . . .	LM-7, LM-310
Create_Job procedure	
Program.Create_Job . . . . .	LM-19, <i>SJM-178</i>
Condition subtype . . . . .	SJM-177
Current function . . . . .	SJM-183, <i>SJM-184</i>
Run procedure . . . . .	SJM-188
Run_Job procedure . . . . .	SJM-190, <i>SJM-193</i>
Started_Successfully function . . . . .	SJM-195
Create_List procedure	
Work_Order.Create_List . . . . .	<i>PM-371</i>
Create_Private procedure	
Ada.Create_Private . . . . .	<i>EST-24</i>
Create_Session procedure	
Operator.Create_Session . . . . .	SMU-53, <i>SMU-63</i>
Create_Subprogram_Specs library switch . . . . .	LM-310
Create_Time enumeration	
Library.Field type . . . . .	LM-239
Create_Unit renamed procedure	
Library.Create_Unit . . . . .	<i>LM-224</i>
Create_User procedure	
Operator.Create_User . . . . .	<i>SMU-64</i>
Create_Venture procedure	
Work_Order.Create_Venture . . . . .	<i>PM-372</i>
Create_World renamed procedure	
Library.Create_World . . . . .	<i>LM-226</i>
Creation_Mode type	
Activity.Creation_Mode . . . . .	<i>PM-146</i>
Creator enumeration	
Library.Field type . . . . .	LM-239
cross-development	
using CDFs with subsystems . . . . .	PM-111
cross-reference information	
Xref package . . . . .	LM-341
current	
cursor position . . . . .	DIO-79, DIO-81
index	
Direct_Io.End_Of_File function . . . . .	DIO-14
macro . . . . .	EI-37
task . . . . .	DEB-7
Debug.Display procedure . . . . .	DEB-54
Current function	
Program.Current . . . . .	<i>SJM-183</i>
Current procedure	
Activity.Current . . . . .	PM-67, <i>PM-147</i>
Current renamed function	
Terminal.Current . . . . .	<i>SMU-303</i>
Current_Debugger procedure	
Debug.Current_Debugger . . . . .	<i>DEB-50</i>
Current_Directory constant	
Compilation.Current_Directory . . . . .	<i>LM-138</i>

Current_Error function	
Io.Current_Error . . . . .	TIO-22
Current_Input function	
Io.Current_Input . . . . .	TIO-23
Text_Io.Current_Input . . . . .	TIO-171
Current_Output constant	
File_Uutilities.Current_Output . . . . .	LM-175
Current_Output function	
Io.Current_Output . . . . .	TIO-24
Text_Io.Current_Output . . . . .	TIO-172
cursor	
current position . . . . .	DIO-79, DIO-81
designation . . . . .	EST-57
Ada images . . . . .	EST-4
movement . . . . .	DIO-80, EI-2, EI-17
Editor.Cursor.Backward procedure . . . . .	EI-18
Editor.Cursor.Down procedure . . . . .	EI-18
Editor.Cursor.Forward procedure . . . . .	EI-18
Editor.Cursor.Left procedure . . . . .	EI-18
Editor.Cursor.Next procedure . . . . .	EI-19
Editor.Cursor.Previous procedure . . . . .	EI-19
Editor.Cursor.Right procedure . . . . .	EI-19
Editor.Cursor.Up procedure . . . . .	EI-20
moving	
Window_Io.Move_Cursor procedure . . . . .	DIO-148
planar movement . . . . .	EI-2
positioning	
Window_Io.Position_Cursor procedure . . . . .	DIO-158
relative movement . . . . .	EI-2, EI-3
reporting	
Window_Io.Report_Cursor procedure . . . . .	DIO-163
stream operations . . . . .	EI-3
Cursor package	
Editor.Cursor . . . . .	EI-17
<CURSOR> special name . . . . .	DEB-3, EST-58, EST-131, EST-155, LM-8, LM-130, LM-170, LM-199, LM-308, PM-128, SJM-5, SMU-2
Cursor_Bottom_Offset session switch . . . . .	SJM-231
Cursor_Left_Offset session switch . . . . .	SJM-231
Cursor_Right_Offset session switch . . . . .	SJM-231
Cursor_Top_Offset session switch . . . . .	SJM-231
Cursor_Transpose_Moves session switch . . . . .	SJM-231
Editor.Char.Transpose procedure . . . . .	EI-15
Editor.Line.Transpose procedure . . . . .	EI-34
Editor.Window.Transpose procedure . . . . .	EI-68
Editor.Word.Transpose procedure . . . . .	EI-71
customizing session behavior . . . . .	EI-7

D

daemon . . . . .	SMU-11
when last run	
Daemon.Last_Run function . . . . .	SMU-25
when next run	
Daemon.Next_Scheduled function . . . . .	SMU-28

Daemon package . . . . .	SMU-11
dailies, <i>see</i> System_Backup package	
Daily client . . . . .	SMU-12, SMU-22, SMU-48
daily message	
What.Message procedure . . . . .	SJM-267
data	
conversion between different types	
Unchecked_Conversion generic function . . . . .	PT-209
Unchecked_Conversion.Unchecked_Conversion function . . . . .	PT-212
Unchecked_Conversions package . . . . .	PT-219
Unchecked_Conversions.Unchecked_Conversion_Package generic package . . . . .	PT-225
Unchecked_Conversions.Unchecked_Conversion_Package.Convert function . . . . .	PT-226
conversion from byte string	
Unchecked_Conversions.Convert_From_Byte_String function . . . . .	PT-232
Unchecked_Conversions.Convert_From_Byte_String generic function . . . . .	PT-231
conversion to byte string	
Unchecked_Conversions.Convert_To_Byte_String function . . . . .	PT-238
Unchecked_Conversions.Convert_To_Byte_String generic function . . . . .	PT-237
structures, referencing . . . . .	DEB-21
tapes . . . . .	SMU-191
Data file . . . . .	LM-88, LM-122
Data format (Debug.Memory_Display) . . . . .	DEB-79
Data_Error exception	
Direct_Io generic package	
Read procedure . . . . .	DIO-24
Io package	
Get procedure . . . . .	TIO-46, TIO-48, TIO-49
Io.Enumeration_Io generic package	
Get procedure . . . . .	TIO-113, TIO-114
Io.Fixed_Io generic package	
Get procedure . . . . .	TIO-124, TIO-125
Io.Float_Io generic package	
Get procedure . . . . .	TIO-136, TIO-137
Io.Integer_Io generic package	
Get procedure . . . . .	TIO-147, TIO-148
Io.Exceptions.Data_Error . . . . .	DIO-30, TIO-156
Polymorphic_Sequential_Io package . . . . .	DIO-39
Polymorphic_Sequential_Io.Operations package	
Element_Type generic formal type . . . . .	DIO-56
Read procedure . . . . .	DIO-57
Write procedure . . . . .	DIO-58
Sequential_Io package	
Element_Type generic formal type . . . . .	DIO-66
Read procedure . . . . .	DIO-76
Text_Io.Enumeration_Io generic package	
Get procedure . . . . .	TIO-221, TIO-222
Text_Io.Fixed_Io generic package	
Get procedure . . . . .	TIO-232, TIO-233
Text_Io.Float_Io generic package	
Get procedure . . . . .	TIO-244, TIO-245
Text_Io.Integer_Io generic package	
Get procedure . . . . .	TIO-255, TIO-256
date	
formats	
Operator.Set_System_Time procedure . . . . .	SMU-84
What.Time procedure . . . . .	SJM-270

Date enumeration	
Profile.Log_Prefix type . . . . .	SJM-114
Date_Format type	
Time_Uilities.Date_Format . . . . .	PT-180
Date_Only enumeration	
Time_Uilities.Image_Contents type . . . . .	PT-192
Day constant	
Time_Uilities.Day . . . . .	PT-181
Day function	
Calendar.Day . . . . .	PT-6
Day_Count type	
Time_Uilities.Day_Count . . . . .	PT-182
Day_Duration subtype	
Calendar.Day_Duration . . . . .	PT-6
Day_Month_Year enumeration	
Time_Uilities.Date_Format type . . . . .	PT-180
Day_Number subtype	
Calendar.Day_Number . . . . .	PT-6
Day_Of_Week function	
Time_Uilities.Day_Of_Week . . . . .	PT-183
Days type	
Time_Uilities.Days . . . . .	PT-184
DDB	
client . . . . .	SMU-13, SMU-18, SMU-20, SMU-27
object manager . . . . .	SMU-11, SMU-58
deallocation	
Allows_Deallocation generic function . . . . .	PT-1
Allows_Deallocation.Allows_Deallocation function . . . . .	PT-2
Unchecked_Deallocation generic procedure . . . . .	PT-241
Unchecked_Deallocation.Unchecked_Deallocation procedure . . . . .	PT-246
Debug package . . . . .	DEB-2, DEB-29
Debug procedure	
Command.Debug . . . . .	DEB-2, DEB-7, EST-54
Debug_Addresses session switch . . . . .	SJM-232
Debug_Break_At_Creation session switch . . . . .	SJM-232
Debug_Declaration_Display session switch . . . . .	SJM-232
Debug_Delete_Temporary_Breaks session switch . . . . .	SJM-232
Debug_Display_Count session switch . . . . .	SJM-232
Debug_Display_Creation session switch . . . . .	SJM-232
Debug_Display_Level session switch . . . . .	SJM-232
Debug_Echo_Commands session switch . . . . .	SJM-232
Debug_Element_Count session switch . . . . .	SJM-232
Debug_First_Element session switch . . . . .	SJM-233
Debug_Freeze_Tasks session switch . . . . .	SJM-233
Debug_History_Count session switch . . . . .	SJM-233
Debug_History_Entries session switch . . . . .	SJM-233
Debug_History_Start session switch . . . . .	SJM-233

Debug_Include_Packages session switch . . . . .	SJM-233
Debug_Interpret_Control_Words session switch . . . . .	SJM-233
Debug_Kill_Old_Jobs session switch . . . . .	SJM-233
Debug_Machine_Level session switch . . . . .	SJM-233
Debug_Memory_Count session switch . . . . .	SJM-234
Debug_Msg enumeration	
Profile.Msg_Kind type . . . . .	SJM-117
Debug_No_History_Timestamps session switch . . . . .	SJM-234
Debug_Off procedure	
Debug.Debug_Off . . . . .	DEB-51
Command.Debug procedure . . . . .	EST-54
Debug_Require_Debug_Off session switch . . . . .	SJM-234
Option type . . . . .	DEB-88
Debug_Tools.Debug_Off . . . . .	DEB-154
Debug.Debug_Off procedure . . . . .	DEB-51
Debug_On procedure . . . . .	DEB-155
Debug_On procedure	
Debug_Tools.Debug_On . . . . .	DEB-155
Command.Debug . . . . .	EST-54
Debug_Off procedure . . . . .	DEB-154
Debug_Optimize_Generic_History session switch . . . . .	SJM-234
Debug_Permanent_Breakpoints session switch . . . . .	SJM-234
Debug_Pointer_Level session switch . . . . .	SJM-234
Debug_Put_Locals session switch . . . . .	SJM-234
Debug_Qualify_Stack_Names session switch . . . . .	SJM-234
Debug_Require_Debug_Off session switch . . . . .	SJM-234
Debug_Save_Exceptions session switch . . . . .	SJM-235
Debug_Show_Location session switch . . . . .	SJM-235
Debug_Stack_Count session switch . . . . .	SJM-235
Debug_Stack_Start session switch . . . . .	SJM-235
Debug_Timestamps session switch . . . . .	SJM-235
Debug_Tools package . . . . .	DEB-2, DEB-16, DEB-151
Debugger	
current	
Debug.Current_Debugger procedure . . . . .	DEB-50
Debugger facilities . . . . .	DEB-6
programmatic . . . . .	DEB-16
show	
Debug.Show procedure . . . . .	DEB-115
debugger images . . . . .	EST-1
Debugger window . . . . .	DEB-2
designation . . . . .	DEB-3
selection . . . . .	DEB-3
write message to	
Debug.Comment procedure . . . . .	DEB-43
Debug_Tools.Message procedure . . . . .	DEB-163
[Debugger Window] key	
Debug.Current_Debugger procedure . . . . .	DEB-3, DEB-50

Debugger_Initialization procedure . . . . .	DEB-14, DEB-16
Debug.Put procedure . . . . .	DEB-101
Debug_Tools.Register generic procedure . . . . .	DEB-165, DEB-167
debugging	
allow to run	
Debug.Release procedure . . . . .	DEB-106
argument prefixes . . . . .	DEB-4
assign nickname	
Debug.Set_Task_Name procedure . . . . .	DEB-112
Debug_Tools.Set_Task_Name procedure . . . . .	DEB-179
automatic source display . . . . .	DEB-3
breakpoints . . . . .	DEB-10
cancel breakpoint	
Debug.Forget procedure . . . . .	DEB-65
Debug.Remove procedure . . . . .	DEB-107
cancel stopping on exception	
Debug.Propagate procedure . . . . .	DEB-96
change value of object	
Debug.Modify procedure . . . . .	DEB-81
clear option flag	
Debug.Disable renamed procedure . . . . .	DEB-52
collect history	
Debug.Take_History procedure . . . . .	DEB-132
command contexts . . . . .	DEB-7
commands from package Common . . . . .	DEB-5
create breakpoint	
Debug.Break procedure . . . . .	DEB-32
current debugger	
Debug.Current_Debugger procedure . . . . .	DEB-50
Debugger facilities . . . . .	DEB-6
Debugger window . . . . .	DEB-1
define Debugger state	
Debug.State_Type type . . . . .	DEB-126
define event class	
Debug.Trace_Event type . . . . .	DEB-146
define events to stop task	
Debug.Stop_Event type . . . . .	DEB-130
designation . . . . .	DEB-3
display a variable	
Debug.Put procedure . . . . .	DEB-100
display absolute memory	
Debug.Memory_Display procedure . . . . .	DEB-79
display code segment address	
Debug.Location_To_Address procedure . . . . .	DEB-77
display current location	
Debug_Tools.Ada_Location function . . . . .	DEB-152
display information about Debugger facilities	
Debug.Show procedure . . . . .	DEB-115
display source	
Debug.Source procedure . . . . .	DEB-121
display stack	
Debug.Stack procedure . . . . .	DEB-123
display task	
Debug.Task_Display procedure . . . . .	DEB-136
display task history	
Debug.History_Display procedure . . . . .	DEB-68
display task information	
Debug.Information procedure . . . . .	DEB-73
display task name	
Debug_Tools.Get_Task_Name function . . . . .	DEB-161
display/modify program data . . . . .	DEB-14



debugging ( <i>continued</i> )	
Editor . . . . .	DEB-1
exception location	
Debug_Tools.Get_Raise_Location function . . . . .	DEB-159
exception name	
Debug_Tools.Get_Exception_Name function . . . . .	DEB-157
exception trapping . . . . .	DEB-12
exceptions	
Debug.Exception_Name subtype . . . . .	DEB-57
Debug.Exception_To_Name procedure . . . . .	DEB-60
flags . . . . .	DEB-16
Debug.Flag procedure . . . . .	DEB-63
Debug.Numeric type . . . . .	DEB-84
Debug.Option type . . . . .	DEB-86
Debug.Set_Value procedure . . . . .	DEB-114
history facility . . . . .	DEB-12
is program being debugged	
Debug_Tools.Debugging function . . . . .	DEB-156
job . . . . .	DEB-2
kill job being debugged	
Debug.Kill procedure . . . . .	DEB-76
numeric conversion . . . . .	DEB-16
Debug.Convert procedure . . . . .	DEB-49
numeric flags . . . . .	DEB-16
options . . . . .	DEB-16
pathnames . . . . .	DEB-17
Debug.Path_Name subtype . . . . .	DEB-90
program . . . . .	DEB-2
programmatic access to Debugger facilities . . . . .	DEB-16
programmatic breakpoint	
Debug_Tools.User_Break procedure . . . . .	DEB-185
programmatic interface	
Debug_Tools package . . . . .	DEB-151
quiet startup . . . . .	DEB-16
Debug.Reset_Defaults procedure . . . . .	DEB-108
Rational Editor . . . . .	DEB-1
reactivate breakpoint	
Debug.Activate procedure . . . . .	DEB-30
referencing data structures . . . . .	DEB-21
referencing generic instantiations . . . . .	DEB-24
referencing library units . . . . .	DEB-20
referencing overloaded subprograms . . . . .	DEB-23
referencing programs . . . . .	DEB-22
remove stepping	
Debug.Clear_Stepping procedure . . . . .	DEB-42
resume execution	
Debug.Execute procedure . . . . .	DEB-61
Debug.Xecute procedure . . . . .	DEB-148
selection . . . . .	DEB-3
send trace output to file	
Debug.Trace_To_File procedure . . . . .	DEB-147
session switches . . . . .	DEB-4, SJM-228
set context	
Debug.Context procedure . . . . .	DEB-44
set exception breakpoint	
Debug.Catch procedure . . . . .	DEB-36
set option flag	
Debug.Enable procedure . . . . .	DEB-56
set trace	
Debug.Trace procedure . . . . .	DEB-142

debugging ( <i>continued</i> )	
show source	
Debug.Display procedure . . . . .	DEB-53
show source location	
Debug.Address_To_Location procedure . . . . .	DEB-31
special characters . . . . .	DEB-18
special display	
Debug_Tools.Register generic procedure . . . . .	DEB-165
Debug_Tools.Register procedure . . . . .	DEB-175
start	
Debug_Tools.Debug_On procedure . . . . .	DEB-155
state . . . . .	DEB-1
step	
Debug.Run procedure . . . . .	DEB-109
stepping . . . . .	DEB-13
stop	
Debug.Debug_Off procedure . . . . .	DEB-51
Debug_Tools.Debug_Off procedure . . . . .	DEB-154
stop special display	
Debug_Tools.Un_Register generic procedure . . . . .	DEB-181
Debug_Tools.Un_Register procedure . . . . .	DEB-183
stop task execution	
Debug.Hold procedure . . . . .	DEB-71
Debug.Stop procedure . . . . .	DEB-128
substituting your own data display routine	
Debug_Tools.Register generic procedure . . . . .	DEB-165
task category	
Debug.Task_Category type . . . . .	DEB-135
task name	
Debug.Task_Name subtype . . . . .	DEB-140
tasks . . . . .	DEB-7
tracing facility . . . . .	DEB-12
unqualified names . . . . .	DEB-20
write message to Debugger window	
Debug.Comment procedure . . . . .	DEB-43
Debug_Tools.Message procedure . . . . .	DEB-163
Debugging function	
Debug_Tools.Debugging . . . . .	DEB-156
decimal point	
after	
Io.Fixed_Io.Default_Aft constant . . . . .	TIO-120
Io.Float_Io.Default_Aft constant . . . . .	TIO-132
Text_Io.Fixed_Io.Default_Aft constant . . . . .	TIO-228
Text_Io.Float_Io.Default_Aft constant . . . . .	TIO-240
before	
Io.Fixed_Io.Default_Fore constant . . . . .	TIO-122
Io.Float_Io.Default_Fore constant . . . . .	TIO-134
Text_Io.Fixed_Io.Default_Fore constant . . . . .	TIO-230
Text_Io.Float_Io.Default_Fore constant . . . . .	TIO-242
Decl_List subclass . . . . .	LM-16, SJM-14
Declaration enumeration	
Library.Field type . . . . .	LM-239
declaration number . . . . .	PM-105
Declaration_Display enumeration	
Debug.Option type . . . . .	DEB-86

Def procedure	
Cmvc.Def . . . . .	PM-192, PM-234
default	
access list . . . . .	LM-1, LM-21
activity . . . . .	PM-65
add	
Access_List.Add_Default procedure . . . . .	LM-33
get	
Access_List_Tools.Get_Default function . . . . .	LM-70
Access_List_Tools.Get_Default procedure . . . . .	LM-72
Profile.Get_Default function . . . . .	SJM-103, SJM-104
include	
Profile.Include_In_Default procedure . . . . .	SJM-110
reset	
Debug.Reset_Defaults procedure . . . . .	DEB-108
Search_List.Reset_To_System_Default procedure . . . . .	SJM-220
response profile . . . . .	DIO-6, LM-5, PM-128, SJM-3, SJM-75, SMU-2, SMU-55, TIO-6
retention count	
Library.Default_Keep_Versions constant . . . . .	LM-229
Library.Set_Retention_Count procedure . . . . .	LM-259
set	
Access_List.Set_Default procedure . . . . .	LM-46
Access_List_Tools.Set_Default procedure . . . . .	LM-83
Profile.Set_Default procedure . . . . .	SJM-141
Profile.Set_Default_Activity procedure . . . . .	SJM-142
Profile.Set_Default_Filter procedure . . . . .	SJM-143, SJM-145
Profile.Set_Default_Log_File procedure . . . . .	SJM-146
Profile.Set_Default_Prefixes procedure . . . . .	SJM-147
Profile.Set_Default_Reaction procedure . . . . .	SJM-148
Profile.Set_Default_Remote_Passwords procedure . . . . .	SJM-149
Profile.Set_Default_Remote_Sessions procedure . . . . .	SJM-150
Profile.Set_Default_Response procedure . . . . .	SJM-151
Profile.Set_Default_Width procedure . . . . .	SJM-153
storage	
Unbounded_String.Default_Maximum_Length generic formal object . . . . .	ST-109
version . . . . .	EST-58
Library.Default procedure . . . . .	LM-228
Library.Default_Keep_Versions constant . . . . .	LM-229
Wsl limits	
Scheduler.Use_Default_Wsl_Limits procedure . . . . .	SMU-188
Default function	
Work_Order.Default . . . . .	PM-373
Default procedure	
Library.Default . . . . .	LM-228
Queue.Default . . . . .	SMU-102
<DEFAULT> special value . . . . .	DIO-6, LM-5, PM-128, SJM-3, SJM-75, SMU-2, SMU-55, TIO-6
Default_Activity function	
Profile.Default_Activity . . . . .	SJM-84
Default_Aft constant	
Io.Fixed_Io.Default_Aft . . . . .	TIO-120
Io.Float_Io.Default_Aft . . . . .	TIO-132
Text_Io.Fixed_Io.Default_Aft . . . . .	TIO-228
Text_Io.Float_Io.Default_Aft . . . . .	TIO-240
Default_Base constant	
Io.Integer_Io.Default_Base . . . . .	TIO-144
Text_Io.Integer_Io.Default_Base . . . . .	TIO-252

Default_Display procedure	
Access_List.Default_Display . . . . .	LM-35
Default_Exp constant	
Io.Fixed_Io.Default_Exp . . . . .	TIO-121
Io.Float_Io.Default_Exp . . . . .	TIO-133
Text_Io.Fixed_Io.Default_Exp . . . . .	TIO-229
Text_Io.Float_Io.Default_Exp . . . . .	TIO-241
Default_File constant	
Switches.Default_File . . . . .	LM-324
Default_Filter constant	
Profile.Default_Filter . . . . .	SJM-85
Default_Font function	
Window_Io.Default_Font . . . . .	DIO-111
Default_Fore constant	
Io.Fixed_Io.Default_Fore . . . . .	TIO-122
Io.Float_Io.Default_Fore . . . . .	TIO-134
Text_Io.Fixed_Io.Default_Fore . . . . .	TIO-230
Text_Io.Float_Io.Default_Fore . . . . .	TIO-242
Default_Job_Page_Limit session switch . . . . .	SJM-235
System_Uilities.Get_Page_Counts procedure . . . . .	SMU-220
System_Uilities.Set_Page_Limit procedure . . . . .	SMU-261
Default_Keep_Versions constant	
Library.Default_Keep_Versions . . . . .	LM-229
Default_List function	
Work_Order.Default_List . . . . .	PM-375
Default_Log_File constant	
Profile.Default_Log_File . . . . .	SJM-86
Default_Maximum_Length generic formal object	
Unbounded_String.Default_Maximum_Length . . . . .	ST-109
Default_Prefixes constant	
Profile.Default_Prefixes . . . . .	SJM-87
Default_Profile function	
Profile.Default_Profile . . . . .	SJM-75, SJM-88
Get_Default function . . . . .	SJM-103, SJM-104
Default_Reaction constant	
Profile.Default_Reaction . . . . .	SJM-90
Default_Remote_Passwords function	
Profile.Default_Remote_Passwords . . . . .	SJM-91
Default_Remote_Sessions function	
Profile.Default_Remote_Sessions . . . . .	SJM-92
Default_Setting constant	
Io.Enumeration_Io.Default_Setting . . . . .	TIO-110
Text_Io.Enumeration_Io.Default_Setting . . . . .	TIO-218
Default_Venture function	
Work_Order.Default_Venture . . . . .	PM-377
Default_Venture session switch . . . . .	PM-364
Work_Order.Set_Default_Venture procedure . . . . .	PM-394
Default_Width constant	
Io.Enumeration_Io.Default_Width . . . . .	TIO-111
Io.Integer_Io.Default_Width . . . . .	TIO-145

Default_Width constant ( <i>continued</i> )	
Profile.Default_Width . . . . .	SJM-93
Width function . . . . .	SJM-172
Text_Io.Enumeration_Io.Default_Width . . . . .	TIO-219
Text_Io.Integer_Io.Default_Width . . . . .	TIO-253
Define procedure	
Concurrent_Map_Generic.Define . . . . .	PT-12
Editor.Key.Define . . . . .	EI-28
Map_Generic.Define . . . . .	PT-70
String_Map_Generic.Define . . . . .	ST-28
Switches.Define . . . . .	LM-325
[Definition In Place] key	
Common.Definition procedure . . . . .	EST-71
[Definition] key	
Common.Definition procedure . . . . .	EST-71
Definition procedure	
Common.Definition . . . . .	EI-6, EST-9, <i>EST-71</i> , PM-136, PM-192
Ada images . . . . .	EST-11
command images . . . . .	EST-48
Debugger . . . . .	DEB-5
Help . . . . .	EST-126
Library package . . . . .	LM-204
Links package . . . . .	LM-276
menu images . . . . .	EST-133
Search_List package . . . . .	SJM-210
session switches . . . . .	SJM-248
Switches package . . . . .	LM-315
What package . . . . .	SJM-254
windows images . . . . .	EST-157
xref images . . . . .	EST-162
delaying in wait service execution message (Debug.Task_Display) . . . . .	DEB-137
delete	
access . . . . .	LM-21
Ada units	
Compilation.Delete procedure . . . . .	LM-139
configuration object . . . . .	PM-49
objects . . . . .	PM-35
old versions	
Library.Expunge procedure . . . . .	LM-237
print request	
Queue.Cancel procedure . . . . .	SMU-98
view . . . . .	PM-48
<i>see also</i> Atomic_Destroy, Cancel, Destroy	
Delete constant	
Access_List.Delete . . . . .	LM-37
Access_List_Tools.Delete . . . . .	LM-65
Delete procedure	
Bounded_String.Delete . . . . .	ST-7
Common.Object.Delete . . . . .	<i>EST-105</i> , PM-35, PM-137
Ada images . . . . .	EST-16
Library package . . . . .	LM-206
Links package . . . . .	LM-278
Search_List package . . . . .	SJM-211
session switches . . . . .	SJM-250
Switches package . . . . .	LM-317
text images . . . . .	EST-142
What package . . . . .	SJM-254
windows images . . . . .	EST-158

Delete procedure ( <i>continued</i> )	
Compilation.Delete . . . . .	LM-139, PM-48
Destroy procedure . . . . .	LM-148
Direct_Io.Delete . . . . .	DIO-12
Editor.Line.Delete . . . . .	EI-31, EI-32
Editor.Region.Delete . . . . .	EI-5, EI-46
Editor.Window.Delete . . . . .	EI-64, EI-65
Editor.Word.Delete . . . . .	EI-69, EI-70
Io.Delete . . . . .	TIO-25
Library.Delete . . . . .	PM-48, PM-49
Links.Delete . . . . .	LM-284
Polymorphic_Sequential_Io.Delete . . . . .	DIO-44
Queue_Generic.Delete . . . . .	PT-98
Search_List.Delete . . . . .	SJM-215
Sequential_Io.Delete . . . . .	DIO-65
Set_Generic.Delete . . . . .	PT-114
Text_Io.Delete . . . . .	TIO-173
Unbounded_String.Delete . . . . .	ST-110
Window_Io.Delete . . . . .	DIO-113, DIO-114
Delete renamed procedure	
Library.Delete . . . . .	LM-230
Destroy renamed procedure . . . . .	LM-232
Delete_Backward procedure	
Editor.Char.Delete_Backward . . . . .	EI-12
Editor.Line.Delete_Backward . . . . .	EI-33
Editor.Word.Delete_Backward . . . . .	EI-70
Delete_Blank_Line procedure	
Ada.Delete_Blank_Line . . . . .	EST-26
Delete_Field procedure	
Work_Order.Delete_Field . . . . .	PM-379
Delete_Forward procedure	
Editor.Char.Delete_Forward . . . . .	EI-12
Editor.Line.Delete_Forward . . . . .	EI-33
Editor.Word.Delete_Forward . . . . .	EI-70
Delete_Group procedure	
Operator.Delete_Group . . . . .	SMU-66
Create_Group procedure . . . . .	SMU-61
Delete_Lines procedure	
Window_Io.Delete_Lines . . . . .	DIO-115
Delete_Next procedure	
Editor.Char.Delete_Next . . . . .	EI-13
Delete_Previous procedure	
Editor.Char.Delete_Previous . . . . .	EI-13
Delete_Spaces procedure	
Editor.Char.Delete_Spaces . . . . .	EI-12, EI-13
Delete_Temporary_Breaks enumeration	
Debug.Option type . . . . .	DEB-87
Delete_Top procedure	
Editor.Hold_Stack.Delete_Top . . . . .	EI-21, EI-22
Editor.Mark.Delete_Top . . . . .	EI-4, EI-41, EI-42
Editor.Screen.Delete_Top . . . . .	EI-7, EI-51, EI-52
Delete_Unreferenced_Leading_Generations procedure	
Cmvc_Maintenance.Delete_Unreferenced_Leading_Generations . . . . .	PM-343

Delete_User procedure	
Operator.Delete_User . . . . .	SMU-67
Delete_Group procedure . . . . .	SMU-66
deleted objects . . . . .	LM-198
referring to . . . . .	LM-13, PM-133, SJM-11, SMU-8
delimiters, value	
colon equals (:=) . . . . .	LM-18, SJM-15, SMU-8
equals (=) . . . . .	LM-18, SJM-15, SMU-8
equals/greater than (=>) . . . . .	LM-18, SJM-15, SMU-8
delta	
System.Fine_Delta constant . . . . .	PT-165
demote	
Ada.Source_Unit procedure . . . . .	EST-42
objects	
Compilation.Delete procedure . . . . .	LM-139
Compilation.Demote procedure . . . . .	LM-141
[Demote] key	
Common.Demote procedure . . . . .	EST-76
Demote procedure	
Common.Demote . . . . .	EST-76, PM-192
Ada images . . . . .	EST-13
command images . . . . .	EST-48
Library package . . . . .	LM-204
menu images . . . . .	EST-133
Redo procedure . . . . .	EST-49
text images . . . . .	EST-141
windows images . . . . .	EST-157
xref images . . . . .	EST-162
Compilation.Demote . . . . .	LM-141
Ada images . . . . .	EST-5
Ada.Source_Unit procedure . . . . .	EST-42
Editor.Window.Demote . . . . .	EI-63, EI-64, EI-65
Promote procedure . . . . .	EI-67
demotion	
effects of . . . . .	PM-90
permitting . . . . .	PM-42
dependency database . . . . .	SMU-11
Dependents procedure	
Compilation.Dependents . . . . .	LM-145
Links.Dependents . . . . .	LM-286
deposit, <i>see</i> Modify	
descriptor, job	
Scheduler.Get_Job_Descriptor function . . . . .	SMU-154
Scheduler.Job_Descriptor type . . . . .	SMU-160
Scheduler.Traverse_Job_Descriptors generic procedure . . . . .	SMU-185
design changes . . . . .	PM-89
designation . . . . .	EST-57
Ada images . . . . .	EST-3
command images . . . . .	EST-46
in Debugger window . . . . .	DEB-3
menu images . . . . .	EST-130
text images . . . . .	EST-137, EST-138
Window Directory . . . . .	EST-154

designation ( <i>continued</i> )	
xref images . . . . .	EST-160
<i>see also</i> selection	
Designation type	
Window_Io.Designation . . . . .	DIO-116
Designation_Off procedure	
Editor.Set.Designation_Off . . . . .	EI-4, EI-59, EI-60
destroy	
Compilation.Atomic_Destroy procedure . . . . .	LM-133
<i>see also</i> Delete_Group, Delete_User	
Destroy procedure	
Compilation.Destroy . . . . .	LM-148, PM-48
Atomic_Destroy procedure . . . . .	LM-133
Delete procedure . . . . .	LM-139
Library.Destroy . . . . .	PM-48
Queue.Destroy . . . . .	SMU-104
Destroy renamed procedure	
Library.Destroy . . . . .	LM-232
Delete renamed procedure . . . . .	LM-230
Destroy_Cdb procedure	
Cmvc_Maintenance.Destroy_Cdb . . . . .	PM-109, PM-344
Destroy_Subsystem procedure	
Cmvc.Destroy_Subsystem . . . . .	PM-236
Destroy_System procedure	
Cmvc.Destroy_System . . . . .	PM-237
Destroy_View procedure	
Cmvc.Destroy_View . . . . .	PM-33, PM-48, PM-50, PM-238
Cmvc.Build procedure . . . . .	PM-212
detach, <i>see</i> Disconnect	
Detach_On_Disconnect function	
System_Uilities.Detach_On_Disconnect . . . . .	SMU-206
Detached enumeration	
Scheduler.Job_Kind type . . . . .	SMU-165
detached job . . . . .	SMU-133
development	
applications using multiple hosts . . . . .	PM-101
applications using multiple subsystems . . . . .	PM-51
copying views among hosts . . . . .	PM-103, PM-109
making design changes . . . . .	PM-89
making implementation changes . . . . .	PM-86
managing CMVC information interactively . . . . .	PM-188
managing views . . . . .	PM-48
moving a primary subsystem to another host . . . . .	PM-108
path . . . . .	PM-8, PM-33, PM-111, PM-268
propagating changes across hosts . . . . .	PM-105
setting up multiple paths . . . . .	PM-47
setting up primary and secondary subsystems . . . . .	PM-103
setting up subsystems . . . . .	PM-96
testing an application . . . . .	PM-85
using CDFs with subsystems . . . . .	PM-111
with joined objects . . . . .	PM-38
<i>see also</i> subsystem	



device . . . . .	DIO-2, SMU-92, TIO-2
add	
Queue.Add procedure . . . . .	SMU-93
associate with class	
Queue.Register procedure . . . . .	SMU-123
class	
Queue.All_Classes constant . . . . .	SMU-96
class name	
Queue.Class_Name subtype . . . . .	SMU-100
disable	
Queue.Disable procedure . . . . .	SMU-107
dissociate from class	
Queue.Unregister procedure . . . . .	SMU-128
enable	
Queue.Enable procedure . . . . .	SMU-111
remove from print spooler	
Queue.Remove procedure . . . . .	SMU-125
spooler	
Queue.All_Spooler_Devices constant . . . . .	SMU-97
Device_Data_Error	
Io_Exceptions.Device_Error exception . . . . .	DIO-31, TIO-157
Device_Error exception	
Io_Exceptions.Device_Error . . . . .	DIO-31, TIO-157
Devices procedure	
Queue.Devices . . . . .	SJM-200, SMU-106
diagnosis	
Operator.Internal_System_Diagnosis procedure . . . . .	SMU-79
Diana_Edit procedure	
Ada.Diana_Edit . . . . .	EST-27
Dictionary subclass . . . . .	LM-17, SJM-15
difference	
Calendar.- function . . . . .	PT-8
Difference procedure	
File_Uilities.Difference . . . . .	LM-176
Merge procedure . . . . .	LM-190
Strip procedure . . . . .	LM-193
Differential enumeration	
Activity.Creation_Mode . . . . .	PM-146
Activity.Display procedure . . . . .	PM-148
digits	
Editor.Set.Argument_Digit procedure . . . . .	EI-60
System.Max_Digits constant . . . . .	PT-165
Direct_Io generic package . . . . .	DIO-7
<DIRECTORIES> special value . . . . .	LM-129, LM-134, LM-138, LM-162, LM-199
Directory	
client . . . . .	SMU-13, SMU-17, SMU-20, SMU-27, SMU-35
object manager . . . . .	SMU-11, SMU-58
directory . . . . .	LM-2, SJM-2
create	
Library.Create procedure . . . . .	LM-220
Library.Create_Directory renamed procedure . . . . .	LM-222
current	
Compilation.Current_Directory constant . . . . .	LM-138

directory ( <i>continued</i> )	
error, <i>see</i> Nonexistent_Directory_Error	
name . . . . .	LM-7, PM-127, SJM-5, SMU-1
same	
Compilation.Same_Directories constant . . . . .	LM-162
<i>see also</i> library	
Directory enumeration	
Library.Kind type . . . . .	LM-246
Directory procedure	
Editor.Window.Directory . . . . .	EI-6, EI-63, EI-66
Window Directory . . . . .	EST-153
Directory subclass . . . . .	LM-15, SJM-13
Disable procedure	
Job.Disable . . . . .	SJM-21
Text.Block procedure . . . . .	EST-146
What.Jobs procedure . . . . .	SJM-262
What.Users procedure . . . . .	SJM-272
Queue.Disable . . . . .	SMU-107
Scheduler.Disable . . . . .	SMU-131, SMU-142
Disable renamed procedure	
Debug.Disable . . . . .	DEB-3, DEB-16, DEB-52
Flag procedure . . . . .	DEB-63
Option type . . . . .	DEB-86
Disable_Deallocation pragma	
Unchecked_Deallocation generic procedure . . . . .	PT-241
Unchecked_Deallocation.Unchecked_Deallocation procedure . . . . .	PT-247
Disable_Terminal procedure	
Operator.Disable_Terminal . . . . .	SMU-68
disabled	
System_Uilities.Login_Disabled function . . . . .	SMU-238
Terminal.Set_Login_Disabled procedure . . . . .	SMU-314
Disabled enumeration	
Scheduler.Job_State type . . . . .	SMU-167
Disabled state . . . . .	SMU-134
disconnect	
System_Uilities.Detach_On_Disconnect function . . . . .	SMU-206
System_Uilities.Logoff_On_Disconnect function . . . . .	SMU-239
Terminal.Set_Disconnect_On_Disconnect procedure . . . . .	SMU-308
Terminal.Set_Disconnect_On_Failed_Login procedure . . . . .	SMU-309
Terminal.Set_Disconnect_On_Logoff procedure . . . . .	SMU-310
Terminal.Set_Logoff_On_Disconnect procedure . . . . .	SMU-315
Disconnect procedure	
Job.Disconnect . . . . .	SJM-23
Interrupt procedure . . . . .	SJM-28
Window_Io.Raw.Disconnect . . . . .	DIO-176
Disconnect_On_Disconnect function	
System_Uilities.Disconnect_On_Disconnect . . . . .	SMU-207
Disconnect_On_Failed_Login function	
System_Uilities.Disconnect_On_Failed_Login . . . . .	SMU-208
Disconnect_On_Logoff function	
System_Uilities.Disconnect_On_Logoff . . . . .	SMU-209

disk

bad blocks

System_Uilities.All_Bad_Blocks constant . . . . .	SMU-198
System_Uilities.Bad_Block_Kinds type . . . . .	SMU-199
System_Uilities.Bad_Block_List function . . . . .	SMU-200
System_Uilities.Block_List type . . . . .	SMU-201
System_Uilities.Manufacturers_Bad_Blocks constant . . . . .	SMU-240

collection . . . . .

priority . . . . .	SMU-15
start . . . . .	SMU-14

drive

Daemon.Volume subtype . . . . .	SMU-47
---------------------------------	--------

retargeted blocks

System_Uilities.Retargeted_Blocks constant . . . . .	SMU-256
--	---------

scheduling . . . . .

parameters . . . . .	SMU-172, SMU-176
----------------------	------------------

space

Library.Space procedure . . . . .	LM-263
Operator.Disk_Space procedure . . . . .	SJM-67

volume

Library.Nil constant . . . . .	LM-254
Library.Volume subtype . . . . .	LM-274

wait load . . . . .

Scheduler.Get_Disk_Wait_Load procedure . . . . .	SMU-152
--	---------

Disk client . . . . .	SMU-12, SMU-13, SMU-20, SMU-27, SMU-31, SMU-38
-----------------------	--

Disk\_Space procedure

Operator.Disk_Space . . . . .	SJM-67, SMU-69
-------------------------------	----------------

Disk\_Waits function

Scheduler.Disk_Waits . . . . .	SMU-143
Job_Descriptor type . . . . .	SMU-161
summary, <i>see</i> Summarize	

display

default

Access_List.Default_Display procedure . . . . .	LM-35
---	-------

defining occurrences

Common.Definition procedure . . . . .	EST-71
---------------------------------------	--------

errors, *see* Filter\_Errors

history

Debug.History_Display procedure . . . . .	DEB-68
---	--------

image, *see* Designation, Font

Library.Ada_List renamed procedure . . . . .	LM-209
--	--------

Library.File_List renamed procedure . . . . .	LM-242
---	--------

Library.List procedure . . . . .	LM-248
----------------------------------	--------

memory

Debug.Memory_Display procedure . . . . .	DEB-79
--	--------

other part of Ada unit

Ada.Other_Part procedure . . . . .	EST-36
------------------------------------	--------

pathname

What.Object procedure . . . . .	SJM-268
---------------------------------	---------

summary, *see* Summarize

task

Debug.Task_Display procedure . . . . .	DEB-136
--	---------

*see also* Default\_Font, Show

Display procedure

Access_List.Display . . . . .	LM-38
-------------------------------	-------

Activity.Display . . . . .	PM-148
----------------------------	--------

Debug.Display . . . . .	DEB-4, DEB-7, DEB-10, DEB-14, DEB-15, DEB-18, DEB-53
-------------------------	--

Context procedure . . . . .	DEB-44, DEB-45
-----------------------------	----------------

Display procedure (*continued*)

Debug.Display (*continued*)

Debug_Declaration_Display session switch . . . . .	SJM-232
Debug_Display_Count session switch . . . . .	SJM-232
Numeric type . . . . .	DEB-84
Option type . . . . .	DEB-86
Library.Display . . . . .	LM-234
Links.Display . . . . .	LM-288
Queue.Display . . . . .	SJM-201, SMU-109
Cancel procedure . . . . .	SJM-198
Scheduler.Display . . . . .	SMU-131, SMU-137, SMU-144
Set procedure . . . . .	SMU-170
Search_List.Display . . . . .	SJM-216
Switches.Display . . . . .	LM-326
Table_Formatter.Display . . . . .	ST-91, ST-94
Work_Order.Display . . . . .	PM-380

display/write, *see* Put\_Condition, Put\_Job\_Messages, Put\_Line, Put\_System\_Messages

Display\_Cdb procedure

Cmvc_Maintenance.Display_Cdb . . . . .	PM-105, PM-346
--	----------------

Display\_Code\_View procedure

Cmvc_Maintenance.Display_Code_View . . . . .	PM-348
--	--------

Display\_Count enumeration

Debug.Numeric type . . . . .	DEB-84
------------------------------	--------

Display\_Creation enumeration

Debug.Option type . . . . .	DEB-87
-----------------------------	--------

Display\_Group procedure

Operator.Display_Group . . . . .	SJM-68, SMU-70
Add_To_Group procedure . . . . .	SMU-56
Remove_From_Group procedure . . . . .	SMU-82

Display\_Level enumeration

Debug.Numeric type . . . . .	DEB-84
------------------------------	--------

Display\_Libraries procedure

Search_List.Display_Libraries . . . . .	SJM-217
---	---------

Display\_List procedure

Work_Order.Display_List . . . . .	PM-381
-----------------------------------	--------

Display\_Message function

Simple_Status.Display_Message . . . . .	PT-134
---	--------

Display\_Message procedure

Simple_Status.Display_Message	
Program.Started_Successfully function . . . . .	SJM-195

Display\_Tape procedure

Tape.Display_Tape . . . . .	SMU-284
-----------------------------	---------

Display\_Venture procedure

Work_Order.Display_Venture . . . . .	PM-382
--------------------------------------	--------

Dissociate procedure

Switches.Dissociate . . . . .	LM-328
Associate procedure . . . . .	LM-318

Documents subclass . . . . .

LM-17, SJM-15

Does procedure

What.Does . . . . .	SJM-257
Editor.Key package . . . . .	EI-27
Editor.Key.Define procedure . . . . .	EI-28

dollar sign (\$)	
file utilities wildcard . . . . .	LM-172, LM-181, LM-184, LM-187
metacharacter . . . . .	EI-56
special character . . . . .	DEB-18, LM-10, LM-11, PM-131, SJM-8, SJM-9, SJM-209, SMU-6
dollar sign, double (\$\$)	
special character . . . . .	DEB-18, DEB-19, LM-10, LM-11, PM-132, SJM-8, SJM-9, SMU-6
Dollar_Msg enumeration	
Profile.Msg_Kind type . . . . .	SJM-117
domain type . . . . .	PT-9, PT-67, ST-25
Domain_Type generic formal type	
Concurrent_Map_Generic.Domain_Type . . . . .	PT-13
Map_Generic.Domain_Type . . . . .	PT-71
Done function	
Concurrent_Map_Generic.Done . . . . .	PT-14
Init procedure . . . . .	PT-19
Iterator type . . . . .	PT-24
Next procedure . . . . .	PT-28
List_Generic.Done . . . . .	PT-50
Init procedure . . . . .	PT-54
Iterator type . . . . .	PT-57
Next procedure . . . . .	PT-61
Map_Generic.Done . . . . .	PT-72
Init procedure . . . . .	PT-77
Iterator type . . . . .	PT-82
Next procedure . . . . .	PT-86
Queue_Generic.Done . . . . .	PT-99
Init procedure . . . . .	PT-102
Iterator type . . . . .	PT-106
Next procedure . . . . .	PT-108
Set_Generic.Done . . . . .	PT-115
Init procedure . . . . .	PT-117
Iterator type . . . . .	PT-121
Next procedure . . . . .	PT-123
Stack_Generic.Done . . . . .	PT-145
Init procedure . . . . .	PT-149
Iterator type . . . . .	PT-151
Next procedure . . . . .	PT-153
String_Map_Generic.Done . . . . .	ST-29
Init procedure . . . . .	ST-33
Iterator type . . . . .	ST-38
Next procedure . . . . .	ST-42
String_Table.Done . . . . .	ST-52
Init procedure . . . . .	ST-56
Iterator type . . . . .	ST-59
Next procedure . . . . .	ST-62
System_Uilities.Done . . . . .	SMU-210, SMU-211, SMU-212
Init procedure . . . . .	SMU-225, SMU-226, SMU-227
Job_Iterator type . . . . .	SMU-232
Next procedure . . . . .	SMU-241, SMU-242, SMU-243
Session_Iterator type . . . . .	SMU-259
Terminal_Iterator type . . . . .	SMU-271
Value function . . . . .	SMU-276
double dollar sign (\$\$)	
special character . . . . .	DEB-18, DEB-19, LM-10, LM-11, PM-132, SJM-8, SJM-9, SMU-6
double dot symbol (..)	LM-18, SJM-16, SMU-9
double question mark (??)	
library wildcard . . . . .	LM-8, LM-9, PM-129, SMU-3, SMU-4

Down procedure	
Editor.Cursor.Down . . . . .	EI-17, EI-18
Editor.Image.Down . . . . .	EI-25
Window_Scroll_Overlap session switch . . . . .	SJM-247
Editor.Screen.Down . . . . .	EI-53
drives, <i>see</i> disk, tape	
dump memory	
Debug.Memory_Display procedure . . . . .	DEB-79
Dump procedure	
Editor.Screen.Dump . . . . .	EI-51, EI-53
Screen_Dump_File session switch . . . . .	SJM-244
File_Utillities.Dump . . . . .	LM-179
duplicate, <i>see</i> Copy	
duration	
Calendar.Day_Duration subtype . . . . .	PT-6
Duration type	
Standard.Duration . . . . .	PT-161
Calendar.Time_Error exception . . . . .	PT-7
Image function . . . . .	PT-190
Interval type . . . . .	PT-193
Time_Utillities.Duration	
Daemon.Schedule procedure . . . . .	SMU-33
Duration_Until function	
Time_Utillities.Duration_Until . . . . .	PT-185
Duration_Until_Next function	
Time_Utillities.Duration_Until_Next . . . . .	PT-186
Daemon.Schedule procedure . . . . .	SMU-33
Dy_Mon_Yr enumeration	
Profile.Log_Prefix type . . . . .	SJM-114

E

Echo procedure	
Io.Echo . . . . .	TIO-26, TIO-27, TIO-28, TIO-29, TIO-31
Echo_Commands enumeration	
Debug.Option type . . . . .	DEB-87
Echo_Line procedure	
Io.Echo_Line . . . . .	TIO-32
edit	
activities . . . . .	PM-135, PM-150
controlled objects . . . . .	PM-26
session switches	
Switches.Edit_Session_Attributes procedure . . . . .	LM-330
State.Exports file . . . . .	PM-58
ventures . . . . .	PM-385, PM-417
work orders . . . . .	PM-403
work-order list . . . . .	PM-384
<i>see also</i> Demote, editing images, editing operations	
[Edit] key . . . . .	PM-136
Common.Edit procedure . . . . .	EST-78

Edit procedure	
Activity.Edit . . . . .	PM-135, <i>PM-150</i>
Cmvc.Edit . . . . .	PM-40, <i>PM-241</i>
Common.Edit . . . . .	EST-9, <i>EST-78</i> , PM-26, PM-27, PM-136, PM-142, PM-174
Activity.Change procedure . . . . .	PM-142
Activity.Visit procedure . . . . .	PM-174
Ada images . . . . .	EST-13
command images . . . . .	EST-48
Library package . . . . .	LM-204
Links package . . . . .	LM-276
menu images . . . . .	EST-134
Redo procedure . . . . .	EST-49
Search_List package . . . . .	SJM-210
session switches . . . . .	SJM-249
Switches package . . . . .	LM-315
text images . . . . .	EST-141
windows images . . . . .	EST-157
Links.Edit . . . . .	<i>LM-290</i>
Visit procedure . . . . .	LM-304
Switches.Edit . . . . .	LM-308, <i>LM-329</i>
Visit procedure . . . . .	LM-338
Work_Order.Edit . . . . .	<i>PM-383</i>
Edit_List procedure	
Work_Order.Edit_List . . . . .	<i>PM-384</i>
Edit_Session_Attributes procedure	
Switches.Edit_Session_Attributes . . . . .	LM-308, <i>LM-330</i>
Create procedure . . . . .	<i>LM-323</i>
session switches . . . . .	SJM-227, SJM-248
Edit_Venture procedure	
Work_Order.Edit_Venture . . . . .	<i>PM-385</i>
editing images	
session switches . . . . .	SJM-228
editing operations . . . . . EI-4	
bell	
Editor.Alert procedure . . . . .	EI-10
case changing	
Editor.Char package . . . . .	EI-11
characters	
Editor.Char package . . . . .	EI-11
cursor movement	
Editor.Cursor package . . . . .	EI-17
do nothing	
Editor.Noop procedure . . . . .	EI-10
edit text . . . . . EI-5	
editor parameters	
Editor.Set package . . . . .	EI-59
find images	
Editor.Image package . . . . .	EI-25
hold stack . . . . . EI-5	
Editor.Hold_Stack package . . . . .	EI-21
insert lines	
Editor.Line package . . . . .	EI-31
join lines	
Editor.Line package . . . . .	EI-31
key bindings	
Editor.Key package . . . . .	EI-27
keyboard macros	
Editor.Macro package . . . . .	EI-37

editing operations ( <i>continued</i> )	
lines and tabs . . . . .	EI-5
Editor.Char package . . . . .	EI-11
Editor.Line package . . . . .	EI-31
log off	
Editor.Quit procedure . . . . .	EI-10
marks	
Editor.Mark package . . . . .	EI-41
retrieve text . . . . .	EI-5
Editor.Hold_Stack package . . . . .	EI-21
screen management	
Editor.Screen package . . . . .	EI-51
scroll images	
Editor.Image package . . . . .	EI-25
search and replace	
Editor.Search package . . . . .	EI-55
select text . . . . .	EI-6
Editor.Region package . . . . .	EI-45
tabs	
Editor.Char package . . . . .	EI-11
Editor.Set package . . . . .	EI-59
window management	
Editor.Window package . . . . .	EI-63
words	
Editor.Word package . . . . .	EI-69
Editor	
Debugger interactions . . . . .	DEB-2
Editor package . . . . .	EI-4, EI-9
Work_Order.Editor . . . . .	PM-403
editor windows	
Window_Io package . . . . .	DIO-79
EEDB (Environment Elaborator Database) interpreter	
Operator.Internal_System_Diagnosis procedure . . . . .	SMU-79
elaboration . . . . .	DEB-10
elapsed duration	
Time_Utilities.Duration_Until function . . . . .	PT-185
Time_Utilities.Duration_Until_Next function . . . . .	PT-186
Elapsed function	
System_Utilities.Elapsed . . . . .	SMU-213
element . . . . .	DIO-7, PM-44
Element generic formal type	
List_Generic.Element . . . . .	PT-51
Queue_Generic.Element . . . . .	PT-100
Set_Generic.Element . . . . .	PT-116
Stack_Generic.Element . . . . .	PT-146
Table_Sort_Generic.Element . . . . .	PT-171
Element_Array generic formal type	
Table_Sort_Generic.Element_Array . . . . .	PT-172
Element_Count enumeration	
Debug.Numeric type . . . . .	DEB-84
Element_Type generic formal type	
Direct_Io.Element_Type . . . . .	DIO-13
Polymorphic_Sequential_Io.Operations.Element_Type . . . . .	DIO-56
Sequential_Io.Element_Type . . . . .	DIO-66



Elide procedure	
Common.Elide . . . . .	EST-81, PM-193
Library package . . . . .	LM-204
Links package . . . . .	LM-277
menu images . . . . .	EST-134
session switches . . . . .	SJM-249
Switches package . . . . .	LM-315
What package . . . . .	SJM-254
xref images . . . . .	EST-162
Common.Object.Elide . . . . .	PM-137
elision . . . . .	
levels . . . . .	LM-199
menu images . . . . .	LM-200
xref images . . . . .	EST-131
xref images . . . . .	EST-160
empty	
Concurrent_Map_Generic.Is_Empty function . . . . .	PT-22
Concurrent_Map_Generic.Make_Empty procedure . . . . .	PT-25
List_Generic.Is_Empty function . . . . .	PT-56
Map_Generic.Is_Empty function . . . . .	PT-80
Map_Generic.Make_Empty procedure . . . . .	PT-83
Queue_Generic.Is_Empty function . . . . .	PT-105
Queue_Generic.Make_Empty procedure . . . . .	PT-107
Set_Generic.Is_Empty function . . . . .	PT-119
Set_Generic.Make_Empty procedure . . . . .	PT-122
Stack_Generic.Make_Empty procedure . . . . .	PT-152
String_Map_Generic.Is_Empty function . . . . .	ST-36
String_Map_Generic.Make_Empty procedure . . . . .	ST-39
<i>see also</i> Is_Nil, Nil, Underflow	
Empty function	
Stack_Generic.Empty . . . . .	PT-147
Empty_Stack constant	
Stack_Generic.Empty_Stack . . . . .	PT-148
Make_Empty procedure . . . . .	PT-152
Enable procedure	
Debug.Enable . . . . .	DEB-16, DEB-56
Flag procedure . . . . .	DEB-63
Option type . . . . .	DEB-86
Job.Enable . . . . .	SJM-25
Disable procedure . . . . .	SJM-21
What.Jobs procedure . . . . .	SJM-262
What.Users procedure . . . . .	SJM-272
Queue.Enable . . . . .	SMU-111
Remove procedure . . . . .	SMU-125
Scheduler.Enable . . . . .	SMU-131, SMU-147
Disable procedure . . . . .	SMU-142
Enable_Deallocation	
library switch . . . . .	LM-311
Unchecked_Deallocation generic procedure . . . . .	PT-241
pragma	
Unchecked_Deallocation generic procedure . . . . .	PT-241, PT-242
Unchecked_Deallocation.Unchecked_Deallocation procedure . . . . .	PT-247
Enable_Privileges procedure	
Operator.Enable_Privileges . . . . .	LM-20, SMU-72
Enable_Terminal procedure	
Operator.Enable_Terminal . . . . .	SMU-73
System_Uilities.Enabled function . . . . .	SMU-214

Enable_Terminal procedure ( <i>continued</i> )		
Operator.Enable_Terminal ( <i>continued</i> )		
System_Uilities.Login_Disabled function . . . . .	SMU-238	
Terminal.Set_Login_Disabled procedure . . . . .	SMU-314	
Enabled function		
Scheduler.Enabled . . . . .	SMU-148	
System_Uilities.Enabled . . . . .	SMU-214	
enclosing		
library . . . . .	DEB-18, LM-11, PM-131, SJM-9, SJM-209, SMU-6	
object . . . . .	DEB-18, LM-11, PM-131, SJM-8, SJM-9, SMU-6	
world . . . . .	DEB-19, LM-11, PM-132, SJM-9, SMU-6	
[Enclosing In Place] key		
Common.Enclosing procedure . . . . .	EST-83	
[Enclosing] key		
Common.Enclosing procedure . . . . .	EST-83	
[Enclosing Library] key		
Common.Enclosing procedure . . . . .	EST-83	
Enclosing procedure		
Common.Enclosing . . . . .	EST-83	
Ada images . . . . .	EST-14	
command images . . . . .	EST-48	
Debugger . . . . .	DEB-5	
Library package . . . . .	LM-205	
Links package . . . . .	LM-277	
session switches . . . . .	SJM-249	
Switches package . . . . .	LM-316	
text images . . . . .	EST-141	
xref images . . . . .	EST-163	
Enclosing_Subsystem procedure		
Activity.Enclosing_Subsystem . . . . .	PM-151	
Enclosing_View procedure		
Activity.Enclosing_View . . . . .	PM-152	
Enclosing_World procedure		
Library.Enclosing_World . . . . .	LM-235	
end, see Quit		
[End of Input] key		
Text.End_Of_Input procedure . . . . .	EST-149	
end-of-file terminator . . . . .		DIO-6, TIO-5
End_Error exception		
Direct_Io generic package		
Read procedure . . . . .	DIO-24	
Io package		
Get procedure . . . . .	TIO-41, TIO-42, TIO-44, TIO-46, TIO-48, TIO-49	
Get_Line procedure . . . . .	TIO-50, TIO-52	
Set_Col procedure . . . . .	TIO-86	
Set_Line procedure . . . . .	TIO-92	
Skip_Line procedure . . . . .	TIO-97	
Skip_Page procedure . . . . .	TIO-98	
Io.Enumeration_Io generic package		
Get procedure . . . . .	TIO-113	
Io.Fixed_Io generic package		
Get procedure . . . . .	TIO-124	
Io.Float_Io generic package		
Get procedure . . . . .	TIO-136	

End_Error exception ( <i>continued</i> )	
Io.Integer_Io generic package	
Get procedure . . . . .	TIO-147
Io_Exceptions package	
Text.End_Of_Input procedure . . . . .	EST-149
Io_Exceptions.End_Error . . . . .	DIO-32, TIO-158
Polymorphic_Sequential_Io.Operations package	
Read procedure . . . . .	DIO-57
Sequential_Io package	
Read procedure . . . . .	DIO-76
Text_Io package	
Get procedure . . . . .	TIO-181, TIO-182
Get_Line procedure . . . . .	TIO-184
Set_Col procedure . . . . .	TIO-204
Set_Line procedure . . . . .	TIO-207
Skip_Line procedure . . . . .	TIO-211
Skip_Page procedure . . . . .	TIO-212
Text_Io.Enumeration_Io generic package	
Get procedure . . . . .	TIO-221
Text_Io.Fixed_Io generic package	
Get procedure . . . . .	TIO-232
Text_Io.Float_Io generic package	
Get procedure . . . . .	TIO-244
Text_Io.Integer_Io generic package	
Get procedure . . . . .	TIO-255
End_Of procedure	
Editor.Image.End_Of . . . . .	EI-25, EI-26
Editor.Line.End_Of . . . . .	EI-31, EI-33
Editor.Region.End_Of . . . . .	EI-46
Editor.Window.End_Of . . . . .	EI-64, EI-66
Editor.Word.End_Of . . . . .	EI-71
End_Of_File function	
Direct_Io.End_Of_File . . . . .	DIO-14
Io.End_Of_File . . . . .	TIO-33
Polymorphic_Sequential_Io.End_Of_File . . . . .	DIO-45
Sequential_Io.End_Of_File . . . . .	DIO-67
Text_Io.End_Of_File . . . . .	TIO-174
Window_Io.End_Of_File . . . . .	DIO-118
End_Of_Input procedure	
Text.End_Of_Input . . . . .	EST-149
End_Of_Line function	
Io.End_Of_Line . . . . .	TIO-34
Text_Io.End_Of_Line . . . . .	TIO-175
Window_Io.End_Of_Line . . . . .	DIO-119
End_Of_Page function	
Io.End_Of_Page . . . . .	TIO-35
Text_Io.End_Of_Page . . . . .	TIO-176
End_Rendezvous enumeration	
Debug.Stop_Event type . . . . .	DEB-130
enter	
Common.Commit procedure . . . . .	EST-65
enter cross-library link	
Links.Add procedure . . . . .	LM-279
[Enter] key	
Common.Commit procedure . . . . .	EST-65

entries, column	
Table_Formatter.Adjust type . . . . .	ST-93
Enum generic formal type	
Io.Enumeration_Io.Enum . . . . .	TIO-112
Get procedure . . . . .	TIO-113, TIO-114
Text_Io.Enumeration_Io.Enum . . . . .	TIO-220
Get procedure . . . . .	TIO-221, TIO-222
enumeration literals	
Io.Lower_Case constant . . . . .	TIO-57
Io.Type_Set subtype . . . . .	TIO-102
Io.Upper_Case constant . . . . .	TIO-104
Text_Io.Type_Set type . . . . .	TIO-215
enumeration value	
read from file	
Io.Enumeration_Io.Get procedure . . . . .	TIO-113
Text_Io.Enumeration_Io.Get procedure . . . . .	TIO-221
read from string	
Io.Enumeration_Io.Get procedure . . . . .	TIO-114
Text_Io.Enumeration_Io.Get procedure . . . . .	TIO-222
write to file	
Io.Enumeration_Io.Put procedure . . . . .	TIO-115
Text_Io.Enumeration_Io.Put procedure . . . . .	TIO-223
write to string	
Io.Enumeration_Io.Put procedure . . . . .	TIO-117
Text_Io.Enumeration_Io.Put procedure . . . . .	TIO-225
Enumeration_Io generic package	
Io.Enumeration_Io . . . . .	TIO-109
Text_Io.Enumeration_Io . . . . .	TIO-217
enumerations	
Activity.Creation_Mode	
Differential enumeration . . . . .	PM-146, PM-148
Exact_Copy enumeration . . . . .	PM-146, PM-148
Value_Copy enumeration . . . . .	PM-146, PM-148
Check.Status	
Compatible enumeration . . . . .	PM-179
Error enumeration . . . . .	PM-179
Incompatible enumeration . . . . .	PM-179
Cmvc.System_Object_Enum	
Combined_Subsystem enumeration . . . . .	PM-324
Spec_Load_Subsystem enumeration . . . . .	PM-324
System enumeration . . . . .	PM-324
Compilation.Promote_Scope	
All_Parts enumeration . . . . .	LM-160
Load_Views enumeration . . . . .	LM-160
Single_Unit enumeration . . . . .	LM-160
Subunits_Too enumeration . . . . .	LM-161
Unit_Only enumeration . . . . .	LM-161
Compilation.Unit_State	
Archived enumeration . . . . .	LM-166
Coded enumeration . . . . .	LM-166
Installed enumeration . . . . .	LM-166
Source enumeration . . . . .	LM-167
Daemon.Condition_Class	
Fatal enumeration . . . . .	SMU-16
Normal enumeration . . . . .	SMU-16
Problem enumeration . . . . .	SMU-16
Warning enumeration . . . . .	SMU-16

enumerations (continued)

Daemon.Log_Threshold	
Commit_Disk enumeration . . . . .	SMU-26
Console_Print enumeration . . . . .	SMU-26
Log_To_Disk enumeration . . . . .	SMU-26
Debug.Context_Type	
Control enumeration . . . . .	DEB-48
Evaluation enumeration . . . . .	DEB-48
Debug.Information_Type	
Exceptions enumeration . . . . .	DEB-75
Rendezvous enumeration . . . . .	DEB-75
Space enumeration . . . . .	DEB-75
Debug.Numeric	
Display_Count enumeration . . . . .	DEB-84
Display_Level enumeration . . . . .	DEB-84
Element_Count enumeration . . . . .	DEB-84
First_Element enumeration . . . . .	DEB-84
History_Count enumeration . . . . .	DEB-85
History_Entries enumeration . . . . .	DEB-85
History_Start enumeration . . . . .	DEB-85
Memory_Count enumeration . . . . .	DEB-85
Pointer_Level . . . . .	DEB-85
Stack_Count enumeration . . . . .	DEB-85
Stack_Start enumeration . . . . .	DEB-85
Debug.Option	
Addresses enumeration . . . . .	DEB-86
Break_At_Creation enumeration . . . . .	DEB-86
Declaration_Display enumeration . . . . .	DEB-86
Delete_Temporary_Breaks enumeration . . . . .	DEB-87
Display_Creation enumeration . . . . .	DEB-87
Echo_Commands enumeration . . . . .	DEB-87
Freeze_Tasks enumeration . . . . .	DEB-87
Include_Packages enumeration . . . . .	DEB-87
Interpret_Control_Words enumeration . . . . .	DEB-87
Kill_Old_Jobs enumeration . . . . .	DEB-87
Machine_Level enumeration . . . . .	DEB-87
No_History_Timestamps enumeration . . . . .	DEB-87
Optimize_Generic_History enumeration . . . . .	DEB-87
Permanent_Breakpoints enumeration . . . . .	DEB-88
Put_Locals enumeration . . . . .	DEB-88
Qualify_Stack_Names enumeration . . . . .	DEB-88
Require_Debug_Off enumeration . . . . .	DEB-88
Save_Exceptions enumeration . . . . .	DEB-88
Show_Location enumeration . . . . .	DEB-88
Timestamps enumeration . . . . .	DEB-88
Debug.State_Type	
All_State enumeration . . . . .	DEB-126
Breakpoints enumeration . . . . .	DEB-126
Contexts enumeration . . . . .	DEB-126
Exceptions enumeration . . . . .	DEB-126
Flags enumeration . . . . .	DEB-126
Histories enumeration . . . . .	DEB-126
Libraries enumeration . . . . .	DEB-126
Special_Types enumeration . . . . .	DEB-126
Steps enumeration . . . . .	DEB-127
Stops_And_Holds enumeration . . . . .	DEB-127
Traces enumeration . . . . .	DEB-127
Debug.Stop_Event	
About_To_Return enumeration . . . . .	DEB-130
Begin_Rendezvous enumeration . . . . .	DEB-130
End_Rendezvous enumeration . . . . .	DEB-130

enumerations (continued)

Debug.Stop\_Event (continued)

Local_Statement enumeration . . . . .	DEB-130
Machine_Instruction enumeration . . . . .	DEB-130
Procedure_Entry enumeration . . . . .	DEB-131
Returned enumeration . . . . .	DEB-131
Statement enumeration . . . . .	DEB-131

Debug.Task\_Category

All_Tasks enumeration . . . . .	DEB-135
Blocked enumeration . . . . .	DEB-135
Held enumeration . . . . .	DEB-135
Not_Running enumeration . . . . .	DEB-135
Running enumeration . . . . .	DEB-135
Stopped enumeration . . . . .	DEB-135

Debug.Trace\_Event

All_Events enumeration . . . . .	DEB-146
Call enumeration . . . . .	DEB-146
Exception_Raised enumeration . . . . .	DEB-146
Propagate_Exception enumeration . . . . .	DEB-146
Rendezvous enumeration . . . . .	DEB-146
Statement enumeration . . . . .	DEB-146

Library.Field

Class enumeration . . . . .	LM-239
Create_Time enumeration . . . . .	LM-239
Creator enumeration . . . . .	LM-239
Declaration enumeration . . . . .	LM-239
Frozen enumeration . . . . .	LM-239
Object enumeration . . . . .	LM-239
Read_Time enumeration . . . . .	LM-239
Reader enumeration . . . . .	LM-240
Retain enumeration . . . . .	LM-240
Size enumeration . . . . .	LM-240
Status enumeration . . . . .	LM-240
Subclass enumeration . . . . .	LM-240
Update_Time enumeration . . . . .	LM-240
Updater enumeration . . . . .	LM-240
Version enumeration . . . . .	LM-240

Library.Kind

Directory enumeration . . . . .	LM-246
Subpackage enumeration . . . . .	LM-246
World enumeration . . . . .	LM-247

Profile.Error\_Reaction

Persevere enumeration . . . . .	SJM-96
Propagate enumeration . . . . .	SJM-96
Quit enumeration . . . . .	SJM-96
Raise_Error enumeration . . . . .	SJM-97

Profile.Log\_Output\_File

Use_Error enumeration . . . . .	SJM-113
Use_Output enumeration . . . . .	SJM-113
Use_Standard_Error enumeration . . . . .	SJM-113
Use_Standard_Output enumeration . . . . .	SJM-113

Profile.Log\_Prefix

Date enumeration . . . . .	SJM-114
Dy_Mon_Yr enumeration . . . . .	SJM-114
Hr_Mn enumeration . . . . .	SJM-114
Hr_Mn_Sc enumeration . . . . .	SJM-114
Mn_Dy_Yr enumeration . . . . .	SJM-114
Nil enumeration . . . . .	SJM-114
Symbols enumeration . . . . .	SJM-114
Time enumeration . . . . .	SJM-115
Yr_Mn_Dy enumeration . . . . .	SJM-115

enumerations (continued)

Profile.Msg_Kind	
At_Msg enumeration . . . . .	SJM-117
Auxiliary_Msg enumeration . . . . .	SJM-117
Debug_Msg enumeration . . . . .	SJM-117
Dollar_Msg enumeration . . . . .	SJM-117
Error_Msg enumeration . . . . .	SJM-118
Exception_Msg enumeration . . . . .	SJM-118
Negative_Msg enumeration . . . . .	SJM-118
Note_Msg enumeration . . . . .	SJM-118
Position_Msg enumeration . . . . .	SJM-118
Positive_Msg enumeration . . . . .	SJM-118
Sharp_Msg enumeration . . . . .	SJM-118
Warning_Msg enumeration . . . . .	SJM-119
Scheduler.Job_Kind	
Attached enumeration . . . . .	SMU-165
Ce enumeration . . . . .	SMU-165
Detached enumeration . . . . .	SMU-165
Oe enumeration . . . . .	SMU-165
Server enumeration . . . . .	SMU-166
Terminated enumeration . . . . .	SMU-166
Scheduler.Job_State	
Disabled enumeration . . . . .	SMU-167
Idle enumeration . . . . .	SMU-167
Queued enumeration . . . . .	SMU-167
Run enumeration . . . . .	SMU-167
Wait enumeration . . . . .	SMU-167
Simple_Status.Condition_Class	
Fatal enumeration . . . . .	PT-130
Normal enumeration . . . . .	PT-130
Problem enumeration . . . . .	PT-130
Warning enumeration . . . . .	PT-130
System_Backup.Kind	
Full enumeration . . . . .	SMU-196
Primary enumeration . . . . .	SMU-196
Secondary enumeration . . . . .	SMU-196
System_Uilities.Parity_Kind	
Even enumeration . . . . .	SMU-249
None enumeration . . . . .	SMU-249
Odd enumeration . . . . .	SMU-249
Table_Formatter.Adjust	
Centered enumeration . . . . .	ST-93
Left enumeration . . . . .	ST-93
Right enumeration . . . . .	ST-93
Text.Image_Kind	
File enumeration . . . . .	EST-150
Input_Output enumeration . . . . .	EST-150
Time_Uilities.Date_Format	
Ada enumeration . . . . .	PT-180
Day_Month_Year enumeration . . . . .	PT-180
Expanded enumeration . . . . .	PT-180
Month_Day_Year enumeration . . . . .	PT-180
Year_Month_Day enumeration . . . . .	PT-180
Time_Uilities.Image_Contents	
Both enumeration . . . . .	PT-192
Date_Only enumeration . . . . .	PT-192
Time_Only enumeration . . . . .	PT-192
Time_Uilities.Time_Format	
Ada enumeration . . . . .	PT-204
Expanded enumeration . . . . .	PT-204

enumerations (*continued*)

Time\_Utilities.Time\_Format (*continued*)

Military enumeration . . . . . PT-204  
 Short enumeration . . . . . PT-204

Window\_Io.Designation

Prompt enumeration . . . . . DIO-116  
 Protected enumeration . . . . . DIO-116  
 Text enumeration . . . . . DIO-116

Window\_Io.File\_Mode

In\_File enumeration . . . . . DIO-120  
 Out\_File enumeration . . . . . DIO-120

Work\_Order.Venture\_Policy\_Switch

Allow\_Edit\_Of\_Work\_Orders enumeration . . . . . PM-370, PM-400  
 Journal\_Comment\_Lines enumeration . . . . . PM-400  
 Require\_Comment\_Lines enumeration . . . . . PM-400  
 Require\_Comments\_At\_Check\_In enumeration . . . . . PM-400  
 Require\_Current\_Work\_Order enumeration . . . . . PM-401

Environment Elaborator Database (EEDB) . . . . . SMU-79

EOF, *see* End\_Error, End\_Of\_File

EOL, *see* End\_Of\_Line

equal

Io.= function . . . . . TIO-9

Equal function

File\_Utilities.Equal . . . . . LM-169, LM-181  
 Simple\_Status.Equal . . . . . PT-135  
 String\_Table.Equal . . . . . ST-53  
 String\_Utilities.Equal . . . . . ST-72

equals (=) value delimiter . . . . . LM-18, SJM-15, SMU-8

equals/greater than (=>) value delimiter . . . . . LM-18, SJM-15, SMU-8

error

Access\_List\_Tools.Access\_Tools\_Error exception . . . . . LM-55

Calendar.Time\_Error exception . . . . . PT-7

Io.Current\_Error function . . . . . TIO-22

Io.Note\_Error generic formal procedure . . . . . TIO-106

Io.Pop\_Error procedure . . . . . TIO-68

Io.Reset\_Error procedure . . . . . TIO-81

Io.Set\_Error procedure . . . . . TIO-87

Io.Standard\_Error function . . . . . TIO-99

Io\_Exceptions.Data\_Error exception

Input\_Syntax\_Error . . . . . DIO-30, TIO-156

Input\_Value\_Error . . . . . DIO-30, TIO-156

Output\_Type\_Error . . . . . DIO-30, TIO-156

Output\_Value\_Error . . . . . DIO-30, TIO-156

Io\_Exceptions.Device\_Error exception

Device\_Data\_Error . . . . . DIO-31, TIO-157

Illegal\_Heap\_Access\_Error . . . . . DIO-31, TIO-157

Illegal\_Reference\_Error . . . . . DIO-31, TIO-157

Page\_Nonexistent\_Error . . . . . DIO-31, TIO-157

Write\_To\_Read\_Only\_Page\_Error . . . . . DIO-31, TIO-157

Io\_Exceptions.Layout\_Error exception

Column\_Error . . . . . DIO-33, TIO-159

Illegal\_Position\_Error . . . . . DIO-33, TIO-159

Item\_Length\_Error . . . . . DIO-33, TIO-159

Io\_Exceptions.Mode\_Error exception

Illegal\_Operation\_On\_Infile . . . . . DIO-34, TIO-160

Illegal\_Operation\_On\_Outfile . . . . . DIO-34, TIO-160



error (continued)

Io.Exceptions.Name_Error exception	
Ambiguous_Name_Error . . . . .	DIO-35, TIO-161
Illformed_Name_Error . . . . .	DIO-35, TIO-161
Nonexistent_Directory_Error . . . . .	DIO-35, TIO-161
Nonexistent_Object_Error . . . . .	DIO-35, TIO-161
Nonexistent_Version_Error . . . . .	DIO-35, TIO-161
Io.Exceptions.Status_Error exception	
Already_Open_Error . . . . .	DIO-36, TIO-162
Not_Open_Error . . . . .	DIO-36, TIO-162
Io.Exceptions.Use_Error exception	
Access_Error . . . . .	DIO-37, TIO-163
Capacity_Error . . . . .	DIO-37, TIO-163
Check_Out_Error . . . . .	DIO-37, TIO-163
Class_Error . . . . .	DIO-37, TIO-163
Frozen_Error . . . . .	DIO-37, TIO-163
Line_Page_Length_Error . . . . .	DIO-37, TIO-163
Lock_Error . . . . .	DIO-37, TIO-163
Reset_Error . . . . .	DIO-37, TIO-163
Unsupported_Error . . . . .	DIO-37, TIO-163
Log.Filter_Errors renamed procedure . . . . .	SJM-39
Log.Pop_Error renamed procedure . . . . .	SJM-44
Standard.Constraint_Error exception . . . . .	PT-161
Standard.Numeric_Error exception . . . . .	PT-161
Standard.Program_Error exception . . . . .	PT-161
Standard.Storage_Error exception . . . . .	PT-161
Standard.Tasking_Error exception . . . . .	PT-161
System.Assertion_Error exception . . . . .	PT-164
System.Capability_Error exception . . . . .	PT-164
System.Tape_Error exception . . . . .	PT-167
see also Bell, Constraint_Error exception, Numeric_Error exception, Storage_Error exception, Unknown_Key	
error condition, severity	
Simple_Status.Condition_Class type . . . . .	PT-130
Error enumeration	
Check.Status . . . . .	PM-179
Error exception	
Profile.Error . . . . .	SJM-94
Error_Reaction type . . . . .	SJM-96, SJM-97
Tape.Error . . . . .	SMU-286
error file . . . . .	DIO-3
current	
Io.Current_Error function . . . . .	TIO-22
Io.Pop_Error procedure . . . . .	TIO-68
Io.Reset_Error procedure . . . . .	TIO-81
Io.Set_Error procedure . . . . .	TIO-87
Log.Pop_Error renamed procedure . . . . .	SJM-44
Log.Reset_Error renamed procedure . . . . .	SJM-52
Log.Set_Error renamed procedure . . . . .	SJM-57
standard . . . . .	TIO-3
Io.Standard_Error function . . . . .	TIO-99
Error function	
Simple_Status.Error . . . . .	PT-136
Condition subtype . . . . .	SJM-177
error log, stable-storage . . . . .	SMU-12
error message . . . . .	SJM-3
Simple_Status.Message function . . . . .	PT-140

error message handling	
Simple_Status package	PT-127
error name	
Simple_Status.Name function	PT-141
error reactions	DIO-6, LM-5, PM-128, SMU-2, TIO-6
Access_List package	LM-30
Archive package	LM-99
Compilation package	LM-131
Library package	LM-195
Switches package	LM-307
Error renamed exception	
Library.Error	LM-236
error severity	
Simple_Status.Severity function	PT-142
Error_Log client	SMU-12
Error_Msg enumeration	
Profile.Msg_Kind type	SJM-118
Error_Name function	
System_Uilities.Error_Name	SMU-215
Error_Reaction profile attribute	SJM-73
Error_Reaction type	
Profile.Error_Reaction	SJM-96
Reaction session switch	SJM-243
Error_Type function	
Simple_Status.Error_Type	PT-138
Errors constant	
Profile.Errors	SJM-95
<ERRORS> special value	SJM-75
Escape session switch	SJM-235
Escape_On_Break session switch	SJM-235
Eval function	
Concurrent_Map_Generic.Eval	PT-15
Map_Generic.Eval	PT-73
String_Map_Generic.Eval	ST-30
evaluation context	DEB-7, DEB-15
Debug.Catch procedure	DEB-39
Debug.Context procedure	DEB-45
Debug.Propagate procedure	DEB-99
Debug.Put procedure	DEB-101
Evaluation enumeration	
Debug.Context_Type type	DEB-48
Even enumeration	
System_Uilities.Parity_Kind type	SMU-249
event	
Debug.Stop_Event type	DEB-130
Debug.Trace_Event type	DEB-146
Exact_Copy enumeration	
Activity.Creation_Mode	PM-146
Activity.Display procedure	PM-148
examine, see Put	

Examine_Labels procedure	
Tape.Examine_Labels . . . . .	SMU-287
exception	
location	
Debug_Tools.Get_Raise_Location function . . . . .	DEB-159
message . . . . .	SJM-3
name	
Debug_Tools.Get_Exception_Name function . . . . .	DEB-157
raise	
Profile.Raise_Exception renamed function . . . . .	SJM-128
trapping . . . . .	DEB-12
Exception_Msg enumeration	
Profile.Msg_Kind type . . . . .	SJM-118
Exception_Name subtype	
Debug.Exception_Name . . . . .	DEB-57
Exception_Raised enumeration	
Debug.Trace_Event type . . . . .	DEB-146
Exception_To_Name procedure	
Debug.Exception_To_Name . . . . .	DEB-60
Debug_Tools.Get_Exception_Name function . . . . .	DEB-157
Debug_Tools.Get_Raise_Location function . . . . .	DEB-159
exceptions . . . . .	DIO-6, TIO-6
Access_List_Tools package	
Access_Tools_Error exception . . . . .	LM-55
Calendar package	
Time_Error exception . . . . .	PT-7
Concurrent_Map_Generic generic package	
Multiply_Defined exception . . . . .	PT-27
Undefined exception . . . . .	PT-34
Debug package	
Constraint_Error exception . . . . .	DEB-57
Numeric_Error exception . . . . .	DEB-57
Program_Error exception . . . . .	DEB-57
Storage_Error exception . . . . .	DEB-57
Tasking_Error exception . . . . .	DEB-57
Io_Exceptions package	
Data_Error exception . . . . .	DIO-30, TIO-156
Device_Error exception . . . . .	DIO-31, TIO-157
End_Error exception . . . . .	DIO-32, EST-149, TIO-158
Layout_Error exception . . . . .	DIO-33, TIO-159
Mode_Error exception . . . . .	DIO-34, TIO-160
Name_Error exception . . . . .	DIO-35, TIO-161
Status_Error exception . . . . .	DIO-36, TIO-162
Use_Error exception . . . . .	DIO-37, TIO-163
Library package	
Error renamed exception . . . . .	LM-236
Map_Generic generic package	
Multiply_Defined exception . . . . .	PT-85
Undefined exception . . . . .	PT-92
Profile package	
Error exception . . . . .	SJM-94
Stack_Generic generic package	
Underflow exception . . . . .	PT-158
Standard package	
Constraint_Error exception . . . . .	PT-161
Numeric_Error exception . . . . .	PT-161
Program_Error exception . . . . .	PT-161

exceptions (continued)

Standard package (continued)

Storage_Error exception . . . . .	PT-161
Tasking_Error exception . . . . .	PT-161
String_Map_Generic generic package	
Multiply_Defined exception . . . . .	ST-41
Undefined exception . . . . .	ST-47
String_Table package	
Table_Full exception . . . . .	ST-65
System package	
Assertion_Error exception . . . . .	PT-164
Capability_Error exception . . . . .	PT-164
Tape_Error exception . . . . .	PT-167
Tape package	
Error exception . . . . .	SMU-286
Window_Io.Raw package	
Unknown_Key exception . . . . .	DIO-187
<i>see also</i> Constraint_Error exception, Numeric_Error exception, Storage_Error exception	

Exceptions enumeration

Debug.Information_Type type . . . . .	DEB-75
Debug.State_Type type . . . . .	DEB-126

exchange, *see* Replace

exclamation mark (!)

special character . . . . .	DEB-18, LM-10, LM-11, PM-131, SJM-8, SMU-5
-----------------------------	--

[Execute] key

Debug.Execute procedure . . . . .	DEB-61
-----------------------------------	--------

Execute procedure

Debug.Execute . . . . .	DEB-9, DEB-16, <i>DEB-61</i>
Context procedure . . . . .	DEB-44
Hold procedure . . . . .	DEB-71
Stop procedure . . . . .	DEB-128
Xecute procedure . . . . .	DEB-148
Editor.Macro.Execute . . . . .	EI-7, EI-37, <i>EI-38</i>

execution . . . . .

setup for compiling multiple subsystems . . . . .	PM-51
state messages . . . . .	DEB-136

Expand procedure

Common.Expand . . . . .	<i>EST-85</i> , PM-190, PM-193
Library package . . . . .	LM-205
Links package . . . . .	LM-277
menu images . . . . .	EST-134
session switches . . . . .	SJM-249
Switches package . . . . .	LM-316
What package . . . . .	SJM-254
xref images . . . . .	EST-163
Common.Object.Expand . . . . .	PM-137
Editor.Window.Expand . . . . .	EI-64, <i>EI-66</i>

expand window size

Editor.Window.Expand procedure . . . . .	EI-66
Editor.Window.Join procedure . . . . .	EI-66

Expand\_Activity procedure

Cmvc_Hierarchy.Expand_Activity . . . . .	<i>PM-334</i>
--	---------------

Expanded enumeration

Time_Uutilities.Date_Format type . . . . .	PT-180
Time_Uutilities.Time_Format type . . . . .	PT-204

expanded generation image . . . . .

PM-29

expansion . . . . .	LM-199
menu images . . . . .	EST-131
xref images . . . . .	EST-160
[Explain] key	
Common.Explain procedure . . . . .	EST-87
Explain procedure	
Common.Explain . . . . .	EST-87, PM-192
Ada images . . . . .	EST-14
command images . . . . .	EST-48
Help . . . . .	EST-126
Library package . . . . .	LM-205
Links package . . . . .	LM-277
session switches . . . . .	SJM-249
Switches package . . . . .	LM-316
What package . . . . .	SJM-254
xref images . . . . .	EST-163
Common.Object.Explain . . . . .	PM-137
Explain_Crash procedure	
Operator.Explain_Crash . . . . .	SMU-74
exponent	
Io.Fixed_Io.Default_Exp constant . . . . .	TIO-121
Io.Float_Io.Default_Exp constant . . . . .	TIO-133
Text_Io.Fixed_Io.Default_Exp constant . . . . .	TIO-229
Text_Io.Float_Io.Default_Exp constant . . . . .	TIO-241
export restriction files . . . . .	PM-22, PM-247
creating . . . . .	PM-70
name resolution . . . . .	PM-72
export restrictions . . . . .	PM-56, PM-69
exports . . . . .	PM-10, PM-51, PM-275
changing private parts . . . . .	PM-87
defining . . . . .	PM-54
Expunge procedure	
Library.Expunge . . . . .	LM-237
Links.Expunge . . . . .	LM-291
Expunge_Database procedure	
Cmvc_Maintenance.Expunge_Database . . . . .	PM-50, PM-350
Cmvc.Make_Uncontrolled procedure . . . . .	PM-285
External constant	
Links.External . . . . .	LM-292
external link . . . . .	LM-6
Extract function	
Bounded_String.Extract . . . . .	ST-8
Unbounded_String.Extract . . . . .	ST-111

F

Faint character attribute . . . . .	DIO-102
Fatal enumeration	
Daemon.Condition_Class type . . . . .	SMU-16
Simple_Status.Condition_Class type . . . . .	PT-130
fetch part of, <i>see</i> Extract	

Field subtype	
Io.Field . . . . .	TIO-36
Echo procedure . . . . .	TIO-28
Io.Integer_Io generic package . . . . .	TIO-143
Put procedure . . . . .	TIO-74
Text_Io.Field . . . . .	TIO-177
Text_Io.Integer_Io generic package . . . . .	TIO-251
Field type	
Library.Field . . . . .	LM-239
Field_List type	
Table_Formatter.Field_List . . . . .	ST-95
fields	
Library.All_Fields constant . . . . .	LM-211
Fields type	
Library.Fields . . . . .	LM-241
FIFO	
Queue_Generic generic package . . . . .	PT-95
Queue_Generic.Queue type . . . . .	PT-109
File	
client . . . . .	SMU-13, SMU-17, SMU-20, SMU-27, SMU-35
object manager . . . . .	SMU-11, SMU-58
file	
access classes . . . . .	DIO-1, TIO-1
Access_List.Read constant . . . . .	LM-21
Access_List.Write constant . . . . .	LM-43
Access_List_Tools.Read constant . . . . .	LM-48
Access_List_Tools.Read constant . . . . .	LM-80
Access_List_Tools.Write constant . . . . .	LM-85
append	
File_Uilities.Append procedure . . . . .	LM-171
association	
Direct_Io.Close procedure . . . . .	DIO-8
Io.Close procedure . . . . .	TIO-14
Polymorphic_Sequential_Io.Close procedure . . . . .	DIO-41
Sequential_Io.Close procedure . . . . .	DIO-62
Text_Io.Close procedure . . . . .	TIO-166
Window_Io.Close procedure . . . . .	DIO-107
class . . . . .	LM-14, LM-17, LM-308, SJM-12, SJM-15, SJM-227
comparison	
File_Uilities.Compare procedure . . . . .	LM-172
compilation	
Compilation.Compile constant . . . . .	LM-136
create binary	
Direct_Io.Create procedure . . . . .	DIO-10
Polymorphic_Sequential_Io.Create procedure . . . . .	DIO-42
Sequential_Io.Create procedure . . . . .	DIO-63
create text	
Io.Create procedure . . . . .	TIO-20
Text.Create procedure . . . . .	EST-148
Text_Io.Create procedure . . . . .	TIO-169
current default error	
Io.Current_Error function . . . . .	TIO-22
current default input	
Io.Current_Input function . . . . .	TIO-23
Text_Io.Current_Input function . . . . .	TIO-171
current default output	
Io.Current_Output function . . . . .	TIO-24
Text_Io.Current_Output function . . . . .	TIO-172

file (continued)

current output	
File_Uilities.Current_Output constant . . . . .	LM-175
default	
Switches.Default_File constant . . . . .	LM-324
delete	
Direct_Io.Delete procedure . . . . .	DIO-12
Io.Delete procedure . . . . .	TIO-25
Polymorphic_Sequential_Io.Delete procedure . . . . .	DIO-44
Sequential_Io.Delete procedure . . . . .	DIO-65
Text_Io.Delete procedure . . . . .	TIO-173
difference	
File_Uilities.Difference procedure . . . . .	LM-176
end of	
Direct_Io.End_Of_File function . . . . .	DIO-14
Io.End_Of_File function . . . . .	TIO-33
Polymorphic_Sequential_Io.End_Of_File function . . . . .	DIO-45
Sequential_Io.End_Of_File function . . . . .	DIO-67
Text_Io.End_Of_File function . . . . .	TIO-174
Window_Io.End_Of_File function . . . . .	DIO-118
force characters to	
Io.Flush procedure . . . . .	TIO-39
handle . . . . .	DIO-4, DIO-79, TIO-4
get, <i>see</i> Open	
Io package . . . . .	TIO-7
Io.File_Type type . . . . .	TIO-38
hexadecimal dump	
File_Uilities.Dump procedure . . . . .	LM-179
identical	
File_Uilities.Equal function . . . . .	LM-181
in	
Io.In_File constant . . . . .	TIO-53
index	
Direct_Io package . . . . .	DIO-7
indirect . . . . .	PM-132, PM-133
length	
Direct_Io.Size function . . . . .	DIO-27
name . . . . .	DIO-5, TIO-4
Direct_Io.Name function . . . . .	DIO-21
Io.Name function . . . . .	TIO-59
Polymorphic_Sequential_Io.Name function . . . . .	DIO-51
Sequential_Io.Name function . . . . .	DIO-73
System_Uilities.Error_Name function . . . . .	SMU-215
System_Uilities.Input_Name function . . . . .	SMU-228
System_Uilities.Output_Name function . . . . .	SMU-246
Text_Io.Name function . . . . .	TIO-189
null filename	
Direct_Io.Create procedure . . . . .	DIO-10
organization . . . . .	DIO-7
out	
Io.Out_File constant . . . . .	TIO-65
overwrite capacity	
Direct_Io.Write procedure . . . . .	DIO-28
pointer	
Direct_Io.Set_Index procedure . . . . .	DIO-26
position	
Direct_Io.Set_Index procedure . . . . .	DIO-26
processing multiple files	
Io.Wildcard_Iterator generic procedure . . . . .	TIO-105
read, <i>see also</i> Get, Get_Line	
read Boolean value from	
Io.Get procedure . . . . .	TIO-49

file (continued)

read character from	
Io.Get procedure . . . . .	TIO-41
Text_Io.Get procedure . . . . .	TIO-181
read enumeration value from	
Io.Enumeration_Io.Get procedure . . . . .	TIO-113
Text_Io.Enumeration_Io.Get procedure . . . . .	TIO-221
read fixed-point value from	
Io.Fixed_Io.Get procedure . . . . .	TIO-123
Text_Io.Fixed_Io.Get procedure . . . . .	TIO-231
read floating-point value from	
Io.Float_Io.Get procedure . . . . .	TIO-135
Io.Get procedure . . . . .	TIO-47
Text_Io.Float_Io.Get procedure . . . . .	TIO-243
read integer value from	
Io.Get procedure . . . . .	TIO-45
Io.Integer_Io.Get procedure . . . . .	TIO-146
Text_Io.Integer_Io.Get procedure . . . . .	TIO-254
read line from	
Io.Get procedure . . . . .	TIO-43
read remaining characters except terminator	
Io.Get_Line function . . . . .	TIO-50
read string from	
Io.Get procedure . . . . .	TIO-42
Text_Io.Get procedure . . . . .	TIO-182
read string from single line except terminator	
Io.Get_Line procedure . . . . .	TIO-51
Text_Io.Get_Line procedure . . . . .	TIO-183
read, with different types of data	
Polymorphic_Sequential_Io package . . . . .	DIO-39
read-only access	
Direct_Io.File_Mode type . . . . .	DIO-15
Io.File_Mode subtype . . . . .	TIO-37
Polymorphic_Sequential_Io.File_Mode type . . . . .	DIO-46
Sequential_Io.File_Mode type . . . . .	DIO-68
Text_Io.File_Mode type . . . . .	TIO-178
read/write access	
Direct_Io.File_Mode type . . . . .	DIO-15
restore . . . . .	LM-89
safe type . . . . .	DIO-4
Direct_Io package . . . . .	DIO-7
Polymorphic_Sequential_Io package . . . . .	DIO-39
Sequential_Io package . . . . .	DIO-61
size	
Direct_Io.End_Of_File function . . . . .	DIO-14
Direct_Io.Size function . . . . .	DIO-27
stack . . . . .	TIO-8
Io package . . . . .	TIO-7
Io.Pop_Error procedure . . . . .	TIO-68
Io.Pop_Input procedure . . . . .	TIO-69
Io.Pop_Output procedure . . . . .	TIO-70
Io.Reset_Error procedure . . . . .	TIO-81
Io.Reset_Input procedure . . . . .	TIO-82
Io.Reset_Output procedure . . . . .	TIO-83
Io.Set_Error procedure . . . . .	TIO-87
Io.Set_Input procedure . . . . .	TIO-89
Io.Set_Output procedure . . . . .	TIO-94
standard error . . . . .	TIO-7
Io.Standard_Error function . . . . .	TIO-99
System_Uilities.Error_Name function . . . . .	SMU-215



file (continued)

standard input . . . . .	TIO-3, TIO-7
Io.Standard_Input function . . . . .	TIO-100
System_Uilities.Input_Name function . . . . .	SMU-228
Text_Io.Standard_Input function . . . . .	TIO-213
standard output . . . . .	TIO-3, TIO-7
Io.Standard_Output function . . . . .	TIO-101
System_Uilities.Output_Name function . . . . .	SMU-246
Text_Io.Standard_Output function . . . . .	TIO-214
storage . . . . .	DIO-2, TIO-2
subclass attributes . . . . .	LM-17, SJM-15
temporary	
Direct_Io.Create procedure . . . . .	DIO-10
Polymorphic_Sequential_Io.Create procedure . . . . .	DIO-42
Sequential_Io.Create procedure . . . . .	DIO-63
text . . . . .	TIO-7
trace	
Debug.Trace_To_File procedure . . . . .	DEB-147
wildcards	
Io.Wildcard_Iterator generic procedure . . . . .	TIO-105
write, <i>see also</i> Put, Put_line	
write Boolean value to	
Io.Put procedure . . . . .	TIO-78
write character to	
Io.Put procedure . . . . .	TIO-72
Io.Text_Io.Put procedure . . . . .	TIO-198
write enumeration value to	
Io.Enumeration_Io.Put procedure . . . . .	TIO-115
Text_Io.Enumeration_Io.Put procedure . . . . .	TIO-223
write fixed-point value to	
Io.Fixed_Io.Put procedure . . . . .	TIO-127
Text_Io.Fixed_Io.Put procedure . . . . .	TIO-235
write floating-point value to	
Io.Float_Io.Put procedure . . . . .	TIO-139
Io.Put procedure . . . . .	TIO-76
Text_Io.Float_Io.Put procedure . . . . .	TIO-247
write integer value to	
Io.Integer_Io.Put procedure . . . . .	TIO-150
Io.Put procedure . . . . .	TIO-74
Text_Io.Integer_Io.Put procedure . . . . .	TIO-258
write string to	
Io.Put procedure . . . . .	TIO-73
Text_Io.Put procedure . . . . .	TIO-199
write string to/advance line	
Io.Put_Line procedure . . . . .	TIO-79
Text_Io.Put_Line procedure . . . . .	TIO-200
write, with different types of data	
Polymorphic_Sequential_Io package . . . . .	DIO-39
write-only access	
Direct_Io.File_Mode type . . . . .	DIO-15
Io.File_Mode subtype . . . . .	TIO-37
Polymorphic_Sequential_Io.File_Mode type . . . . .	DIO-46
Sequential_Io.File_Mode type . . . . .	DIO-68
Text_Io.File_Mode type . . . . .	TIO-178
File enumeration	
Text.Image_Kind type . . . . .	EST-150
File_List renamed procedure	
Library.File_List . . . . .	LM-242
File_Map subclass . . . . .	LM-17, SJM-15

File_Mode subtype	
Io.File_Mode . . . . .	TIO-37
File_Mode type	
Direct_Io.File_Mode . . . . .	DIO-15
Polymorphic_Sequential_Io.File_Mode . . . . .	DIO-46
Sequential_Io.File_Mode . . . . .	DIO-68
Text_Io.File_Mode . . . . .	TIO-178
Window_Io.File_Mode . . . . .	DIO-120
File_Name subtype	
Switches.File_Name . . . . .	LM-331
File_Type type	
Direct_Io.File_Type . . . . .	DIO-7, DIO-16
Io.File_Type . . . . .	TIO-7, TIO-38
Convert function . . . . .	TIO-16, TIO-17
Polymorphic_Sequential_Io.File_Type . . . . .	DIO-39, DIO-47
Sequential_Io.File_Type . . . . .	DIO-61, DIO-69
Text_Io.File_Type . . . . .	TIO-165, TIO-179
Io.Convert function . . . . .	TIO-16
Io.Convert procedure . . . . .	TIO-18
Window_Io.File_Type . . . . .	DIO-79, DIO-121
File_Uilities package . . . . .	LM-169, PM-45
fill mode . . . . .	EI-5, EI-59
Editor.Set.Fill_Mode procedure . . . . .	EI-61
Fill procedure	
Editor.Region.Fill . . . . .	EI-47
Editor.Set.Fill_Column procedure . . . . .	EI-61
Image_Fill_Prefix session switch . . . . .	SJM-236
fill string	
Bounded_String.Set_Length procedure . . . . .	ST-19
Unbounded_String.Set_Length procedure . . . . .	ST-122
Fill_Column procedure	
Editor.Set.Fill_Column . . . . .	EI-59, EI-61
Fill_Mode procedure . . . . .	EI-61
Image_Fill_Column session switch . . . . .	SJM-236
Fill_Mode procedure	
Editor.Set.Fill_Mode . . . . .	EI-5, EI-59, EI-61
Image_Fill_Mode session switch . . . . .	SJM-236
filter	
Log.Filter procedure . . . . .	SJM-36
Profile.Default_Filter constant . . . . .	SJM-85
Profile.Filter function . . . . .	SJM-98
Profile.Log_Filter type . . . . .	SJM-112
Profile.Set_Default_Filter procedure . . . . .	SJM-143, SJM-145
Profile.Set_Filter procedure . . . . .	SJM-154, SJM-156
Filter function	
Profile.Filter . . . . .	SJM-98
Filter procedure	
Log.Filter . . . . .	SJM-36
Filter_Errors renamed procedure	
Log.Filter_Errors . . . . .	SJM-39
find, <i>see also</i> Locate, Reverse_Locate	
Find function	
String_Table.Find . . . . .	ST-54
Nil function . . . . .	ST-63

Find procedure	
Concurrent_Map_Generic.Find . . . . .	PT-16
Editor.Image.Find . . . . .	EI-25, EI-26
File_Uutilities.Find . . . . .	LM-169, LM-184
Map_Generic.Find . . . . .	PT-74
String_Map_Generic.Find . . . . .	ST-31
find value, <i>see</i> Eval	
Fine_Delta constant	
System.Fine_Delta . . . . .	PT-165
Finish procedure	
Editor.Macro.Finish . . . . .	EI-7, EI-37, EI-38
Editor.Region.Finish . . . . .	EI-45, EI-47
finished, <i>see</i> Done	
first	
List_Generic.Set_First procedure . . . . .	PT-64
First function	
List_Generic.First . . . . .	PT-52
Queue_Generic.First . . . . .	PT-101
Delete procedure . . . . .	PT-98
First_Child procedure	
Common.Object.First_Child . . . . .	EST-106, PM-137
Ada images . . . . .	EST-16
command images . . . . .	EST-50
Debugger . . . . .	DEB-6
Help . . . . .	EST-126
Library package . . . . .	LM-206
Links package . . . . .	LM-278
menu images . . . . .	EST-135
Search_List package . . . . .	SJM-211
session switches . . . . .	SJM-250
Switches package . . . . .	LM-317
text images . . . . .	EST-142
What package . . . . .	SJM-255
windows images . . . . .	EST-158
xref images . . . . .	EST-163
First_Element enumeration	
Debug.Numeric type . . . . .	DEB-84
fixed-point type, <i>see</i> Num	
fixed-point value	
read from file	
Io.Fixed_Io.Get procedure . . . . .	TIO-123
Text_Io.Fixed_Io.Get procedure . . . . .	TIO-231
read from string	
Io.Fixed_Io.Get procedure . . . . .	TIO-125
Text_Io.Fixed_Io.Get procedure . . . . .	TIO-233
write to file	
Io.Fixed_Io.Put procedure . . . . .	TIO-127
Text_Io.Fixed_Io.Put procedure . . . . .	TIO-235
write to string	
Io.Fixed_Io.Put procedure . . . . .	TIO-129
Text_Io.Fixed_Io.Put procedure . . . . .	TIO-237
Fixed_Io generic package	
Io.Fixed_Io . . . . .	TIO-119
Text_Io.Fixed_Io . . . . .	TIO-227

Flag procedure	
Debug.Flag . . . . .	DEB-16, <i>DEB-63</i>
State_Type type . . . . .	DEB-126
Flag_Errors debugger flag . . . . .	DEB-63
flags	
clear option	
Debug.Disable renamed procedure . . . . .	DEB-52
numeric . . . . .	DEB-4, DEB-16
Debug.Numeric type . . . . .	DEB-84
Debug.Set_Value procedure . . . . .	DEB-114
options . . . . .	DEB-4, DEB-16
Debug.Option type . . . . .	DEB-86
set	
Debug.Flag procedure . . . . .	DEB-63
set option	
Debug.Enable procedure . . . . .	DEB-56
Flags enumeration	
Debug.State_Type type . . . . .	DEB-126
Float type	
Standard.Float . . . . .	<i>PT-161</i>
Float_Io generic package	
Io.Float_Io . . . . .	<i>TIO-131</i>
Text_Io.Float_Io . . . . .	<i>TIO-239</i>
floating-point value	
read from file	
Io.Float_Io.Get procedure . . . . .	TIO-135
Io.Get procedure . . . . .	TIO-47
Text_Io.Float_Io.Get procedure . . . . .	TIO-243
read from string	
Io.Float_Io.Get procedure . . . . .	TIO-137
Text_Io.Float_Io.Get procedure . . . . .	TIO-245
write to current error file (Message window)	
Io.Echo procedure . . . . .	TIO-29
write to file	
Io.Float_Io.Put procedure . . . . .	TIO-139
Io.Put procedure . . . . .	TIO-76
Text_Io.Float_Io.Put procedure . . . . .	TIO-247
write to string	
Io.Float_Io.Put procedure . . . . .	TIO-141
Text_Io.Float_Io.Put procedure . . . . .	TIO-249
flow control	
bytes	
Terminal.Set_Xon_Xoff_Bytes procedure . . . . .	SMU-324
characters	
Terminal.Set_Xon_Xoff_Characters procedure . . . . .	SMU-325
receive	
System_Uilities.Receive_Flow_Control function . . . . .	SMU-252
set	
Terminal.Set_Flow_Control procedure . . . . .	SMU-311
set receive	
Terminal.Set_Receive_Flow_Control procedure . . . . .	SMU-318
Flow_Control function	
System_Uilities.Flow_Control . . . . .	<i>SMU-216</i>
Flush procedure	
Io.Flush . . . . .	<i>TIO-39</i>
Convert function . . . . .	TIO-17

Flush procedure ( <i>continued</i> )	
Io.Flush ( <i>continued</i> )	
Save procedure . . . . .	TIO-84
Log.Flush . . . . .	SJM-42
Focus procedure	
Editor.Window.Focus . . . . .	EI-64, EI-66
font . . . . .	DIO-80, DIO-82
declarations . . . . .	DIO-82
default	
Window_Io.Default_Font function . . . . .	DIO-111
Font type	
Window_Io.Font . . . . .	DIO-122
Font_At function	
Window_Io.Font_At . . . . .	DIO-123
Footer session switch . . . . .	SJM-235
Force_Logoff procedure	
Operator.Force_Logoff . . . . .	SJM-70, SMU-75
What.Users procedure . . . . .	SJM-271
foreground	
budget . . . . .	SMU-135, SMU-136
job . . . . .	SMU-132, SMU-134, SMU-135
<i>see also</i> Connect	
[Forget] key	
Debug.Forget procedure . . . . .	DEB-65
Forget procedure	
Debug.Forget . . . . .	DEB-12, DEB-13, DEB-65
Catch procedure . . . . .	DEB-37
Context procedure . . . . .	DEB-44, DEB-45
Propagate procedure . . . . .	DEB-96, DEB-97
fork, <i>see</i> Create_Job, Run_Job, Spawn	
form . . . . .	DIO-80, DIO-89
Form function	
Direct_Io.Form . . . . .	DIO-17
Io.Form . . . . .	TIO-40
Polymorphic_Sequential_Io.Form . . . . .	DIO-48
Sequential_Io.Form . . . . .	DIO-70
Text_Io.Form . . . . .	TIO-180
Window_Io.Form . . . . .	DIO-124
format	
Ada	
Library.Ada_Format constant . . . . .	LM-208
terse	
Library.Terse_Format constant . . . . .	LM-265
Time_Uilities.Date_Format type . . . . .	PT-180
Time_Uilities.Time_Format type . . . . .	PT-204
verbose	
Library.Verbose_Format constant . . . . .	LM-270
[Format] key	
Common.Format procedure . . . . .	EST-88
Format procedure	
Common.Format . . . . .	EST-88, PM-189
Ada images . . . . .	EST-14
command images . . . . .	EST-49
Library package . . . . .	LM-205

Format_Tape procedure	
Tape.Format_Tape . . . . .	SMU-288
formatting	
Table_Formatter package . . . . .	ST-91
forward	
deletion	
Editor.Char.Delete_Forward procedure . . . . .	EI-12
Editor.Line.Delete_Forward procedure . . . . .	EI-33
Editor.Word.Delete_Forward procedure . . . . .	EI-70
movement	
Editor.Cursor.Next procedure . . . . .	EI-19
Editor.Line.Next procedure . . . . .	EI-34
Editor.Window.Next procedure . . . . .	EI-67
Editor.Word.Next procedure . . . . .	EI-71
search . . . . .	EI-4
Editor.Search.Next procedure . . . . .	EI-57
search and replace . . . . .	EI-4
Editor.Search.Replace_Next procedure . . . . .	EI-57
tab	
Editor.Char.Tab_Forward procedure . . . . .	EI-14
Forward procedure	
Editor.Cursor.Forward . . . . .	EI-17, EI-18
Found function	
File_Uilities.Found . . . . .	LM-169, LM-187
frame . . . . .	DEB-8
Frames procedure	
Editor.Window.Frames . . . . .	EI-63, EI-66
Window_Frames session switch . . . . .	SJM-247
Free procedure	
Bounded_String.Free . . . . .	ST-10
Move procedure . . . . .	ST-16
List_Generic.Free . . . . .	PT-53
Is_Empty function . . . . .	PT-56
Unbounded_String.Free . . . . .	ST-103, ST-112
Move procedure . . . . .	ST-118
Freeze procedure	
Library.Freeze . . . . .	LM-244
Unfreeze procedure . . . . .	LM-268
Freeze_Tasks enumeration	
Debug.Option type . . . . .	DEB-87
frozen . . . . .	PM-8
window state . . . . .	EI-63
Frozen enumeration	
Library.Field type . . . . .	LM-239
Frozen_Error	
Io_Exceptions.Use_Error exception . . . . .	DIO-37, TIO-163
full	
backup . . . . .	SMU-191
pathname . . . . .	DEB-90
saves, see System_Backup package	
Full constant	
Profile.Full . . . . .	SJM-99
Verbose renamed function . . . . .	SJM-168

Full enumeration	
System_Backup.Kind type . . . . .	SMU-196
full-view release . . . . .	PM-30
fully qualified name . . . . .	DEB-18, LM-11, PM-131, SJM-8, SMU-5
Func_Body subclass . . . . .	LM-16, SJM-14
Func_Inst subclass . . . . .	LM-16, SJM-14
Func_Ren subclass . . . . .	LM-16, SJM-14
Func_Spec subclass . . . . .	LM-16, SJM-14
function keys . . . . .	EI-27

**G**

garbage collection, *see* disk collection

Gen_Func subclass . . . . .	LM-16, SJM-14
Gen_Pack subclass . . . . .	LM-16, SJM-14
Gen_Param subclass . . . . .	LM-16, SJM-14
Gen_Proc subclass . . . . .	LM-16, SJM-14
generation . . . . .	PM-6, PM-9, PM-25, PM-26, PM-305, PM-310, PM-317, PM-319, PM-321, PM-323, PM-340
collecting and displaying information . . . . .	PM-29
images . . . . .	PM-188, PM-193
retrieving latest at checkout . . . . .	PM-41
reverting to previous . . . . .	PM-28
generic formals . . . . .	DIO-94
generic instantiations, referencing . . . . .	DEB-24
Get function	
Access_List_Tools.Get . . . . .	LM-66
Profile.Get . . . . .	SJM-100, SJM-101
Scheduler.Get . . . . .	SMU-149
Set procedure . . . . .	SMU-170
Get procedure	
Access_List_Tools.Get . . . . .	LM-68
Max_Acl_Length constant . . . . .	LM-75
Io.Enumeration_Io.Get . . . . .	TIO-113, TIO-114
Io.Fixed_Io.Get . . . . .	TIO-123, TIO-125
Io.Float_Io.Get . . . . .	TIO-135, TIO-137
Io.Get . . . . .	TIO-41, TIO-42, TIO-43, TIO-45, TIO-47, TIO-49
Set_Col procedure . . . . .	TIO-85
Io.Integer_Io.Get . . . . .	TIO-146, TIO-148
Text_Io.Enumeration_Io.Get . . . . .	TIO-221, TIO-222
Text_Io.Fixed_Io.Get . . . . .	TIO-231, TIO-233
Text_Io.Float_Io.Get . . . . .	TIO-243, TIO-245
Text_Io.Get . . . . .	TIO-181, TIO-182
Set_Col procedure . . . . .	TIO-203
Text_Io.Integer_Io.Get . . . . .	TIO-254, TIO-256
Window_Io.Get . . . . .	DIO-125, DIO-127
Window_Io.Raw.Get . . . . .	DIO-177
Get_Access_List_Compaction function	
Daemon.Get_Access_List_Compaction . . . . .	SMU-17

Get_Archive_On_Shutdown function	
Operator.Get_Archive_On_Shutdown . . . . .	SMU-76
Get_Board_Info function	
System_Uilities.Get_Board_Info . . . . .	SMU-218
Get_Cached_Resolution procedure	
Profile.Get_Cached_Resolution . . . . .	SJM-102
Get_Consistency_Checking function	
Daemon.Get_Consistency_Checking . . . . .	SMU-18
Get_Cpu_Priority function	
Scheduler.Get_Cpu_Priority . . . . .	SMU-150
Job_Descriptor type . . . . .	SMU-160
Get_Cpu_Time_Used function	
Scheduler.Get_Cpu_Time_Used . . . . .	SMU-151
Get_Default function	
Access_List_Tools.Get_Default . . . . .	LM-70
Profile.Get_Default . . . . .	SJM-103, SJM-104
Get_Default procedure	
Access_List_Tools.Get_Default . . . . .	LM-72
Max_Acl_Length constant . . . . .	LM-75
Get_Disk_Wait_Load procedure	
Scheduler.Get_Disk_Wait_Load . . . . .	SMU-152
Set procedure . . . . .	SMU-176
State procedure . . . . .	SMU-183
Get_Errors procedure	
Ada.Get_Errors . . . . .	EST-28
Get_Exception_Name function	
Debug_Tools.Get_Exception_Name . . . . .	DEB-17, DEB-157
Get_Job function	
System_Uilities.Get_Job . . . . .	SMU-219
Get_Job_Attribute function	
Scheduler.Get_Job_Attribute . . . . .	SMU-153
Get_Job_Descriptor function	
Scheduler.Get_Job_Descriptor . . . . .	SMU-154
Get_Job_Kind function	
Scheduler.Get_Job_Kind . . . . .	SMU-132, SMU-155
Get_Job_State function	
Scheduler.Get_Job_State . . . . .	SMU-133, SMU-156
Job_Descriptor type . . . . .	SMU-160
Get_Line function	
Io.Get_Line . . . . .	TIO-50
Window_Io.Get_Line . . . . .	DIO-131
Get_Line procedure	
Io.Get_Line . . . . .	TIO-51
Set_Col procedure . . . . .	TIO-85
Text_Io.Get_Line . . . . .	TIO-183
Set_Col procedure . . . . .	TIO-203
Window_Io.Get_Line . . . . .	DIO-134
Line_Image function . . . . .	DIO-144
Get_Log_Threshold function	
Daemon.Get_Log_Threshold . . . . .	SMU-19



Get_Login_Limit function	
Operator.Get_Login_Limit . . . . .	SMU-77
Get_Notes procedure	
Cmvc.Get_Notes . . . . .	PM-244
Cmvc.Append_Notes procedure . . . . .	PM-210
Cmvc.Check_In procedure . . . . .	PM-216
Cmvc.Check_Out procedure . . . . .	PM-219
Cmvc.Put_Notes procedure . . . . .	PM-292
Get_Page_Counts procedure	
System_Uilities.Get_Page_Counts . . . . .	SMU-220
System_Uilities.Set_Page_Counts	
Set_Page_Limit procedure . . . . .	SMU-261
Get_Raise_Location function	
Debug_Tools.Get_Raise_Location . . . . .	DEB-17, DEB-159
Get_Run_Queue_Load procedure	
Scheduler.Get_Run_Queue_Load . . . . .	SMU-157
State procedure . . . . .	SMU-183
Get_Session function	
System_Uilities.Get_Session . . . . .	SMU-222
Get_Shutdown_Interval function	
Operator.Get_Shutdown_Interval . . . . .	SMU-78
Get_Size procedure	
Daemon.Get_Size . . . . .	SMU-20
Get_Snapshot_Settings procedure	
Daemon.Get_Snapshot_Settings . . . . .	SMU-21
Get_Task_Name function	
Debug_Tools.Get_Task_Name . . . . .	DEB-161
Get_Time function	
Time_Uilities.Get_Time . . . . .	PT-187
Get_Warning_Interval function	
Daemon.Get_Warning_Interval . . . . .	SMU-22
Get_Withheld_Task_Load procedure	
Scheduler.Get_Withheld_Task_Load . . . . .	SMU-158
State procedure . . . . .	SMU-183
Get_Wsl_Limits procedure	
Scheduler.Get_Wsl_Limits . . . . .	SMU-159
go, <i>see</i> Execute	
go to other part of Ada unit	
Ada.Other_Part procedure . . . . .	EST-36
goto, <i>see</i> Source	
graphics character set . . . . .	DIO-82, DIO-88, DIO-137
Graphics constant	
Window_Io.Graphics . . . . .	DIO-137
graphics utilities . . . . .	DIO-88
grave (^)	
special character . . . . .	DEB-20, LM-10, LM-12, PM-132, SJM-4, SJM-8, SJM-10, SJM-209, SMU-7
greater than	
Calendar.> function . . . . .	PT-8
Io.> function . . . . .	TIO-11

greater than/equal to	
Calendar.>= function	PT-8
Greater_Than function	
String_Utilities.Greater_Than	ST-73
GREP, <i>see</i> Find, Found	
group	
access control	LM-20
add	
Operator.Add_To_Group procedure	SMU-56
create	
Operator.Create_Group procedure	SMU-61
delete	
Operator.Delete_Group procedure	SMU-66
display	
Operator.Display_Group procedure	SJM-68, SMU-70
Network_Public	LM-20, SMU-54
Operator	SMU-54
Privileged	LM-20, SMU-54
Public	LM-20, SMU-54
remove	
Operator.Remove_From_Group procedure	SMU-82
special	SMU-54
user-defined	SMU-55
username	SMU-54
Group class	LM-14, SJM-12
Group object manager	SMU-11, SMU-58

## H

handle, file	TIO-4
get, <i>see</i> Open	
Io package	TIO-7
Io.File_Type type	TIO-38
hardware error, <i>see</i> Device_Data_Error	
hardware flow control	SMU-311
Has_Operator_Capability function	
Access_List_Tools.Has_Operator_Capability	LM-74
Hash generic formal function	
Concurrent_Map_Generic.Hash	PT-18
Map_Generic.Hash	PT-76
Hash package	PT-37
hash table mapping	
Concurrent_Map_Generic package	PT-9
Map_Generic generic package	PT-67
String_Map_Generic generic package	ST-25
Hash_String function	
String_Utilities.Hash_String	ST-74
head, <i>see</i> First	
Header procedure	
Table_Formatter.Header	ST-91, ST-96
Header session switch	SJM-236

headers, column	
Table_Formatter.Adjust type	ST-93
Held enumeration	
Debug.Task_Category type	DEB-135
held task state	DEB-9
help	EST-121
Common.Explain procedure	EST-87
facility, on-line	
commands from package Common	EST-126
designation	EST-122
determining key bindings	EST-125
help on a topic	EST-124
help on commands	EST-124
help on keys	EST-123
help using a Command window	EST-125
help using selection	EST-123
menus	EST-122
moving the cursor	EST-122
organization	EST-121
reviewing previous help messages	EST-122
special names	EST-123
images	EST-1
What.Command procedure	SJM-256
What.Does procedure	SJM-257
<i>see also</i> Complete	
[Help] key	
What.Does procedure	SJM-257
[Help On Help] key	
What.Does procedure	SJM-257
[Help On Key] key	
Editor.Key.Name procedure	EI-29
Hex_Number subtype	
Debug.Hex_Number	DEB-67
Hex_Values debugger flag	DEB-63
hexadecimal display	SMU-284
hexadecimal dump	
File_Uilities.Dump procedure	LM-179
Histories enumeration	
Debug.State_Type type	DEB-126
history	EST-59
command images	EST-46
collect	
Debug.Take_History procedure	DEB-132
Common.Redo procedure	EST-93
Common.Undo procedure	EST-99
facility	DEB-12
images	PM-188, PM-194
History procedure	
System_Backup.History	SMU-194
Id subtype	SMU-195
History_Count enumeration	
Debug.Numeric type	DEB-85

History_Display procedure	
Debug.History_Display . . . . .	DEB-12, <i>DEB-68</i>
Context procedure . . . . .	DEB-44
Debug_History_Count session switch . . . . .	SJM-233
Debug_History_Start session switch . . . . .	SJM-233
Debug_No_History_Timestamps session switch . . . . .	SJM-234
Numeric type . . . . .	DEB-85
Option type . . . . .	DEB-87
History_Entries enumeration	
Debug.Numeric type . . . . .	DEB-85
History_Start enumeration	
Debug.Numeric type . . . . .	DEB-85
hold, <i>see</i> Disable, Quiesce	
hold output	
Text.Block procedure . . . . .	EST-146
Hold procedure	
Debug.Hold . . . . .	DEB-9, <i>DEB-71</i>
Context procedure . . . . .	DEB-44
Execute procedure . . . . .	DEB-61
Stop procedure . . . . .	DEB-128
Xecute procedure . . . . .	DEB-148
hold stack . . . . .	EI-5
current selection	
Editor.Hold_Stack.Push procedure . . . . .	EI-22
move from bottom to top	
Editor.Hold_Stack.Rotate procedure . . . . .	EI-22
replace with next item	
Editor.Hold_Stack.Next procedure . . . . .	EI-22
replace with previous item	
Editor.Hold_Stack.Previous procedure . . . . .	EI-22
retrieve most recent item	
Editor.Hold_Stack.Top procedure . . . . .	EI-23
top	
Editor.Hold_Stack.Copy_Top procedure . . . . .	EI-21
Editor.Hold_Stack.Delete_Top procedure . . . . .	EI-22
Editor.Hold_Stack.Top procedure . . . . .	EI-23
transpose top two items	
Editor.Hold_Stack.Swap procedure . . . . .	EI-23
Hold_Stack package	
Editor.Hold_Stack . . . . .	EI-5, <i>EI-21</i>
[Home Library] key	
What.Home_Library procedure . . . . .	SJM-259
Home_Library function	
System_Uilities.Home_Library . . . . .	SMU-223
Home_Library procedure	
What.Home_Library . . . . .	SJM-259
horizontal layout . . . . .	DIO-93, DIO-94
Hour constant	
Time_Uilities.Hour . . . . .	PT-188
hours	
Time_Uilities.Military_Hours type . . . . .	PT-195
Hours type	
Time_Uilities.Hours . . . . .	PT-189

Hr_Mn enumeration	
Profile.Log_Prefix type . . . . .	SJM-114
Hr_Mn_Sc enumeration	
Profile.Log_Prefix type . . . . .	SJM-114
hyphen (-)	
indicating nondefault versions . . . . .	LM-198

I

I/O window	
definition . . . . .	EST-137
Io.Current_Input function . . . . .	TIO-23
Io.Current_Output function . . . . .	TIO-24
Io.Standard_Input function . . . . .	TIO-100
Io.Standard_Output function . . . . .	TIO-101
Text_Io.Current_Input function . . . . .	TIO-171
Text_Io.Current_Output function . . . . .	TIO-172
Text_Io.Standard_Input function . . . . .	TIO-213
Text_Io.Standard_Output function . . . . .	TIO-214
Id	
Program.Job_Id subtype . . . . .	SJM-186
Scheduler.Job_Id subtype . . . . .	SMU-164
System_Uilities.Job_Id subtype . . . . .	SMU-231
System_Uilities.Session_Id subtype . . . . .	SMU-258
Id subtype	
Job.Id . . . . .	<i>SJM-27</i>
System_Backup.Id . . . . .	<i>SMU-195</i>
Id_Case library switch . . . . .	LM-311
identical match, <i>see</i> Equal	
identity . . . . .	LM-19
access control . . . . .	LM-19
Program.Change_Identity procedure . . . . .	SJM-174
Idle enumeration	
Scheduler.Job_State type . . . . .	SMU-167
Idle state . . . . .	SMU-133
Ignore renamed function	
Profile.Ignore . . . . .	<i>SJM-105</i>
<IGNORE> special value . . . . .	SJM-75
Ignore_Case generic formal object	
String_Map_Generic.Ignore_Case . . . . .	<i>ST-32</i>
Ignore_Interface_Pragmas library switch . . . . .	LM-311
Ignore_Minor_Errors library switch . . . . .	LM-311
Ignore_Unsupported_Rep_Specs library switch . . . . .	LM-311
Illegal_Heap_Access_Error	
Io_Exceptions.Device_Error exception . . . . .	DIO-31, TIO-157
Illegal_Operation_On_Infile	
Io_Exceptions.Mode_Error exception . . . . .	DIO-34, TIO-160
Illegal_Operation_On_Outfile	
Io_Exceptions.Mode_Error exception . . . . .	DIO-34, TIO-160

Illegal_Position_Error	
Io_Exceptions.Layout_Error exception . . . . .	DIO-33, TIO-159
Illegal_Reference_Data_Error	
Io_Exceptions.Device_Error exception . . . . .	DIO-31, TIO-157
Illformed_Name_Error	
Io_Exceptions.Name_Error exception . . . . .	DIO-35, TIO-161
image . . . . .	DIO-79, DIO-81
active	
Editor.Window.Directory procedure . . . . .	EI-66
activity . . . . .	EST-1
Ada . . . . .	EST-1
bottom of	
Editor.Image.End_Of procedure . . . . .	EI-26
command . . . . .	EST-1, EST-45
committing . . . . .	EST-59
configuration . . . . .	PM-189
coordinates . . . . .	DIO-81
create	
Window_Io.Create procedure . . . . .	DIO-110
debugger . . . . .	EST-1
delete	
Window_Io.Delete procedure . . . . .	DIO-113
display, <i>see</i> Designation, Font	
editing operations	
Editor.Image package . . . . .	EI-25
find	
Editor.Image.Find procedure . . . . .	EI-26
generation . . . . .	PM-193
help . . . . .	EST-1
history . . . . .	PM-194
job . . . . .	EST-1
library . . . . .	EST-1
line	
Window_Io.Line_Image function . . . . .	DIO-144
links . . . . .	EST-1
marks	
Editor.Mark package . . . . .	EI-41
menu . . . . .	EST-1
name	
Window_Io.Name function . . . . .	DIO-150
read-only access	
Window_Io.File_Mode type . . . . .	DIO-120
reformat	
Library.Reformat_Image procedure . . . . .	LM-255
remembered positions	
Editor.Mark package . . . . .	EI-41
scroll down	
Editor.Image.Down procedure . . . . .	EI-25
scroll left	
Editor.Image.Left procedure . . . . .	EI-26
scroll right	
Editor.Image.Right procedure . . . . .	EI-26
scroll up	
Editor.Image.Up procedure . . . . .	EI-26
searchlist . . . . .	EST-1
session switches . . . . .	SJM-228
structure	
Ada images . . . . .	EST-3
command images . . . . .	EST-45
menu images . . . . .	EST-129

image (continued)	
structure (continued)	
text images	EST-137
Window Directory	EST-153
windows images	EST-153
xref images	EST-159
switch	EST-1
text	EST-1, EST-137
top of	
Editor.Image.Beginning_Of procedure	EI-25
types	EST-1, EST-57
updating	EST-59
value	
Switches.Value_Image subtype	LM-337
venture	EST-1
windows	EST-1, EST-153
work list	EST-1
work order	EST-2
write-only access	
Window_Io.File_Mode type	DIO-120
xref	EST-2
Image function	
Bounded_String.Image	ST-11
Profile.Image	SJM-107
String_Table.Image	ST-55
System_Uilities.Image	SMU-224
Time_Uilities.Image	PT-190
Date_Format type	PT-180
Image_Contents type	PT-192
Time_Format type	PT-204
Value function	PT-205
Unbounded_String.Image	ST-113
Window_Io.Raw.Image	DIO-179
Image generic formal function	
Debug_Tools.Image	DEB-173
Image package	
Editor.Image	EI-25
<IMAGE> special name	DEB-3, EST-58, EST-131, EST-155, LM-8, LM-130, LM-170, LM-199, LM-308, PM-128, SJM-5, SMU-2
Image_Contents type	
Time_Uilities.Image_Contents	PT-192
Image_Fill_Column session switch	
Editor.Region.Fill procedure	EI-47
Editor.Region.Justify procedure	EI-48
Image_Fill_Extra_Space session switch	
Editor.Region.Justify procedure	EI-48
Image_Fill_Indent session switch	
Editor.Region.Justify procedure	EI-48
Image_Fill_Mode session switch	
Editor.Region.Justify procedure	EI-48
Image_Fill_Prefix session switch	
Editor.Region.Fill procedure	EI-47
Editor.Region.Justify procedure	EI-48
Image_Insert_Mode session switch	
Editor.Region.Justify procedure	EI-48
Image_Kind type	
Text.Image_Kind	EST-150

Image_Tab_Stops session switch . . . . .	SJM-237
implementation changes . . . . .	PM-86
Import format (Debug.Memory_Display) . . . . .	DEB-79
Import procedure	
Cmvc.Import . . . . .	PM-63, PM-70, PM-76, PM-92, PM-246
Cmvc.Copy procedure . . . . .	PM-223
import restriction files . . . . .	PM-22, PM-247
creating . . . . .	PM-72
filenames . . . . .	PM-73
import restrictions . . . . .	PM-69
Imported_Views function	
Cmvc.Imported_Views . . . . .	PM-251
Cmvc.Build procedure . . . . .	PM-213
Cmvc.Initial procedure . . . . .	PM-258
imports . . . . .	PM-10, PM-11, PM-22, PM-51, PM-63, PM-76, PM-246, PM-251, PM-270, PM-283, PM-299, PM-301
circular . . . . .	PM-53, PM-79
consistency . . . . .	PM-78
defining . . . . .	PM-62
links . . . . .	PM-64
removing . . . . .	PM-64
in wait service execution message (Debug.Task_Display) . . . . .	DEB-137
In_File constant	
Io.In_File . . . . .	TIO-53
In_File enumeration	
Window_Io.File_Mode type . . . . .	DIO-120
In_Progress function	
Daemon.In_Progress . . . . .	SMU-23
inactive breakpoint . . . . .	DEB-11
Include procedure	
Profile.Include . . . . .	SJM-108
Include_In_Default procedure	
Profile.Include_In_Default . . . . .	SJM-110
Include_Packages enumeration	
Debug.Option type . . . . .	DEB-87
Includes function	
Profile.Updates . . . . .	SJM-109
Log.Put_Line procedure . . . . .	SJM-49
Incompatible enumeration	
Check.Status . . . . .	PM-179
increment, <i>see</i> Next	
incremental compilation	
Ada images . . . . .	EST-7
coded units . . . . .	EST-8
installed units . . . . .	EST-7
<i>see also</i> compilation	
incrementals, <i>see</i> System_Backup package	
Indent procedure	
Editor.Line.Indent . . . . .	EI-31, EI-33



index	EST-165, SJM-275
Direct_Io.Set_Index procedure	DIO-26
String_Table.Unique_Index function	ST-67
String_Uilities.Hash_String function	ST-74
Index file	LM-88, LM-122
Index function	
Direct_Io.Index	DIO-18
index functions	
Hash package	PT-37
Index generic formal type	
Table_Sort_Generic.Index	PT-173
indirect files	PM-132, PM-133
Information procedure	
Cmvc.Information	PM-63, PM-92, PM-253
Debug.Information	DEB-73
Debug_Addresses session switch	SJM-232
Information_Type type	DEB-75
Option type	DEB-86
Information_Type type	
Debug.Information_Type	DEB-75
Init procedure	
Concurrent_Map_Generic.Init	PT-19
Iterator type	PT-24
List_Generic.Init	PT-54
Iterator type	PT-57
Map_Generic.Init	PT-77
Iterator type	PT-82
Queue_Generic.Init	PT-102
Iterator type	PT-106
Set_Generic.Init	PT-117
Iterator type	PT-121
Stack_Generic.Init	PT-149
Iterator type	PT-151
String_Map_Generic.Init	ST-33
Iterator type	ST-38
String_Table.Init	ST-56
Iterator type	ST-59
System_Uilities.Init	SMU-225, SMU-226, SMU-227
Done function	SMU-210
Job_Iterator type	SMU-232
Next procedure	SMU-241
Session_Iterator type	SMU-259
Terminal_Iterator type	SMU-271
Value function	SMU-276
Initial procedure	
Cmvc.Initial	PM-20, PM-21, PM-22, PM-103, PM-256, PM-326
Cmvc_Maintenance.Make_Primary procedure	PM-351
Initialize procedure	
Concurrent_Map_Generic.Initialize	PT-21
Map_Generic.Initialize	PT-79
Queue_Generic.Initialize	PT-104
Set_Generic.Initialize	PT-118
Simple_Status.Initialize	PT-139
Condition type	PT-129
String_Map_Generic.Initialize	ST-35

initiate, <i>see</i> Run	
inline	
Ada.Make_Inline procedure . . . . .	EST-33
Inline pragma . . . . .	PM-116
input file . . . . .	DIO-3, TIO-3
current	
Io.Current_Input function . . . . .	TIO-23
Io.Pop_Input procedure . . . . .	TIO-69
Io.Reset_Input procedure . . . . .	TIO-82
Io.Set_Input procedure . . . . .	TIO-89
Log.Pop_Input renamed procedure . . . . .	SJM-45
Log.Reset_Input renamed procedure . . . . .	SJM-53
Log.Set_Input renamed procedure . . . . .	SJM-58
Text_Io.Current_Input function . . . . .	TIO-171
Text_Io.Set_Input procedure . . . . .	TIO-205
standard	
Io.Standard_Input function . . . . .	TIO-100
Text_Io.Standard_Input function . . . . .	TIO-213
input rate	
Terminal.Set_Input_Rate procedure . . . . .	SMU-312
input type	
terminal device characteristic . . . . .	SMU-299
input window	
Io.Current_Input function . . . . .	TIO-23
Io.Standard_Input function . . . . .	TIO-100
Text_Io.Current_Input function . . . . .	TIO-171
Text_Io.Standard_Input function . . . . .	TIO-213
input/output to windows	
Window_Io package . . . . .	DIO-79
Input_Count function	
System_Uilities.Input_Count . . . . .	SMU-229
Output_Count function . . . . .	SMU-245
Input_From procedure	
Editor.Set.Input_From . . . . .	EI-59, EI-61
Input_Logging_Off procedure	
Editor.Set.Input_Logging_Off . . . . .	EI-59, EI-61
Input_Logging_To procedure	
Editor.Set.Input_Logging_To . . . . .	EI-59, EI-61
Input_From procedure . . . . .	EI-61
Input_Name function	
System_Uilities.Input_Name . . . . .	SMU-228
Input_Output enumeration	
Text.Image_Kind type . . . . .	EST-150
Input_Rate function	
System_Uilities.Input_Rate . . . . .	SMU-230
Input_Syntax_Error	
Io_Exceptions.Data_Error exception . . . . .	DIO-30, TIO-156
Input_Value_Error	
Io_Exceptions.Data_Error exception . . . . .	DIO-30, TIO-156
insert, <i>see also</i> Add, Push, Unique	
insert mode . . . . .	EI-5, EI-11, EI-59
Editor.Set.Insert_Mode procedure . . . . .	EI-62

Insert procedure . . . . .	EI-11
Activity.Insert . . . . .	PM-66, PM-153
Bounded_String.Insert . . . . .	ST-12
Common.Object.Insert . . . . .	EST-108, PM-137, PM-153
Activity.Insert procedure . . . . .	PM-153
Ada images . . . . .	EST-16
Library package . . . . .	LM-206
Links package . . . . .	LM-278
Search_List package . . . . .	SJM-211
session switches . . . . .	SJM-250
Switches package . . . . .	LM-317
Switches.Insert procedure . . . . .	LM-332
windows images . . . . .	EST-158
Editor.Line.Insert . . . . .	EI-31, EI-33
Links.Insert . . . . .	LM-293
Add procedure . . . . .	LM-279
Replace procedure . . . . .	LM-298
Switches.Insert . . . . .	LM-332
Unbounded_String.Insert . . . . .	ST-114
Window_Io.Insert . . . . .	DIO-138
Designation type . . . . .	DIO-117
New_Line procedure . . . . .	DIO-151
Insert_Blank_Line procedure . . . . .	
Ada.Insert_Blank_Line . . . . .	EST-29
Insert_Character procedure . . . . .	
Editor.Char.Insert_Character . . . . .	EI-13
Quote procedure . . . . .	EI-14
Insert_File procedure . . . . .	
Common.Insert_File . . . . .	EST-90
Ada images . . . . .	EST-15
command images . . . . .	EST-49
text images . . . . .	EST-141
Insert_Mode procedure . . . . .	
Editor.Set.Insert_Mode . . . . .	EI-5, EI-59, EI-62
Image_Insert_Mode session switch . . . . .	SJM-236
Insert_String procedure . . . . .	
Editor.Char.Insert_String . . . . .	EI-13
Quote procedure . . . . .	EI-14
insertion points . . . . .	LM-196
Ada images . . . . .	EST-7
Insertion subclass . . . . .	LM-16, SJM-14
[Install (All Worlds)] key . . . . .	
Compilation.Promote procedure . . . . .	LM-157
[Install (This World)] key . . . . .	
Compilation.Promote procedure . . . . .	LM-157
install objects . . . . .	
Compilation.Demote procedure . . . . .	LM-141
Compilation.Make renamed procedure . . . . .	LM-151
Compilation.Promote procedure . . . . .	LM-157
[Install Stub] key . . . . .	
Ada.Install_Stub procedure . . . . .	EST-30
[Install Unit] key . . . . .	
Ada.Install_Unit procedure . . . . .	EST-31

Install_Stub procedure	
Ada.Install_Stub . . . . .	EST-30, PM-36
Install_Unit procedure	
Ada.Install_Unit . . . . .	EST-31
Installed enumeration	
Compilation.Unit_State type . . . . .	LM-166
installed unit state . . . . .	EST-5
incremental compilation . . . . .	EST-7
integer	
Hash.Long_Integer_To_Integer function . . . . .	PT-38
Hash.Pointer_To_Integer function . . . . .	PT-40
Hash.Pointer_To_Integer generic function . . . . .	PT-39
Hash.Pointer_To_Long_Integer function . . . . .	PT-44
Hash.Pointer_To_Long_Integer generic function . . . . .	PT-43
System.Max_Int constant . . . . .	PT-165
System.Min_Int constant . . . . .	PT-166
Integer type	
Standard.Integer . . . . .	PT-161
integer value	
read from file	
Io.Get procedure . . . . .	TIO-45
Io.Integer_Io.Get procedure . . . . .	TIO-146
Text_Io.Integer_Io.Get procedure . . . . .	TIO-254
read from string	
Io.Integer_Io.Get procedure . . . . .	TIO-148
Text_Io.Integer_Io.Get procedure . . . . .	TIO-256
write to current error file (Message window)	
Io.Echo procedure . . . . .	TIO-28
write to file	
Io.Integer_Io.Put procedure . . . . .	TIO-150
Io.Put procedure . . . . .	TIO-74
Text_Io.Integer_Io.Put procedure . . . . .	TIO-258
write to string	
Io.Integer_Io.Put procedure . . . . .	TIO-152
Text_Io.Integer_Io.Put procedure . . . . .	TIO-260
Integer_Io generic package	
Io.Integer_Io . . . . .	TIO-143
Text_Io.Integer_Io . . . . .	TIO-251
interactive, <i>see</i> Connect	
interfaces, among subsystems . . . . .	PM-10, PM-11
Internal constant	
Links.Internal . . . . .	LM-295
internal link . . . . .	LM-6
Internal_System_Diagnosis procedure	
Operator.Internal_System_Diagnosis . . . . .	SMU-79
Interpret_Control_Words enumeration	
Debug.Option type . . . . .	DEB-87
Interpret_Import_Words debugger flag . . . . .	DEB-63
Interpreter_Dump debugger flag . . . . .	DEB-63
Interpreter_Trace debugger flag . . . . .	DEB-63

Interrupt procedure	
Job.Interrupt . . . . .	<i>SJM-28</i>
Command.Spawn procedure . . . . .	<i>EST-56</i>
Connect procedure . . . . .	<i>SJM-20</i>
Debugger . . . . .	<i>DEB-3</i>
Disconnect procedure . . . . .	<i>SJM-23</i>
Scheduler.Job_Kind type . . . . .	<i>SMU-165</i>
interrupt program	
Debug.Stop procedure . . . . .	<i>DEB-128</i>
interval	
Daemon.Warning_Interval procedure . . . . .	<i>SMU-48</i>
Operator.Get_Shutdown_Interval function . . . . .	<i>SMU-78</i>
Interval function	
Daemon.Interval . . . . .	<i>SMU-24</i>
Interval type	
Time_Uilities.Interval . . . . .	<i>PT-175, PT-193</i>
Image function . . . . .	<i>PT-190</i>
Inverse character attribute . . . . .	<i>DIO-102</i>
Io package . . . . .	<i>TIO-7</i>
Io_Exceptions package . . . . .	<i>DIO-6, DIO-29, TIO-6, TIO-155</i>
Is_Empty function	
Concurrent_Map_Generic.Is_Empty . . . . .	<i>PT-22</i>
List_Generic.Is_Empty . . . . .	<i>PT-56</i>
Map_Generic.Is_Empty . . . . .	<i>PT-80</i>
Queue_Generic.Is_Empty . . . . .	<i>PT-105</i>
Set_Generic.Is_Empty . . . . .	<i>PT-119</i>
String_Map_Generic.Is_Empty . . . . .	<i>ST-36</i>
Is_Member function	
Set_Generic.Is_Member . . . . .	<i>PT-120</i>
Is_Nil function	
Concurrent_Map_Generic.Is_Nil . . . . .	<i>PT-23</i>
Map_Generic.Is_Nil . . . . .	<i>PT-81</i>
String_Map_Generic.Is_Nil . . . . .	<i>ST-37</i>
String_Table.Is_Nil . . . . .	<i>ST-57</i>
Time_Uilities.Is_Nil . . . . .	<i>PT-194</i>
Unbounded_String.Is_Nil . . . . .	<i>ST-116</i>
Is_Open function	
Direct_Io.Is_Open . . . . .	<i>DIO-19</i>
Io.Is_Open . . . . .	<i>TIO-54</i>
Polymorphic_Sequential_Io.Is_Open . . . . .	<i>DIO-49</i>
Sequential_Io.Is_Open . . . . .	<i>DIO-71</i>
Text_Io.Is_Open . . . . .	<i>TIO-185</i>
Window_Io.Is_Open . . . . .	<i>DIO-140</i>
item . . . . .	<i>EI-1</i>
next	
Editor.Cursor.Next procedure . . . . .	<i>EI-19</i>
off	
Editor.Set.Designation_Off procedure . . . . .	<i>EI-60</i>
previous	
Editor.Cursor.Previous procedure . . . . .	<i>EI-19</i>
[Item Off] key	
Editor.Set.Designation_Off procedure . . . . .	<i>EI-60</i>

Item procedure	
Table_Formatter.Item . . . . .	ST-91, ST-97
Item type	
String_Table.Item . . . . .	ST-58
Nil function . . . . .	ST-63
Item_Length_Error	
Io_Exceptions.Layout_Error exception . . . . .	DIO-33, TIO-159
iterate, <i>see</i> Init, Next	
iterator	
job	
System_Utilities.Init procedure . . . . .	SMU-226
System_Utilities.Job_Iterator type . . . . .	SMU-232
System_Utilities.Next procedure . . . . .	SMU-241
session	
System_Utilities.Init procedure . . . . .	SMU-225
System_Utilities.Next procedure . . . . .	SMU-242
System_Utilities.Session_Iterator type . . . . .	SMU-259
stepping through jobs	
System_Utilities.Done function . . . . .	SMU-210
stepping through sessions	
System_Utilities.Done function . . . . .	SMU-211
stepping through terminals	
System_Utilities.Done function . . . . .	SMU-212
terminal	
System_Utilities.Init procedure . . . . .	SMU-227
System_Utilities.Next procedure . . . . .	SMU-243
System_Utilities.Terminal_Iterator type . . . . .	SMU-271
wildcard	
Io.Wildcard_Iterator generic procedure . . . . .	TIO-105
Io.Wildcard_Iterator procedure . . . . .	TIO-108
Iterator type	
Concurrent_Map_Generic.Iterator . . . . .	PT-24
List_Generic.Iterator . . . . .	PT-57
Map_Generic.Iterator . . . . .	PT-82
Queue_Generic.Iterator . . . . .	PT-106
Set_Generic.Iterator . . . . .	PT-121
Stack_Generic.Iterator . . . . .	PT-151
String_Map_Generic.Iterator . . . . .	ST-38
String_Table.Iterator . . . . .	ST-59

**J**

job . . . . .	DEB-2, DIO-5, SJM-1, SMU-132, TIO-5
access control . . . . .	LM-19
association . . . . .	SMU-132
attached . . . . .	SMU-133
attribute	
Scheduler.Get_Job_Attribute function . . . . .	SMU-153
Scheduler.Set_Job_Attribute procedure . . . . .	SMU-179
background . . . . .	SMU-132, SMU-136
streams . . . . .	SMU-172
background streams . . . . .	SMU-137
Scheduler.Display procedure . . . . .	SMU-144
classes . . . . .	SMU-132

job (continued)

connect terminal to	
Job.Connect procedure . . . . .	SJM-20
core editor (Ce) . . . . .	SMU-132
CPU time	
Scheduler.Get_Cpu_Time_Used function . . . . .	SMU-151
create	
Program.Create_Job procedure . . . . .	SJM-178
descriptor	
Scheduler.Get_Job_Descriptor function . . . . .	SMU-154
Scheduler.Traverse_Job_Descriptors generic procedure . . . . .	SMU-185
Scheduler.Traverse_Job_Descriptors procedure . . . . .	SMU-187
detached . . . . .	SMU-133
disconnect	
Job.Disconnect procedure . . . . .	SJM-23
enabled or disabled	
Scheduler.Enabled function . . . . .	SMU-148
foreground . . . . .	SMU-132, SMU-134, SMU-135
get	
Scheduler.Get_Job_Attribute function . . . . .	SMU-153
Scheduler.Get_Job_Descriptor function . . . . .	SMU-154
Scheduler.Get_Job_Kind function . . . . .	SMU-155
Scheduler.Get_Job_State function . . . . .	SMU-156
System_Uilities.Get_Job function . . . . .	SMU-219
Id	
Scheduler.Job_Id subtype . . . . .	SMU-164
System_Uilities.Job_Id subtype . . . . .	SMU-231
identification number . . . . .	SMU-132
identifier . . . . .	SJM-19
System_Uilities.Value function . . . . .	SMU-276
images . . . . .	EST-1
interrupt	
Job.Interrupt procedure . . . . .	SJM-28
I/O	
text images . . . . .	EST-139
kill	
Job.Kill procedure . . . . .	SJM-29
kind . . . . .	SMU-132
Scheduler.Get_Job_Kind function . . . . .	SMU-155
Scheduler.Job_Kind type . . . . .	SMU-165
numbers . . . . .	SJM-19, SMU-132
object editor (Oe) . . . . .	SMU-132
priority	
System_Uilities.Priority function . . . . .	SMU-251
response profile . . . . .	DIO-6, LM-5, PM-128, SJM-3, SJM-33, SJM-75, SMU-2, SMU-55, TIO-6
restart	
Job.Enable procedure . . . . .	SJM-25
resume execution	
Scheduler.Enable procedure . . . . .	SMU-147
run	
Program.Run_Job procedure . . . . .	SJM-190
What.Jobs procedure . . . . .	SJM-260
running . . . . .	SMU-133
server . . . . .	SMU-133
state . . . . .	SMU-133
Scheduler.Get_Job_State function . . . . .	SMU-156
Scheduler.Job_State type . . . . .	SMU-167
stepping	
System_Uilities.Done function . . . . .	SMU-210
stop temporarily	
Job.Disable procedure . . . . .	SJM-21

job ( <i>continued</i> )	
stream time limits . . . . .	SMU-137
subtype	
Job.Id subtype . . . . .	SJM-27
suspend temporarily	
Scheduler.Disable . . . . .	SMU-142
terminated . . . . .	SMU-133
termination message	
Job.Set_Termination_Message procedure . . . . .	SJM-31
withheld . . . . .	SMU-133, SMU-136
working set	
limit . . . . .	SMU-139
size . . . . .	SMU-139
working set limits	
Scheduler.Set_Wsl_Limits procedure . . . . .	SMU-180
[Job Connect] key	
Job.Connect procedure . . . . .	SJM-20
[Job Disable] key	
Job.Disable procedure . . . . .	SJM-21
[Job Enable] key	
Job.Enable procedure . . . . .	SJM-25
[Job Kill] key	
Job.Kill procedure . . . . .	SJM-29
Job package . . . . .	SJM-19
Job_Context_First session switch . . . . .	SJM-237
Job_Context_Length session switch . . . . .	SJM-237
Job_Descriptor type	
Scheduler.Job_Descriptor . . . . .	SMU-160
Get_Job_Descriptor function . . . . .	SMU-154
Job_Id . . . . .	SMU-132
Job_Id subtype	
Program.Job_Id . . . . .	SJM-186
Scheduler.Job_Id . . . . .	SMU-164
Traverse_Job_Descriptors procedure . . . . .	SMU-187
System_Uilities.Job_Id . . . . .	SMU-231
Job_Iterator type	
System_Uilities.Job_Iterator . . . . .	SMU-232
Job_Kind type	
Scheduler.Job_Kind . . . . .	SMU-133, SMU-165
Get_Job_Attribute function . . . . .	SMU-153
Get_Job_Kind function . . . . .	SMU-155
Job_Descriptor type . . . . .	SMU-162
State procedure . . . . .	SMU-182
Job_Name function	
System_Uilities.Job_Name . . . . .	SMU-233
Job_Name_Length session switch . . . . .	SJM-237
Job_Name_Separator session switch . . . . .	SJM-237
Job_Number function	
Window_Io.Job_Number . . . . .	DIO-141
Set_Banner procedure . . . . .	DIO-168



Job_State type	
Scheduler.Job_State . . . . .	SMU-133, <i>SMU-167</i>
Get_Job_State function . . . . .	SMU-156
Job_Descriptor type . . . . .	SMU-160
State procedure . . . . .	SMU-182
What.Jobs procedure . . . . .	SJM-261
What.Users procedure . . . . .	SJM-272
Job_Time function	
Window_Io.Job_Time . . . . .	<i>DIO-142</i>
Set_Banner procedure . . . . .	DIO-168
Jobs procedure	
What.Jobs . . . . .	<i>SJM-260</i>
Job package . . . . .	SJM-19
join . . . . .	PM-14, PM-226, PM-260, PM-266, PM-268, PM-288, PM-310
set . . . . .	PM-38, PM-43, PM-44, PM-205, PM-308, PM-317, PM-350
<i>see also</i> Wait_For	
Join procedure	
Cmvc.Join . . . . .	PM-43, PM-44, PM-45, PM-48, <i>PM-260</i>
Cmvc.Copy procedure . . . . .	PM-223, PM-226
Cmvc.Make_Path procedure . . . . .	PM-272
Editor.Line.Join . . . . .	EI-31, <i>EI-34</i>
Editor.Window.Join . . . . .	EI-64, <i>EI-66</i>
joined . . . . .	PM-38
object	
accepting changes . . . . .	PM-41
checking out . . . . .	PM-39
creating new . . . . .	PM-42
developing with . . . . .	PM-38
keeping updated . . . . .	PM-40
permitting demotion . . . . .	PM-42
preventing automatic updating . . . . .	PM-42
retrieving latest at checkout . . . . .	PM-41
Journal_Comment_Lines enumeration	
Work_Order.Venture_Policy_Switch . . . . .	PM-400
Justify procedure	
Editor.Region.Justify . . . . .	<i>EI-48</i>
Editor.Set.Fill_Column procedure . . . . .	EI-61
Image_Fill_Extra_Space session switch . . . . .	SJM-236
Image_Fill_Prefix session switch . . . . .	SJM-236

K

keep window on screen	
Editor.Window.Promote procedure . . . . .	EI-67
key . . . . .	DIO-83
argument prefix . . . . .	DEB-4
bindings . . . . .	EI-1
determining . . . . .	EST-125
Editor.Key package . . . . .	EI-27
change default parameters	
Editor.Key.Prompt procedure . . . . .	EI-29
define	
Editor.Key.Define procedure . . . . .	EI-28
function keys . . . . .	EI-27

key ( <i>continued</i> )	
getting help on	EST-123
help on key	
Editor.Key.Name procedure	EI-29
keymap	EI-1
log keystrokes	
Editor.Set.Input_Logging_To procedure	EI-61
macros	
Editor.Macro package	EI-37
modifier keys	EI-2
names	DIO-85
prompt for	
Editor.Key.Prompt procedure	EI-29
rebinding	
Editor.Key package	EI-27
redefine	DIO-80
sequence	
Window_Io.Raw.Stream_Type type	DIO-185
simple	
Window_Io.Raw.Simple_Key subtype	DIO-184
stop logging keystrokes	
Editor.Set.Input_Logging_Off procedure	EI-61
key concepts	DEB-1, DIO-1, EI-1, EST-1, LM-1, PM-1, SJM-1, SMU-1, TIO-1
Ada images	EST-3
command images	EST-46
menu images	EST-130
text images	EST-138
Window Directory	EST-154
windows images	EST-154
xref images	EST-160
Key package	
Editor.Key	EI-27
Key procedure	
What.Key	
Editor.Key package	EI-27
Editor.Key.Define procedure	EI-28
Key type	
Window_Io.Raw.Key	DIO-181
Key_Directory session switch	SJM-237
Key_String type	
Window_Io.Raw.Key_String	DIO-182
keyboard	
input	DIO-85
macros	EI-7
Editor.Macro package	EI-37
keymap	EI-1
keystrokes	
program's access to	
Window_Io.Raw.Close procedure	DIO-172
read, typed by users	
Window_Io.Raw package	DIO-171
Keyword_Case library switch	EST-9, EST-14, LM-311
kill	
buffer	
Editor.Hold_Stack package	EI-21

kill (continued)

ring	
Editor.Hold_Stack package	EI-21
user session, <i>see</i> Force_Logoff	
<i>see also</i> Cancel, Delete_Group, Delete_User, Force_Logoff	
Kill procedure	
Debug.Kill	DEB-76
Job.Kill	SJM-29
Text.Block procedure	EST-146
What.Users procedure	SJM-271
Kill_Old_Jobs enumeration	
Debug.Option type	DEB-87
Kill_Print_Spooler procedure	
Queue.Kill_Print_Spooler	SMU-112
kind	SMU-132
job	SMU-132
Scheduler.Get_Job_Kind function	SMU-155
Scheduler.Job_Kind type	SMU-165
link	
Links.Link_Kind subtype	LM-296
message	
Profile.Msg_Kind type	SJM-117
parity	
System_Utilities.Parity_Kind type	SMU-249
Terminal.Parity_Kind subtype	SMU-304
Kind type	
Library.Kind	LM-246
System_Backup.Kind	SMU-196

L

label	
Tape.Examine_Labels procedure	SMU-287
Last_Child procedure	
Common.Object.Last_Child	EST-110, PM-138
Ada images	EST-17
command images	EST-50
Debugger	DEB-6
Help	EST-126
Library package	LM-207
Links package	LM-278
menu images	EST-135
Search_List package	SJM-212
session switches	SJM-250
Switches package	LM-317
text images	EST-143
What package	SJM-255
windows images	EST-158
xref images	EST-164
Last_Line function	
Window_Io.Last_Line	DIO-143
Report_Size procedure	DIO-166
Last_Login function	
System_Utilities.Last_Login	SMU-234

Last_Logout function	
System_Uilities.Last_Logout . . . . .	SMU-235
Last_Run function	
Daemon.Last_Run . . . . .	SMU-25
Last_Subitem procedure	
Table_Formatter.Last_Subitem . . . . .	ST-91, ST-98
Subitem procedure . . . . .	ST-101
Layout_Error exception	
Io package	
Col function . . . . .	TIO-15
Line function . . . . .	TIO-55
Page function . . . . .	TIO-66
Set_Col procedure . . . . .	TIO-86
Set_Line procedure . . . . .	TIO-92
Io.Enumeration_Io generic package	
Put procedure . . . . .	TIO-117
Io.Fixed_Io generic package	
Put procedure . . . . .	TIO-129, TIO-130
Io.Float_Io generic package	
Put procedure . . . . .	TIO-141
Io.Integer_Io generic package	
Put procedure . . . . .	TIO-152
Io_Exceptions.Layout_Error . . . . .	DIO-33, TIO-159
Text_Io package	
Col function . . . . .	TIO-167
Line function . . . . .	TIO-186
Page function . . . . .	TIO-195
Set_Col procedure . . . . .	TIO-204
Set_Line procedure . . . . .	TIO-207
Text_Io.Enumeration_Io generic package	
Put procedure . . . . .	TIO-225
Text_Io.Fixed_Io generic package	
Put procedure . . . . .	TIO-237
Text_Io.Float_Io generic package	
Put procedure . . . . .	TIO-249
Text_Io.Integer_Io generic package	
Put procedure . . . . .	TIO-260
Window_Io package	
Move_Cursor procedure . . . . .	DIO-149
leading characters	
String_Uilities.Strip function . . . . .	ST-86
String_Uilities.Strip_Leading function . . . . .	ST-87
left brace ( { )	
file utilities wildcard . . . . .	LM-172, LM-181, LM-184, LM-187
metacharacter . . . . .	EI-56
Left enumeration	
Table_Formatter.Adjust type . . . . .	ST-93
Left procedure	
Editor.Cursor.Left . . . . .	EI-17, EI-18
Editor.Image.Left . . . . .	EI-25, EI-26
Window_Shift_Overlap session switch . . . . .	SJM-248
Editor.Screen.Left . . . . .	EI-53
length . . . . .	DIO-79
error, see Item_Length_Error	
file	
Direct_Io.Size function . . . . .	DIO-27

length ( <i>continued</i> )	
line	TIO-8
Io.Line_Length function	TIO-56
Io.Set_Line_Length procedure	TIO-93
Io.Unbounded constant	TIO-103
Text_Io.Line_Length function	TIO-187
Text_Io.Set_Line_Length procedure	TIO-208
Text_Io.Unbounded constant	TIO-216
Window_Io.Line_Length function	DIO-145
maximum	
Bounded_String.Max_Length function	ST-15
Unbounded_String.Default_Maximum_Length generic formal object	ST-109
maximum ACL	
Access_List_Tools.Max_Acl_Length constant	LM-75
page	TIO-8
Io.Page_Length function	TIO-67
Io.Set_Page_Length procedure	TIO-96
Io.Unbounded constant	TIO-103
Text_Io.Page_Length function	TIO-196
Text_Io.Set_Page_Length procedure	TIO-210
Text_Io.Unbounded constant	TIO-216
set	
Bounded_String.Set_Length procedure	ST-19
Unbounded_String.Set_Length procedure	ST-122
string	
Bounded_String.String_Length subtype	ST-21
Unbounded_String.String_Length subtype	ST-123
Length function	
Bounded_String.Length	ST-14
List_Generic.Length	PT-58
String_Table.Length	ST-60
Unbounded_String.Length	ST-117
less than	
Calendar.< function	PT-8
Io.< function	TIO-10
Table_Sort_Generic.< generic formal function	PT-170
less than/equal to	
Calendar.<= function	PT-8
Less_Than function	
String_Utilities.Less_Than	ST-75
level numbers	PM-34, PM-230, PM-275, PM-303
coordinating in spec- and released-view names	PM-94
spec-view names	PM-59
Libraries enumeration	
Debug.State_Type type	DEB-126
library	LM-2, SJM-2
class	LM-15, SJM-13
compact	
Library.Compact_Library procedure	LM-212
create	
Library.Create_Directory renamed procedure	LM-222
Library.Create_World renamed procedure	LM-226
create switch file	
Switches.Define procedure	LM-325
designation	LM-198

library (continued)

display	
Library.Display procedure . . . . .	LM-234
Library.Enclosing_World procedure . . . . .	LM-235
Search_List.Display_Libraries procedure . . . . .	SJM-217
session switches . . . . .	SJM-229
editing . . . . .	LM-3
elision and expansion . . . . .	LM-199
enclosing . . . . .	DEB-18, LM-11, PM-131, SJM-9, SJM-209, SMU-6
home	
System_Uilities.Home_Library function . . . . .	SMU-223
image structure . . . . .	LM-195
image type . . . . .	LM-195
images . . . . .	EST-1
listing . . . . .	LM-3
management . . . . .	PM-9
operations for controlled objects . . . . .	PM-35
name . . . . .	LM-7, PM-127, SJM-5, SMU-1
parameter placeholders . . . . .	LM-199
referencing units . . . . .	DEB-20
root . . . . .	DEB-18, LM-10, LM-11, PM-131, SJM-8, SMU-5
session switches . . . . .	LM-200
special names . . . . .	LM-198
special values . . . . .	LM-199
subclass attributes . . . . .	LM-15, SJM-13
switch file association	
Switches.Associate procedure . . . . .	LM-318
Switches.Associated function . . . . .	LM-320
Switches.Dissociate procedure . . . . .	LM-328
switch filename	
Switches.Of_Library constant . . . . .	LM-333
switches . . . . .	EST-60, LM-1, LM-5, LM-308, LM-309, SJM-227
Ada images . . . . .	EST-9
command images . . . . .	EST-47
<i>see also</i> switches, library (for specific switch names)	
system . . . . .	LM-2
user	
What.Home_Library procedure . . . . .	SJM-259
Library class . . . . .	LM-14, SJM-12
Library package . . . . .	LM-195
Library_Break_Long_Lines session switch . . . . .	LM-200, SJM-237
Library_Capitalize session switch . . . . .	LM-200, SJM-237
Library_Indentation session switch . . . . .	LM-201, SJM-238
Library_Lazy_Realignment session switch . . . . .	LM-201, SJM-238
Library_Line_Length session switch . . . . .	LM-201, SJM-238
Library_Misc_Show_Edit_Info session switch . . . . .	LM-201, SJM-238
Library_Misc_Show_Frozen session switch . . . . .	LM-201, SJM-238
Library_Misc_Show_Retention session switch . . . . .	LM-201, SJM-238
Library_Misc_Show_Size session switch . . . . .	LM-201, SJM-238
Library_Misc_Show_Subclass session switch . . . . .	LM-201, SJM-238
Library_Misc_Show_Unit_State session switch . . . . .	LM-201, SJM-238
Library_Misc_Show_Volume session switch . . . . .	LM-201, SJM-239
Library_Shorten_Names session switch . . . . .	LM-201, SJM-239

Library_Shorten_Subclass session switch . . . . .	LM-202, SJM-239
Library_Shorten_Unit_State session switch . . . . .	LM-202, SJM-239
Library_Show_Deleted_Objects session switch . . . . .	LM-202, SJM-239
Library_Show_Deleted_Versions session switch . . . . .	LM-202, SJM-239
Library_Show_Miscellaneous session switch . . . . .	LM-202, SJM-239
Library_Show_Standard session switch . . . . .	LM-202, SJM-239
Library_Show_Subunits session switch . . . . .	LM-202, SJM-239
Library_Show_Version_Number session switch . . . . .	LM-202, SJM-240
Library_Std_Show_Class session switch . . . . .	LM-202, SJM-240
Library_Std_Show_Subclass session switch . . . . .	LM-202, SJM-240
Library_Std_Show_Unit_State session switch . . . . .	LM-202, SJM-240
Library_Uppercase session switch . . . . .	LM-203, SJM-240
LIFO	
Stack_Generic generic package . . . . .	PT-143
Stack_Generic.Stack type . . . . .	PT-156
limit	
Compilation.Change_Limit subtype . . . . .	LM-134
Operator.Get_Login_Limit function . . . . .	SMU-77
Operator.Show_Login_Limit procedure . . . . .	SMU-86
Scheduler.Get_Wsl_Limits procedure . . . . .	SMU-159
System_Utilities.Set_Page_Limit procedure . . . . .	SMU-261
limit number users logging in	
Operator.Limit_Login procedure . . . . .	SMU-80
Limit_Login procedure	
Operator.Limit_Login . . . . .	SMU-80
Get_Login_Limit function . . . . .	SMU-77
Show_Login_Limit procedure . . . . .	SMU-86
line . . . . .	
beginning of	DIO-79, TIO-7
Editor.Line.Beginning_Of procedure . . . . .	EI-32
case conversion	
Editor.Line.Capitalize procedure . . . . .	EI-32
Editor.Line.Lower_Case procedure . . . . .	EI-34
Editor.Line.Upper_Case procedure . . . . .	EI-35
center	
Editor.Line.Center procedure . . . . .	EI-32
copy	
Editor.Line.Copy procedure . . . . .	EI-32
current number	
Io.Line function . . . . .	TIO-55
Io.Set_Line procedure . . . . .	TIO-91
Text_Io.Line function . . . . .	TIO-186
Text_Io.Set_Line procedure . . . . .	TIO-206
delete	
Editor.Line.Delete procedure . . . . .	EI-32
Editor.Line.Delete_Backward procedure . . . . .	EI-33
Editor.Line.Delete_Forward procedure . . . . .	EI-33
Window_Io.Delete_Lines procedure . . . . .	DIO-115
echo	
Io.Echo_Line procedure . . . . .	TIO-32
editing operations	
Editor.Line package . . . . .	EI-31

line (continued)

end of	
Editor.Line.End_Of procedure . . . . .	EI-33
Io.End_Of_Line function . . . . .	TIO-34
Text_Io.End_Of_Line function . . . . .	TIO-175
Window_Io.End_Of_Line function . . . . .	DIO-119
get	
Io.Get_Line function . . . . .	TIO-50
Io.Get_Line procedure . . . . .	TIO-51
Text_Io.Get_Line procedure . . . . .	TIO-183
Window_Io.Get_Line function . . . . .	DIO-131
Window_Io.Get_Line procedure . . . . .	DIO-134
join current and next	
Editor.Line.Join procedure . . . . .	EI-34
last	
Window_Io.Last_Line function . . . . .	DIO-143
length . . . . .	TIO-8
Io.Set_Line_Length procedure . . . . .	TIO-93
Io.Unbounded constant . . . . .	TIO-103
Text_Io.Set_Line_Length procedure . . . . .	TIO-208
Text_Io.Unbounded constant . . . . .	TIO-216
new	
Io.New_Line procedure . . . . .	TIO-60
Text_Io.New_Line procedure . . . . .	TIO-190
Window_Io.New_Line procedure . . . . .	DIO-151
new line and indent	
Editor.Line.Indent procedure . . . . .	EI-33
new line before cursor	
Editor.Line.Insert procedure . . . . .	EI-33
new line below cursor	
Editor.Line.Open procedure . . . . .	EI-34
next	
Editor.Line.Next procedure . . . . .	EI-34
number	
What.Line procedure . . . . .	SJM-263
Xref package . . . . .	LM-341
previous	
Editor.Line.Previous procedure . . . . .	EI-34
put	
Io.Put_Line procedure . . . . .	TIO-79
Text_Io.Put_Line procedure . . . . .	TIO-200
read from file	
Io.Get procedure . . . . .	TIO-43
set	
Io.Set_Line procedure . . . . .	TIO-91
Text_Io.Set_Line procedure . . . . .	TIO-206
skip	
Io.Skip_Line procedure . . . . .	TIO-97
Text_Io.Skip_Line procedure . . . . .	TIO-211
terminal device characteristic . . . . .	SMU-299
terminator (Ascii.Lf) . . . . .	DIO-6, TIO-5
transpose current and previous	
Editor.Line.Transpose procedure . . . . .	EI-34
write to current log file	
Log.Put_Line procedure . . . . .	SJM-49
Line function	
Io.Line . . . . .	TIO-55
Text_Io.Line . . . . .	TIO-186
Line package	
Editor.Line . . . . .	EI-5, EI-31



Line procedure	
What.Line . . . . .	SJM-263
Line_Image function	
Window_Io.Line_Image . . . . .	DIO-144
Get_Line function . . . . .	DIO-131
Line_Length function	
Io.Line_Length . . . . .	TIO-56
Text_Io.Line_Length . . . . .	TIO-187
Window_Io.Line_Length . . . . .	DIO-145
Line_Length library switch . . . . .	LM-311
Line_Number subtype	
Window_Io.Line_Number . . . . .	DIO-146
Line_Page_Length_Error	
Io_Exceptions.Use_Error exception . . . . .	DIO-37, TIO-163
link . . . . .	LM-1, LM-6, LM-275, PM-64, PM-301, PM-303
access control . . . . .	LM-22
add	
Links.Add procedure . . . . .	LM-279
attributes . . . . .	LM-15, SJM-13
change	
Links.Edit procedure . . . . .	LM-290
copy	
Links.Copy procedure . . . . .	LM-282
delete	
Links.Delete procedure . . . . .	LM-284
dependents	
Links.Dependents procedure . . . . .	LM-286
display	
Links.Display procedure . . . . .	LM-288
edit	
Links.Edit procedure . . . . .	LM-290
enter cross-library link	
Links.Add procedure . . . . .	LM-279
external . . . . .	LM-275
Links.External constant . . . . .	LM-292
insert	
Links.Insert procedure . . . . .	LM-293
internal . . . . .	LM-275
Links.Internal constant . . . . .	LM-295
kind	
Links.Link_Kind subtype . . . . .	LM-296
library switches . . . . .	LM-309
name . . . . .	LM-275
Links.Link_Name subtype . . . . .	LM-297
name resolution mode . . . . .	LM-12, PM-132, SJM-10, SMU-7
remove	
Links.Delete procedure . . . . .	LM-284
Links.Expunge procedure . . . . .	LM-291
replace	
Links.Replace procedure . . . . .	LM-298
source name . . . . .	LM-275
Links.Source_Name subtype . . . . .	LM-300
source pattern	
Links.Source_Pattern subtype . . . . .	LM-301
special character grave ( ` ) . . . . .	LM-12, PM-132, SJM-10, SMU-7
update	
Links.Update procedure . . . . .	LM-302

link ( <i>continued</i> )	
visit	
Links.Visit procedure . . . . .	LM-304
world name	
Links.World_Name subtype . . . . .	LM-305
Link object manager . . . . .	SMU-11, SMU-58
Link_Kind subtype	
Links.Link_Kind . . . . .	LM-296
Link_Name subtype	
Links.Link_Name . . . . .	LM-275, LM-297
linked list	
List_Generic generic package . . . . .	PT-49
List_Generic.List type . . . . .	PT-59
links images . . . . .	EST-1
Links package . . . . .	LM-275
list	
Ada	
Library.Ada_List renamed procedure . . . . .	LM-209
file	
Library.File_List renamed procedure . . . . .	LM-242
System_Uilities.Block_List type . . . . .	SMU-201
Table_Formatter.Field_List type . . . . .	ST-95
verbose	
Library.Verbose_List renamed procedure . . . . .	LM-271
<i>see also</i> Display	
List procedure	
Archive.List . . . . .	LM-87, LM-109
Library.List . . . . .	LM-248
Ada_Format constant . . . . .	LM-208
Ada_List renamed procedure . . . . .	LM-209
All_Fields constant . . . . .	LM-211
Field type . . . . .	LM-239
Fields type . . . . .	LM-241
File_List renamed procedure . . . . .	LM-242
Terse_Format constant . . . . .	LM-265
Unfreeze procedure . . . . .	LM-268
Verbose_Format constant . . . . .	LM-270
Verbose_List renamed procedure . . . . .	LM-271
List type	
List_Generic.List . . . . .	PT-59
List_Editor package	
Work_Order.List_Editor . . . . .	PM-413
List_Generic generic package . . . . .	PT-49
listings	
library switches . . . . .	LM-309
literals	
in options . . . . .	SMU-9
load	
Scheduler.Get_Disk_Wait_Load procedure . . . . .	SMU-152
Scheduler.Get_Run_Queue_Load procedure . . . . .	SMU-157
Scheduler.Get_Withheld_Task_Load procedure . . . . .	SMU-158
Load procedure	
What.Load . . . . .	SJM-264

load view . . . . .	PM-10, PM-11, PM-52, PM-136, PM-137, PM-187
specifying compatible . . . . .	PM-94
Load_Factor subtype	
Scheduler.Load_Factor . . . . .	SMU-168
Set procedure . . . . .	SMU-176
Load_Func subclass . . . . .	LM-16, SJM-14
Load_Proc subclass . . . . .	LM-16, SJM-14
Load_View subclass . . . . .	LM-15, SJM-13
Load_Views enumeration	
Compilation.Promote_Scope type . . . . .	LM-160
loading . . . . .	PM-68
Local_Statement enumeration	
Debug.Stop_Event type . . . . .	DEB-130
locate	
String_Uilities.Reverse_Locate function . . . . .	ST-83
<i>see also</i> Find, Found	
Locate function	
String_Uilities.Locate . . . . .	ST-76
location	
display current	
Debug_Tools.Ada_Location function . . . . .	DEB-152
raise	
Debug_Tools.Get_Raise_Location function . . . . .	DEB-159
show source	
Debug.Address_To_Location procedure . . . . .	DEB-31
Window_Io.Report_Location procedure . . . . .	DIO-164
Location_To_Address procedure	
Debug.Location_To_Address . . . . .	DEB-77
Address_To_Location procedure . . . . .	DEB-31
Lock_Error	
Io_Exceptions.Use_Error exception . . . . .	DIO-37, TIO-163
Compilation.Delete procedure . . . . .	LM-140
Compilation.Demote procedure . . . . .	LM-142
Compilation.Destroy procedure . . . . .	LM-149
Compilation.Make_renamed procedure . . . . .	LM-152
Compilation.Promote procedure . . . . .	LM-158
Lock_Error on Ada unit, <i>see</i> locks	
locks . . . . .	EST-59
Ada images . . . . .	EST-9
read-only . . . . .	EST-59
text images . . . . .	EST-139
write . . . . .	EST-59
Locks procedure	
What.Lockes . . . . .	SJM-266
log	
filter . . . . .	SJM-33
reset	
Log.Reset_Log procedure . . . . .	SJM-54
session switches . . . . .	SJM-229
set	
Log.Set_Log procedure . . . . .	SJM-59
special names . . . . .	SJM-34

log ( <i>continued</i> )	
width	
Profile.Width function . . . . .	SJM-172
log failed logins	
Terminal.Set_Log_Failed_Logins procedure . . . . .	SMU-313
log file . . . . .	SJM-3, SJM-33
copy	
Log.Copy procedure . . . . .	SJM-35
default	
Profile.Default_Log_File constant . . . . .	SJM-86
filter	
Log.Filter procedure . . . . .	SJM-36
Log.Filter_Errors renamed procedure . . . . .	SJM-39
Profile.Log_Filter type . . . . .	SJM-112
filter/write	
Log.Summarize renamed procedure . . . . .	SJM-61
force output from buffer to file	
Log.Flush procedure . . . . .	SJM-42
name	
Log.Name subtype . . . . .	SJM-43
output	
Profile.Log_File function . . . . .	SJM-111
Profile.Log_Output_File type . . . . .	SJM-113
save	
Log.Save procedure . . . . .	SJM-56
set	
Profile.Set_Log_File procedure . . . . .	SJM-157
set default	
Profile.Set_Default_Log_File procedure . . . . .	SJM-146
write system message log to current	
Log.Put_Job_Messages procedure . . . . .	SJM-48
Log.Put_System_Messages procedure . . . . .	SJM-51
write to	
Log.Put_Condition procedure . . . . .	SJM-47
write to current	
Log.Put_Line procedure . . . . .	SJM-49
log off	
Editor.Quit procedure . . . . .	EI-10
Log package . . . . .	SJM-4, SJM-33
Log subclass . . . . .	LM-17, SJM-15
log threshold	
Daemon.Get_Log_Threshold function . . . . .	SMU-19
Daemon.Set_Log_Threshold procedure . . . . .	SMU-37
Daemon.Show_Log_Thresholds procedure . . . . .	SMU-39
Log_At_Sign_Msgs session switch . . . . .	SJM-241
Log_Auxiliary_Msgs session switch . . . . .	SJM-241
Log_Diagnostic_Msgs session switch . . . . .	SJM-241
Log_Dollar_Msgs session switch . . . . .	SJM-241
Log_Error_Msgs session switch . . . . .	SJM-241
Log_Exception_Msgs session switch . . . . .	SJM-241
Log_Failed_Logins function	
System_Uilities.Log_Failed_Logins . . . . .	SMU-236
Log_File function	
Profile.Log_File . . . . .	SJM-111

Log_File profile attribute . . . . .	SJM-73
Log_File session switch . . . . .	SJM-241
Log_Filter profile attribute . . . . .	SJM-73
Log_Filter type	
Profile.Log_Filter . . . . .	SJM-112
Log_Line_Width session switch . . . . .	SJM-241
Log_Negative_Msgs session switch . . . . .	SJM-242
Log_Note_Msgs session switch . . . . .	SJM-242
Log_Output_File type	
Profile.Log_Output_File . . . . .	SJM-113
Log_Position_Msgs session switch . . . . .	SJM-242
Log_Positive_Msgs session switch . . . . .	SJM-242
Log_Prefix type	
Profile.Log_Prefix . . . . .	SJM-114, SJM-242
Log_Prefix_1 session switch . . . . .	SJM-242
Log_Prefix_2 session switch . . . . .	SJM-242
Log_Prefix_3 session switch . . . . .	SJM-242
Log_Prefixes profile attribute . . . . .	SJM-73
Log_Prefixes type	
Profile.Log_Prefixes . . . . .	SJM-116
Log_Sharp_Msgs session switch . . . . .	SJM-242
Log_Threshold type	
Daemon.Log_Threshold . . . . .	SMU-26
Log_To_Disk enumeration	
Daemon.Log_Threshold type . . . . .	SMU-26
Log_Warning_Msgs session switch . . . . .	SJM-243
Logged_In function	
System_Uilities.Logged_In . . . . .	SMU-237
login	
from non-Rational type . . . . .	SMU-300
Operator.Get_Login_Limit function . . . . .	SMU-77
Operator.Limit_Login procedure . . . . .	SMU-80
Operator.Show_Login_Limit procedure . . . . .	SMU-86
System_Uilities.Disconnect_On_Failed_Login function . . . . .	SMU-208
System_Uilities.Last_Login function . . . . .	SMU-234
System_Uilities.Log_Failed_Logins function . . . . .	SMU-236
Terminal.Set_Disconnect_On_Failed_Login procedure . . . . .	SMU-309
Terminal.Set_Log_Failed_Logins procedure . . . . .	SMU-313
Terminal.Set_Login_Disabled procedure . . . . .	SMU-314
Login_Disabled function	
System_Uilities.Login_Disabled . . . . .	SMU-238
logoff	
Operator.Force_Logoff procedure . . . . .	SJM-70, SMU-75
System_Uilities.Disconnect_On_Logoff function . . . . .	SMU-209
Terminal.Set_Disconnect_On_Logoff procedure . . . . .	SMU-310
Terminal.Set_Logoff_On_Disconnect procedure . . . . .	SMU-315
Logoff_On_Disconnect function	
System_Uilities.Logoff_On_Disconnect . . . . .	SMU-239

logout	
System_Uilities.Last_Logout function	SMU-235
Long_Integer type	
Standard.Long_Integer	PT-161
Long_Integer_To_Integer function	
Hash.Long_Integer_To_Integer	PT-38
look up, <i>see</i> Unique	
Lower_Case constant	
Io.Lower_Case	TIO-57
Lower_Case function	
String_Uilities.Lower_Case	ST-78
Lower_Case procedure	
Editor.Char.Lower_Case	EI-14
Editor.Line.Lower_Case	EI-34
Editor.Region.Lower_Case	EI-48
Editor.Word.Lower_Case	EI-71
String_Uilities.Lower_Case	ST-79

## M

Machine_Instruction enumeration	
Debug.Stop_Event type	DEB-130
Machine_Level enumeration	
Debug.Option type	DEB-87
Macro package	
Editor.Macro	EI-7, EI-37
macros	EI-7
begin definition	
Editor.Macro.Start procedure	EI-39
bind	
Editor.Macro.Bind procedure	EI-38
current	EI-37
end definition	
Editor.Macro.Finish procedure	EI-38
execute	
Editor.Macro.Execute procedure	EI-38
reread macro file	
Editor.Macro.Restore procedure	EI-38
save macro file	
Editor.Macro.Save procedure	EI-39
Mail subclass	LM-17, SJM-15
Mail_Db subclass	LM-17, SJM-15
Mailbox subclass	LM-15, SJM-13
Main pragma	PM-114
main program	DEB-7
execution	PM-84
Main_Body subclass	LM-16, SJM-14
Main_Func subclass	LM-16, SJM-14
Main_Proc subclass	LM-16, SJM-14

major clients	
Actions . . . . .	SMU-13
Ada . . . . .	SMU-13
DDB . . . . .	SMU-13
Directory . . . . .	SMU-13
Disk . . . . .	SMU-13
File . . . . .	SMU-13
Snapshot . . . . .	SMU-13
Major_Clients constant	
Daemon.Major_Clients . . . . .	SMU-13, SMU-27
Status procedure . . . . .	SMU-44
Major_Indentation library switch . . . . .	LM-312
make changes permanent	
Io.Save procedure . . . . .	TIO-84
make comment	
Editor.Region.Comment procedure . . . . .	EI-46
make compiled	
Compilation.Make renamed procedure . . . . .	LM-151
Compilation.Promote procedure . . . . .	LM-157
Make function	
List_Generic.Make . . . . .	PT-60
[Make Inline] key	
Ada.Make_Inline procedure . . . . .	EST-33
Make renamed procedure	
Compilation.Make . . . . .	LM-151
Demote procedure . . . . .	LM-141
Promote procedure . . . . .	LM-157, LM-158
[Make Separate] key	
Ada.Make_Separate procedure . . . . .	EST-35
Make_Code_View procedure	
Cmvc.Make_Code_View . . . . .	PM-262
Make_Controlled procedure	
Cmvc.Make_Controlled . . . . .	PM-25, PM-36, PM-43, PM-44, PM-71, PM-264
Cmvc.Copy procedure . . . . .	PM-226
Cmvc.Make_Path procedure . . . . .	PM-272
Make_Empty procedure	
Concurrent_Map_Generic.Make_Empty . . . . .	PT-25
Is_Empty function . . . . .	PT-22
Map_Generic.Make_Empty . . . . .	PT-83
Is_Empty function . . . . .	PT-80
Queue_Generic.Make_Empty . . . . .	PT-107
Set_Generic.Make_Empty . . . . .	PT-122
Is_Empty function . . . . .	PT-119
Stack_Generic.Make_Empty . . . . .	PT-152
String_Map_Generic.Make_Empty . . . . .	ST-39
Is_Empty function . . . . .	ST-36
Make_Inline procedure	
Ada.Make_Inline . . . . .	EST-33
Make_Path procedure	
Cmvc.Make_Path . . . . .	PM-48, PM-50, PM-268, PM-326
Cmvc.Copy procedure . . . . .	PM-222
Cmvc.Merge_Changes procedure . . . . .	PM-287

Make_Primary procedure	
Cmvc_Maintenance.Make_Primary . . . . .	PM-108, PM-351
Make_Secondary procedure	
Cmvc_Maintenance.Make_Secondary . . . . .	PM-108, PM-354
Make_Separate procedure	
Ada.Make_Separate . . . . .	EST-35
Create_Body procedure . . . . .	EST-22
Make_Spec_View procedure	
Cmvc.Make_Spec_View . . . . .	PM-55, PM-58, PM-92, PM-275
Cmvc.Copy procedure . . . . .	PM-222
Make_Subpath procedure	
Cmvc.Make_Subpath . . . . .	PM-38, PM-50, PM-280
Cmvc.Copy procedure . . . . .	PM-222
Make_Uncontrolled procedure	
Cmvc.Make_Uncontrolled . . . . .	PM-35, PM-285
management, storage, <i>see</i> Allocate, Allows_Deallocation, Free, Unchecked_Deallocation	
mantissa	
System.Max_Mantissa constant . . . . .	PT-165
Manufacturers_Bad_Blocks constant	
System_Uilities.Manufacturers_Bad_Blocks . . . . .	SMU-240
map generic, concurrent	
Concurrent_Map_Generic generic package . . . . .	PT-9
Map type	
Concurrent_Map_Generic.Map . . . . .	PT-26
Map_Generic.Map . . . . .	PT-84
String_Map_Generic.Map . . . . .	ST-40
Map_Generic generic package . . . . .	PT-67
mapping	
hash string	
String_Uilities.Hash_String function . . . . .	ST-74
hash table	
Concurrent_Map_Generic package . . . . .	PT-9
Map_Generic generic package . . . . .	PT-67
String_Map_Generic generic package . . . . .	ST-25
many-to-one	
Hash package . . . . .	PT-37
<i>see also</i> Pair	
Mark package	
Editor.Mark . . . . .	EI-4, EI-41
marks . . . . .	EI-2, EI-4
move from bottom to top	
Editor.Mark.Rotate procedure . . . . .	EI-42
move to next	
Editor.Mark.Next procedure . . . . .	EI-42
move to previous	
Editor.Mark.Previous procedure . . . . .	EI-42
return to most recently set mark	
Editor.Mark.Top procedure . . . . .	EI-43
set	
Editor.Mark.Push procedure . . . . .	EI-42
stack . . . . .	EI-41



marks ( <i>continued</i> )	
top	
Editor.Mark.Copy_Top procedure . . . . .	EI-41
Editor.Mark.Delete_Top procedure . . . . .	EI-42
Editor.Mark.Top procedure . . . . .	EI-43
transpose top two marks	
Editor.Mark.Swap procedure . . . . .	EI-43
match, identical, <i>see</i> Equal	
Max version attribute . . . . .	LM-14, SJM-12
Max_Acl_Length constant	
Access_List_Tools.Max_Acl_Length . . . . .	LM-75
Get procedure . . . . .	LM-68
Get_Default procedure . . . . .	LM-72
Max_Digits constant	
System.Max_Digits . . . . .	PT-165
Max_Int constant	
System.Max_Int . . . . .	PT-165
Max_Length function	
Bounded_String.Max_Length . . . . .	ST-15
Max_Mantissa constant	
System.Max_Mantissa . . . . .	PT-165
medium-term scheduler	
Scheduler package . . . . .	SMU-131
Megabyte constant	
System.Megabyte . . . . .	PT-165
membership test	
Set_Generic.Is_Member function . . . . .	PT-120
memory scheduling . . . . .	SMU-139
page withdrawal . . . . .	SMU-140
parameters . . . . .	SMU-171, SMU-174
Memory_Count enumeration	
Debug.Numeric type . . . . .	DEB-85
Memory_Display procedure	
Debug.Memory_Display . . . . .	DEB-79
Flag procedure . . . . .	DEB-63
Numeric type . . . . .	DEB-85
Option type . . . . .	DEB-87
Memory_Dump procedure	
Debug.Memory_Dump	
Debug_Interpret_Control_Words session switch . . . . .	SJM-233
Debug_Memory_Count session switch . . . . .	SJM-234
Memory_Size constant	
System.Memory_Size . . . . .	PT-165
menu . . . . .	DIO-80, DIO-93, EST-129
definition . . . . .	DIO-95, EST-129
disconnecting from . . . . .	DIO-100
images . . . . .	EST-1
commands from package Common . . . . .	EST-133
designation . . . . .	EST-130
elision . . . . .	EST-131
expansion . . . . .	EST-131

menu ( <i>continued</i> )	
images ( <i>continued</i> )	
key concepts . . . . .	EST-130
special names . . . . .	EST-131
structure . . . . .	EST-129
selection . . . . .	DIO-94
merge	
changes . . . . .	PM-14, PM-45
files, <i>see</i> Append	
Merge procedure	
Activity.Merge . . . . .	PM-82, PM-155
File_Uilities.Merge . . . . .	LM-190
Strip procedure . . . . .	LM-193
Merge_Changes procedure	
Cmvc.Merge_Changes . . . . .	PM-45, PM-46, PM-287
Cmvc.Copy procedure . . . . .	PM-223
Cmvc.Join procedure . . . . .	PM-260
Cmvc.Make_Path procedure . . . . .	PM-268
message	
commentary . . . . .	LM-5, SJM-3
Daemon.Snapshot_Finish_Message procedure . . . . .	SMU-41
Daemon.Snapshot_Start_Message procedure . . . . .	SMU-42
Daemon.Snapshot_Warning_Message procedure . . . . .	SMU-43
daily	
What.Message procedure . . . . .	SJM-267
Debugger window	
Debug.Comment procedure . . . . .	DEB-43
Debug_Tools.Message procedure . . . . .	DEB-163
error . . . . .	LM-5, SJM-3
Profile.Errors constant . . . . .	SJM-95
exception . . . . .	LM-5, SJM-3
execution state	
Debug.Task_Display procedure . . . . .	DEB-136
handling, error	
Simple_Status package . . . . .	PT-127
progress . . . . .	LM-5, SJM-3
set termination	
Job.Set_Termination_Message procedure . . . . .	SJM-31
Simple_Status.Display_Message function . . . . .	PT-134
trace	
Debug.Trace procedure . . . . .	DEB-142
user-defined . . . . .	LM-6, SJM-4
warning . . . . .	LM-5, SJM-3
write to current log file	
Log.Put_Job_Message procedure . . . . .	SJM-48
Log.Put_System_Messages procedure . . . . .	SJM-51
Message function	
Simple_Status.Message . . . . .	PT-140
Message package . . . . .	SMU-49
Message procedure	
Debug_Tools.Message . . . . .	DEB-163
What.Message . . . . .	SJM-267
Message window . . . . .	
Io package . . . . .	TIO-7
Message.Send procedure . . . . .	SMU-50

Message window (continued)

Message.Send_All procedure . . . . .	SMU-51
write to	
Io.Current_Error function . . . . .	TIO-22
Io.Echo procedure . . . . .	TIO-26, TIO-27, TIO-28, TIO-29, TIO-31
Io.Echo_Line procedure . . . . .	TIO-32
Io.Standard_Error function . . . . .	TIO-99
metacharacters . . . . .	EI-56
asterisk (*) . . . . .	EI-56
backslash (\) . . . . .	EI-57
brackets ([ ]) . . . . .	EI-57
caret (^) . . . . .	EI-56
dollar sign (\$) . . . . .	EI-56
left brace ( { ) . . . . .	EI-56
percent (%) . . . . .	EI-56
question mark (?) . . . . .	EI-56
right brace ( } ) . . . . .	EI-56
<i>see also</i> substitution characters, wildcards	
Military enumeration	
Time_Uilities.Time_Format type . . . . .	PT-204
Military_Hours type	
Time_Uilities.Military_Hours . . . . .	PT-195
Milliseconds subtype	
Scheduler.Milliseconds . . . . .	SMU-169
Milliseconds type	
Time_Uilities.Milliseconds . . . . .	PT-196
Min version attribute . . . . .	LM-14, SJM-12
Min_Int constant	
System.Min_Int . . . . .	PT-166
Minor_Indentation library switch . . . . .	LM-312
minus	
Editor.Set.Argument_Minus procedure . . . . .	EI-60
Minute constant	
Time_Uilities.Minute . . . . .	PT-197
Minutes type	
Time_Uilities.Minutes . . . . .	PT-198
Mn_Dy_Yr enumeration	
Profile.Log_Prefix type . . . . .	SJM-114
mode . . . . .	PM-146, PM-148
file	
Direct_Io.File_Mode type . . . . .	DIO-15
Io.File_Mode subtype . . . . .	TIO-37
Polymorphic_Sequential_Io.File_Mode type . . . . .	DIO-46
Sequential_Io.File_Mode type . . . . .	DIO-68
Text_Io.File_Mode type . . . . .	TIO-178
Window_Io.File_Mode type . . . . .	DIO-120
fill . . . . .	EI-5, EI-59
Editor.Set.Fill_Mode procedure . . . . .	EI-61
insert . . . . .	EI-5, EI-11, EI-59
Editor.Set.Insert_Mode procedure . . . . .	EI-62
overwrite . . . . .	EI-5, EI-11, EI-59
Editor.Set.Insert_Mode procedure . . . . .	EI-62

mode (continued)	
privileged . . . . .	SMU-54
Operator.Privileged_Mode function . . . . .	SMU-81
Mode function	
Direct_Io.Mode . . . . .	DIO-20
Io.Mode . . . . .	TIO-58
Polymorphic_Sequential_Io.Mode . . . . .	DIO-50
Sequential_Io.Mode . . . . .	DIO-72
Text_Io.Mode . . . . .	TIO-188
Window_Io.Mode . . . . .	DIO-147
Mode_Error exception	
Direct_Io generic package	
End_Of_File function . . . . .	DIO-14
Read procedure . . . . .	DIO-24
Write procedure . . . . .	DIO-28
Io package	
End_Of_File function . . . . .	TIO-33
End_Of_Line function . . . . .	TIO-34
End_Of_Page function . . . . .	TIO-35
Get procedure . . . . .	TIO-41, TIO-42, TIO-44, TIO-46, TIO-48, TIO-49
Get_Line procedure . . . . .	TIO-50, TIO-52
Line_Length function . . . . .	TIO-56
New_Line procedure . . . . .	TIO-60
New_Page procedure . . . . .	TIO-61
Page_Length function . . . . .	TIO-67
Put procedure . . . . .	TIO-72, TIO-73, TIO-75, TIO-77, TIO-78
Put_Line procedure . . . . .	TIO-79
Reset procedure . . . . .	TIO-80
Set_Error procedure . . . . .	TIO-88
Set_Input procedure . . . . .	TIO-90
Set_Line_Length procedure . . . . .	TIO-93
Set_Output procedure . . . . .	TIO-95
Set_Page_Length procedure . . . . .	TIO-96
Skip_Line procedure . . . . .	TIO-97
Skip_Page procedure . . . . .	TIO-98
Io.Enumeration_Io generic package	
Get procedure . . . . .	TIO-113
Put procedure . . . . .	TIO-116
Io.Fixed_Io generic package	
Get procedure . . . . .	TIO-124
Put procedure . . . . .	TIO-128
Io.Float_Io generic package	
Get procedure . . . . .	TIO-136
Put procedure . . . . .	TIO-140
Io.Integer_Io generic package	
Get procedure . . . . .	TIO-147
Put procedure . . . . .	TIO-151
Io.Exceptions.Mode_Error . . . . .	DIO-34, TIO-160
Polymorphic_Sequential_Io package	
End_Of_File function . . . . .	DIO-45
Polymorphic_Sequential_Io.Operations package	
Read procedure . . . . .	DIO-57
Write procedure . . . . .	DIO-58
Sequential_Io package	
End_Of_File function . . . . .	DIO-67
Read procedure . . . . .	DIO-76
Write procedure . . . . .	DIO-78

Mode\_Error exception (continued)

Text_Io package	
End_Of_File function . . . . .	TIO-174
End_Of_Line function . . . . .	TIO-175
End_Of_Page function . . . . .	TIO-176
Get procedure . . . . .	TIO-181, TIO-182
Get_Line procedure . . . . .	TIO-184
Line_Length function . . . . .	TIO-187
New_Line procedure . . . . .	TIO-190
New_Page procedure . . . . .	TIO-191
Page_Length function . . . . .	TIO-196
Put procedure . . . . .	TIO-198, TIO-199
Put_Line procedure . . . . .	TIO-200
Reset procedure . . . . .	TIO-202
Set_Input procedure . . . . .	TIO-205
Set_Line_Length procedure . . . . .	TIO-208
Set_Output procedure . . . . .	TIO-209
Set_Page_Length procedure . . . . .	TIO-210
Skip_Line procedure . . . . .	TIO-211
Skip_Page procedure . . . . .	TIO-212
Text_Io.Enumeration_Io generic package	
Get procedure . . . . .	TIO-221
Put procedure . . . . .	TIO-224
Text_Io.Fixed_Io generic package	
Get procedure . . . . .	TIO-232
Put procedure . . . . .	TIO-236
Text_Io.Float_Io generic package	
Get procedure . . . . .	TIO-244
Put procedure . . . . .	TIO-248
Text_Io.Integer_Io generic package	
Get procedure . . . . .	TIO-255
Put procedure . . . . .	TIO-259
Window_Io package	
Delete procedure . . . . .	DIO-114
Delete_Lines procedure . . . . .	DIO-115
Get procedure . . . . .	DIO-126, DIO-128
Get_Line function . . . . .	DIO-132
Get_Line procedure . . . . .	DIO-135
Insert procedure . . . . .	DIO-139
New_Line procedure . . . . .	DIO-151
Overwrite procedure . . . . .	DIO-156
model	
replacing in a path . . . . .	PM-96
world . . . . .	PM-22
setting up . . . . .	PM-97
modifier keys . . . . .	EI-2
modify	
session switches	
Switches.Edit_Session_Attributes procedure . . . . .	LM-330
<i>see also</i> Demote	
[Modify] key	
Debug.Modify procedure . . . . .	DEB-81
Modify procedure	
Debug.Modify . . . . .	DEB-4, DEB-14, DEB-15, DEB-81
Context procedure . . . . .	DEB-45
Month function	
Calendar.Month . . . . .	PT-6

Month_Day_Year enumeration	
Time_Uilities.Date_Format type . . . . .	PT-180
Month_Number subtype	
Calendar.Month_Number . . . . .	PT-6
Months type	
Time_Uilities.Months . . . . .	PT-199
move	
between windows	
Editor.Window.Next procedure . . . . .	EI-67
Editor.Window.Previous procedure . . . . .	EI-67
objects . . . . .	PM-35
to next window	
Editor.Window.Child procedure . . . . .	EI-64
to previous window	
Editor.Window.Parent procedure . . . . .	EI-67
<i>see also</i> Archive package	
Move procedure	
Bounded_String.Move . . . . .	ST-16
Common.Object.Move . . . . .	EST-112, PM-36
Ada images . . . . .	EST-17
Library package . . . . .	LM-207
Links package . . . . .	LM-278
Search_List package . . . . .	SJM-212
session switches . . . . .	SJM-250
Switches package . . . . .	LM-317
text images . . . . .	EST-143
Editor.Region.Move . . . . .	EI-48
Library.Move . . . . .	LM-250
Unbounded_String.Move . . . . .	ST-118
Move_Cursor procedure	
Window_Io.Move_Cursor . . . . .	DIO-148
Insert procedure . . . . .	DIO-138
Overwrite procedure . . . . .	DIO-155
Msg_In subclass . . . . .	LM-17, SJM-15
Msg_Kind type	
Profile.Msg_Kind . . . . .	SJM-117
Msg_Out subclass . . . . .	LM-17, SJM-15
multihost development . . . . .	PM-16, PM-101
copying views among hosts . . . . .	PM-103, PM-109
moving a primary subsystem to another host . . . . .	PM-108
propagating changes across hosts . . . . .	PM-105
setting up primary and secondary subsystems . . . . .	PM-103
using CDF's with subsystems . . . . .	PM-111
multiple files, processing	
Io.Wildcard_Iterator generic procedure . . . . .	TIO-105
multiple paths . . . . .	PM-37
Multiply_Defined exception	
Concurrent_Map_Generic.Multiply_Defined . . . . .	PT-9, PT-27
Define procedure . . . . .	PT-12
Map_Generic.Multiply_Defined . . . . .	PT-67, PT-85
Define procedure . . . . .	PT-70
String_Map_Generic.Multiply_Defined . . . . .	ST-25, ST-41
Define procedure . . . . .	ST-28
multisite development . . . . .	PM-16

N

n version attribute . . . . .	LM-14, SJM-12
name . . . . .	SJM-5
Ada . . . . .	LM-7, PM-127, SJM-5
character pairs ([ ] and { }) . . . . .	LM-10, PM-131, SMU-5
class	
Queue.Class_Name subtype . . . . .	SMU-100
composite	
Switches.Composite_Name subtype . . . . .	LM-322
context	
Library.Context_Name subtype . . . . .	LM-215
display source of exception	
Debug.Exception_To_Name procedure . . . . .	DEB-60
error	
System_Uilities.Error_Name function . . . . .	SMU-215
<i>see also</i> Ambiguous_Name_Error, Illformed_Name_Error	
exception	
Debug.Exception_Name subtype . . . . .	DEB-57
Debug.Exception_To_Name procedure . . . . .	DEB-60
Debug_Tools.Get_Exception_Name function . . . . .	DEB-157
file	
Switches.File_Name subtype . . . . .	LM-331
fully qualified . . . . .	DEB-18, LM-11, PM-131, SJM-8, SMU-5
input	
System_Uilities.Input_Name function . . . . .	SMU-228
job	
System_Uilities.Job_Name function . . . . .	SMU-233
link	
Links.Link_Name subtype . . . . .	LM-297
log file	
Log.Name subtype . . . . .	SJM-43
objects . . . . .	LM-7
output	
System_Uilities.Output_Name function . . . . .	SMU-246
pathname	
Debug.Path_Name subtype . . . . .	DEB-90
session	
System_Uilities.Session_Name function . . . . .	SMU-260
simple	
Library.Simple_Name subtype . . . . .	LM-262
Simple_Status.Condition_Name type . . . . .	PT-131
Simple_Status.Create_Condition_Name function . . . . .	PT-133
source	
Links.Source_Name subtype . . . . .	LM-300
special . . . . .	LM-7, PM-127, SJM-5
special characters . . . . .	PM-131, SMU-5
string . . . . .	LM-7, PM-127, SJM-5
System.System_Name constant . . . . .	PT-166
tape	
System_Uilities.Tape_Name function . . . . .	SMU-268
task	
Debug.Set_Task_Name procedure . . . . .	DEB-112
Debug.Task_Name subtype . . . . .	DEB-140
Debug_Tools.Get_Task_Name function . . . . .	DEB-161
Debug_Tools.Set_Task_Name procedure . . . . .	DEB-179
terminal	
System_Uilities.Terminal_Name function . . . . .	SMU-272
unit	
Compilation.Unit_Name subtype . . . . .	LM-165

name ( <i>continued</i> )	
unqualified . . . . .	DEB-18
user	
System_Uilities.User_Name function . . . . .	SMU-275
world	
Links.World_Name subtype . . . . .	LM-305
<i>see also</i> naming	
Name function	
Direct_Io.Name . . . . .	DIO-21
Io.Name . . . . .	TIO-59
Polymorphic_Sequential_Io.Name . . . . .	DIO-51
Sequential_Io.Name . . . . .	DIO-73
Simple_Status.Name . . . . .	PT-141
Text_Io.Name . . . . .	TIO-189
Window_Io.Name . . . . .	DIO-150
Name generic formal type	
Allows_Deallocation.Name . . . . .	PT-1, PT-3
Unchecked_Deallocation.Name . . . . .	PT-241, PT-244
Name procedure	
Editor.Key.Name . . . . .	EI-29
Name subtype	
Access_List.Name . . . . .	LM-41
Access_List_Tools.Name . . . . .	LM-76
Compilation.Name . . . . .	LM-154
File_Uilities.Name . . . . .	LM-192
Library.Name . . . . .	LM-253
Log.Name . . . . .	SJM-43
Profile.Name . . . . .	SJM-120
Name type	
System.Name . . . . .	PT-166
Name_Error exception	
Direct_Io generic package	
Create procedure . . . . .	DIO-11
Open procedure . . . . .	DIO-22
Io package	
Append procedure . . . . .	TIO-13
Create procedure . . . . .	TIO-21
Open procedure . . . . .	TIO-64
Set_Error procedure . . . . .	TIO-88
Set_Input procedure . . . . .	TIO-90
Set_Output procedure . . . . .	TIO-95
Io_Exceptions.Name_Error . . . . .	DIO-35, TIO-161
Polymorphic_Sequential_Io package	
Append procedure . . . . .	DIO-40
Create procedure . . . . .	DIO-43
Open procedure . . . . .	DIO-52
Program package	
Current function . . . . .	SJM-184
Sequential_Io package	
Create procedure . . . . .	DIO-64
Open procedure . . . . .	DIO-75
Text_Io package	
Create procedure . . . . .	TIO-170
Open procedure . . . . .	TIO-194
naming . . . . .	
data structures . . . . .	DEB-17, PM-127
files . . . . .	DEB-21
	DIO-5, TIO-4



naming ( <i>continued</i> )	
generic instantiations . . . . .	DEB-24
objects . . . . .	LM-7, PM-127, SJM-5, SMU-1
overloaded subprograms . . . . .	DEB-23
pathnames . . . . .	DEB-17
programs . . . . .	DEB-22
referencing library units . . . . .	DEB-20
referencing overloaded subprograms . . . . .	DEB-23
referencing programs . . . . .	DEB-22
special characters . . . . .	DEB-18
unqualified names . . . . .	DEB-20
Natural subtype	
Standard.Natural . . . . .	PT-161
Negative_Msg enumeration	
Profile.Msg_Kind type . . . . .	SJM-118
Network_Public group . . . . .	LM-20, SMU-54
networking	
access control . . . . .	LM-22
library switches . . . . .	LM-309
Profile package . . . . .	SJM-76
session switches . . . . .	SJM-229
New_Line procedure	
Io.New_Line . . . . .	TIO-60
Echo procedure . . . . .	TIO-26
Echo_Line procedure . . . . .	TIO-32
Put procedure . . . . .	TIO-72
Put_Line procedure . . . . .	TIO-79
Set_Col procedure . . . . .	TIO-85
Set_Line procedure . . . . .	TIO-91
Text_Io.New_Line . . . . .	TIO-190
Put procedure . . . . .	TIO-198
Put_Line procedure . . . . .	TIO-200
Set_Col procedure . . . . .	TIO-203
Set_Line procedure . . . . .	TIO-206
Window_Io.New_Line . . . . .	DIO-151
New_Page procedure	
Io.New_Page . . . . .	TIO-61
Close procedure . . . . .	TIO-14
Set_Line procedure . . . . .	TIO-91
Text_Io.New_Page . . . . .	TIO-191
Close procedure . . . . .	TIO-166
Reset procedure . . . . .	TIO-201
Set_Line procedure . . . . .	TIO-206
New_Table function	
String_Table.New_Table . . . . .	ST-61
next	
Editor.Char.Delete_Next procedure . . . . .	EI-13
Time_Uilities.Duration_Until_Next function . . . . .	PT-186
[Next Item] key	
Editor.Cursor.Next procedure . . . . .	EI-19
Next procedure	
Common.Object.Next . . . . .	EST-113, PM-138
Ada images . . . . .	EST-4, EST-17
command images . . . . .	EST-51
Debugger . . . . .	DEB-6
Help . . . . .	EST-127

Next procedure (*continued*)

Common.Object.Next (*continued*)

Library package	LM-207
Links package	LM-278
menu images	EST-135
Search_List package	SJM-212
session switches	SJM-250
Switches package	LM-317
text images	EST-143
What package	SJM-255
windows images	EST-158
xref images	EST-164
Concurrent_Map_Generic.Next	PT-28
Iterator type	PT-24
Editor.Cursor.Next	EI-3, EI-17, EI-19, PM-190
Ada.Show_Usage procedure	EST-39
Editor.Hold_Stack.Next	EI-21, EI-22
Editor.Line.Next	EI-34
Editor.Mark.Next	EI-4, EI-41, EI-42
Editor.Screen.Next	EI-7, EI-51, EI-53
Editor.Search.Next	EI-55, EI-56, EI-57
Editor.Window.Next	EI-64, EI-67
Editor.Word.Next	EI-71
List_Generic.Next	PT-61
Init procedure	PT-54
Iterator type	PT-57
Map_Generic.Next	PT-86
Iterator type	PT-82
Queue_Generic.Next	PT-108
Init procedure	PT-102
Iterator type	PT-106
Set_Generic.Next	PT-123
Iterator type	PT-121
Stack_Generic.Next	PT-153
Init procedure	PT-149
Iterator type	PT-151
String_Map_Generic.Next	ST-42
Iterator type	ST-38
String_Table.Next	ST-62
Iterator type	ST-59
System_Uilities.Next	SMU-241, SMU-242, SMU-243
Done function	SMU-210
Init procedure	SMU-226
Job_Iterator type	SMU-232
Session_Iterator type	SMU-259
Terminal_Iterator type	SMU-271
Value function	SMU-276

[Next Prompt] key

Editor.Cursor.Next procedure	EI-19
------------------------------	-------

[Next Underline] key

Editor.Cursor.Next procedure	EI-19
------------------------------	-------

Next\_Scheduled function

Daemon.Next_Scheduled	SMU-28
-----------------------	--------

nickname

attributes	LM-15, SJM-13
overload resolution	DEB-23
task	
Debug.Context procedure	DEB-46, DEB-47
Debug.Set_Task_Name procedure	DEB-112
Debug_Tools.Set_Task_Name procedure	DEB-179

Nickname pragma . . . . .	LM-15, SJM-13
nil	
Concurrent_Map_Generic.Is_Nil function . . . . .	PT-23
Map_Generic.Is_Nil function . . . . .	PT-81
String_Map_Generic.Is_Nil function . . . . .	ST-37
String_Table.Is_Nil function . . . . .	ST-57
Time_Uilities.Is_Nil function . . . . .	PT-194
Unbounded_String.Is_Nil function . . . . .	ST-116
<i>see also</i> Is_Empty	
Nil constant	
Library.Nil . . . . .	LM-254
Nil enumeration	
Profile.Log_Prefix type . . . . .	SJM-114
Nil function	
Activity.Nil . . . . .	PM-157
Concurrent_Map_Generic.Nil . . . . .	PT-29
Initialize procedure . . . . .	PT-21
List_Generic.Nil . . . . .	PT-62
Is_Empty function . . . . .	PT-56
Map_Generic.Nil . . . . .	PT-87
String_Map_Generic.Nil . . . . .	ST-43
Initialize procedure . . . . .	ST-35
String_Table.Nil . . . . .	ST-63
Time_Uilities.Nil . . . . .	PT-200
Unbounded_String.Nil . . . . .	ST-119
Is_Nil function . . . . .	ST-116
Nil renamed function	
Profile.Nil . . . . .	SJM-121
<NIL> special value . . . . .	SJM-75
No_History_Timestamps enumeration	
Debug.Option type . . . . .	DEB-87
No_Pointers debugger flag . . . . .	DEB-63
No_Prefixes constant	
Profile.No_Prefixes . . . . .	SJM-123
No_Task_Numbers debugger flag . . . . .	DEB-63
None enumeration	
System_Uilities.Parity_Kind type . . . . .	SMU-249
Nonexistent_Directory_Error	
Io_Exceptions.Name_Error exception . . . . .	DIO-35, TIO-161
Nonexistent_Object_Error	
Io_Exceptions.Name_Error exception . . . . .	DIO-35, TIO-161
Nonexistent_Version_Error	
Io_Exceptions.Name_Error exception . . . . .	DIO-35, TIO-161
Noop procedure	
Editor.Noop . . . . .	EI-10
Normal constant	
Window_Io.Normal . . . . .	DIO-152
Normal enumeration	
Daemon.Condition_Class type . . . . .	SMU-16
Simple_Status.Condition_Class type . . . . .	PT-130
normal window state . . . . .	EI-63

Normalize function	
Access_List_Tools.Normalize . . . . .	LM-77
Not_Open_Error	
Io_Exceptions.Status_Error exception . . . . .	DIO-36, TIO-162
Not_Running enumeration	
Debug.Task_Category type . . . . .	DEB-135
note . . . . .	PM-15, PM-29, PM-232, PM-244, PM-317, PM-386, PM-387, PM-388, PM-412
Note_Error generic formal procedure	
Io.Note_Error . . . . .	TIO-106
Io.Wildcard_Iterator generic procedure . . . . .	TIO-105
Io.Wildcard_Iterator procedure . . . . .	TIO-108
Note_Msg enumeration	
Profile.Msg_Kind type . . . . .	SJM-118
Notes function	
Work_Order.Notes . . . . .	PM-386
Notes procedure	
Cmvc.Notes . . . . .	PM-29, PM-195, PM-290
Notes_List function	
Work_Order.Notes_List . . . . .	PM-387
Notes_Venture function	
Work_Order.Notes_Venture . . . . .	PM-388
notify, <i>see</i> Send, Send_All	
Notify_Warnings session switch . . . . .	SJM-243
null, <i>see</i> Is_Empty, Is_Nil, Make_Empty	
Null_Address constant	
System.Null_Address . . . . .	PT-166
Null_Device class . . . . .	LM-14, SJM-12
Null_Device object manager . . . . .	SMU-11, SMU-58
Num generic formal type	
Io.Fixed_Io.Num . . . . .	TIO-126
Get procedure . . . . .	TIO-123, TIO-124, TIO-125
Io.Float_Io.Num . . . . .	TIO-138
Get procedure . . . . .	TIO-135, TIO-136, TIO-137
Io.Integer_Io.Num . . . . .	TIO-149
Get procedure . . . . .	TIO-146, TIO-147, TIO-148
Text_Io.Fixed_Io.Num . . . . .	TIO-234
Get procedure . . . . .	TIO-231, TIO-232, TIO-233
Text_Io.Float_Io.Num . . . . .	TIO-246
Get procedure . . . . .	TIO-243, TIO-245
Text_Io.Integer_Io.Num . . . . .	TIO-257
Get procedure . . . . .	TIO-254, TIO-255, TIO-256
number	
column	
Window_Io.Column_Number subtype . . . . .	DIO-108
day	
Calendar.Day_Number subtype . . . . .	PT-6
hex	
Debug.Hex_Number subtype . . . . .	DEB-67
job	
Window_Io.Job_Number function . . . . .	DIO-141
line	
Window_Io.Line_Number subtype . . . . .	DIO-146
Xref package . . . . .	LM-341

number ( <i>continued</i> )	
month	
Calendar.Month_Number subtype . . . . .	PT-6
statement and declaration rules . . . . .	DEB-92
string to	
String_Uilities.String_To_Number procedure . . . . .	ST-84
year	
Calendar.Year_Number subtype . . . . .	PT-7
<i>see also</i> Cardinality	
Number_Base subtype	
Io.Number_Base . . . . .	TIO-62
Echo procedure . . . . .	TIO-28
Io.Integer_Io generic package . . . . .	TIO-143
Put procedure . . . . .	TIO-74
Text_Io.Number_Base . . . . .	TIO-192
Text_Io.Integer_Io generic package . . . . .	TIO-251
Number_Case library switch . . . . .	LM-312
Number_Of_Columns generic formal object	
Table_Formatter.Number_Of_Columns . . . . .	ST-99
Number_To_String function	
String_Uilities.Number_To_String . . . . .	ST-80
numeric conversion	
Debug.Convert procedure . . . . .	DEB-49
numeric flags	
set	
Debug.Numeric type . . . . .	DEB-84
Debug.Set_Value procedure . . . . .	DEB-114
numeric tag . . . . .	PM-423, PM-427
Numeric type	
Debug.Numeric . . . . .	DEB-84
Flag procedure . . . . .	DEB-63
Set_Value procedure . . . . .	DEB-114
State_Type type . . . . .	DEB-126
Numeric_Error exception	
Bounded_String package . . . . .	ST-1
Append procedure . . . . .	ST-3
Char_At function . . . . .	ST-4
Copy procedure . . . . .	ST-5
Delete procedure . . . . .	ST-7
Insert procedure . . . . .	ST-13
Replace procedure . . . . .	ST-18
Set_Length procedure . . . . .	ST-19
Debug package . . . . .	DEB-57
Standard.Numeric_Error . . . . .	PT-161
String_Table package . . . . .	ST-49
Char_At function . . . . .	ST-51
String_Uilities package . . . . .	ST-69
Equal function . . . . .	ST-72
Greater_Than function . . . . .	ST-73
Hash_String function . . . . .	ST-74
Less_Than function . . . . .	ST-75
Locate function . . . . .	ST-77
Lower_Case function . . . . .	ST-78
Lower_Case procedure . . . . .	ST-79
Reverse_Locate function . . . . .	ST-83
String_To_Number procedure . . . . .	ST-85

Numeric\_Error\_exception (continued)

String\_Utilities package (continued)

Strip function	ST-86
Strip_Leading function	ST-87
Strip_Trailing function	ST-88
Upper_Case function	ST-89
Upper_Case procedure	ST-90
Time_Utilities package	
Value function	PT-205
Unbounded_String package	
Append procedure	ST-103
Char_At function	ST-105
Char_At function	ST-106
Copy procedure	ST-107
Delete procedure	ST-110
Insert procedure	ST-115
Replace procedure	ST-121
Set_Length procedure	ST-122

O

object

binary, controlling	PM-25
class	LM-4, LM-197
configuration	
deleting	PM-49
controlled	
accessing concurrently	PM-43
deleting	PM-35
moving	PM-35
withdrawing	PM-35
copy	LM-90
Archive.Copy procedure	LM-100
deleted	LM-198
display locks	
What.Locks procedure	SJM-266
enclosing	DEB-18, LM-11, PM-131, SJM-8, SJM-9, SMU-6
error, see Nonexistent_Object_Error	
joined	
accepting changes	PM-41
checking out	PM-39
creating new	PM-42
developing with	PM-38
keeping updated	PM-40
permitting demotion	PM-42
preventing automatic updating	PM-42
retrieving latest generation at checkout	PM-41
list	
Archive.List procedure	LM-109
managers	
Action	SMU-11
Actions	SMU-58
Ada	SMU-11, SMU-58
Archived_Code	SMU-11, SMU-58
Code_Segment	SMU-11, SMU-58
Configuration	SMU-11, SMU-58
DDB	SMU-11, SMU-58
Directory	SMU-11, SMU-58
File	SMU-11, SMU-58
Group	SMU-11, SMU-58
Link	SMU-11, SMU-58

object (continued)

managers (continued)

Null_Device . . . . .	SMU-11, SMU-58
Pipe . . . . .	SMU-11, SMU-58
Session . . . . .	SMU-11, SMU-58
Tape . . . . .	SMU-11, SMU-58
Terminal . . . . .	SMU-11, SMU-58
User . . . . .	SMU-11, SMU-58
name . . . . .	LM-7, PM-127, SJM-5, SMU-1
object editor (Oe) job . . . . .	SMU-132
pathname	
What.Object procedure . . . . .	SJM-268
referring to deleted . . . . .	LM-13, PM-133, SJM-11, SMU-8
restore . . . . .	LM-89
Archive.Restore procedure . . . . .	LM-112
retention count . . . . .	LM-197
save . . . . .	LM-88
Archive.Save procedure . . . . .	LM-122
severed	
merging changes . . . . .	PM-45
rejoining . . . . .	PM-45
size . . . . .	LM-4
status . . . . .	LM-4
subclass . . . . .	LM-4, LM-197
unit state . . . . .	LM-197
version number . . . . .	LM-4

Object enumeration

Library.Field type . . . . .	LM-239
------------------------------	--------

Object generic formal type

Allows_Deallocation.Object . . . . .	PT-1, PT-4
Unchecked_Deallocation.Object . . . . .	PT-241, PT-245

Object package

Common.Object . . . . .	EST-101
-------------------------	---------

Object procedure

What.Object . . . . .	SJM-268
-----------------------	---------

Object subtype

System_Uilities.Object . . . . .	SMU-244
----------------------------------	---------

[Object] - [I] key combination

Common.Object.Insert procedure . . . . .	EST-108
see also insertion points	

Objects subclass

LM-17, SJM-15

obsolesced link

LM-6

Odd enumeration

System_Uilities.Parity_Kind type . . . . .	SMU-249
--	---------

Oe enumeration

Scheduler.Job_Kind type . . . . .	SMU-165
-----------------------------------	---------

Of\_Library constant

Switches.Of_Library . . . . .	LM-333
-------------------------------	--------

Of\_Session constant

Switches.Of_Session . . . . .	LM-334
-------------------------------	--------

off

Debug.Debug_Off procedure . . . . .	DEB-51
Debug_Tools.Debug_Off procedure . . . . .	DEB-154

Off procedure	
Editor.Region.Off . . . . .	EI-45, <i>EI-49</i>
On procedure . . . . .	EI-49
on	
Debug_Tools.Debug_On procedure . . . . .	DEB-155
On procedure	
Editor.Region.On . . . . .	EI-45, <i>EI-49</i>
online help facility, <i>see help</i>	
open	
Command window, <i>see Create_Command</i>	
Direct_Io.Is_Open function . . . . .	DIO-19
error, <i>see Already_Open_Error, Not_Open_Error</i>	
Io.Is_Open function . . . . .	TIO-54
Polymorphic_Sequential_Io.Is_Open function . . . . .	DIO-49
private part . . . . .	PM-89, PM-113
Sequential_Io.Is_Open function . . . . .	DIO-71
Text_Io.Is_Open function . . . . .	TIO-185
Window_Io.Is_Open function . . . . .	DIO-140
<i>see also Edit</i>	
Open procedure	
Direct_Io.Open . . . . .	DIO-22
Form function . . . . .	DIO-17
Editor.Line.Open . . . . .	EI-31, <i>EI-34</i>
Io.Open . . . . .	TIO-63
Form function . . . . .	TIO-40
Polymorphic_Sequential_Io.Open . . . . .	DIO-52
Form function . . . . .	DIO-48
Sequential_Io.Open . . . . .	DIO-74
Form function . . . . .	DIO-70
Text_Io.Open . . . . .	TIO-193
Form function . . . . .	TIO-180
Window_Io.Open . . . . .	DIO-153
Form function . . . . .	DIO-124
Window_Io.Raw.Open . . . . .	DIO-183
operations . . . . .	EI-1
Operations generic package	
Polymorphic_Sequential_Io.Operations . . . . .	DIO-39, <i>DIO-55</i>
operator capability . . . . .	LM-20, SMU-54
Access_List_Tools.Has_Operator_Capability function . . . . .	LM-74
Operator group . . . . .	SMU-54
Operator package . . . . .	<i>SJM-65, SMU-53</i>
Optimize_Generic_History enumeration	
Debug.Option type . . . . .	DEB-87
option	
clear flag	
Debug.Disable renamed procedure . . . . .	DEB-52
set flag	
Debug.Enable procedure . . . . .	DEB-56
Debug.Option type . . . . .	DEB-86
Option type	
Debug.Option . . . . .	DEB-3, <i>DEB-86</i>
Catch procedure . . . . .	DEB-37
Flag procedure . . . . .	DEB-63



Option type ( <i>continued</i> )	
Debug.Option ( <i>continued</i> )	
Stack procedure . . . . .	DEB-123
State_Type type . . . . .	DEB-126
options	
Boolean . . . . .	LM-18, SJM-16, SMU-9
literals . . . . .	LM-19, SJM-17, SMU-9
specification . . . . .	LM-17, SJM-15, SMU-8
Options parameter . . . . .	LM-17, LM-88, LM-90, LM-169, SJM-15, SMU-8
restore . . . . .	LM-89
save . . . . .	LM-88
Options session switch . . . . .	SJM-243
order, <i>see</i> Sort	
ordering	
Table_Sort_Generic.< generic formal function . . . . .	PT-170
origin	
Window_Io.Report_Origin procedure . . . . .	DIO-165
[Other Part In Place] key	
Ada.Other_Part procedure . . . . .	EST-36
[Other Part] key	
Ada.Other_Part procedure . . . . .	EST-36
Other_Part procedure	
Ada.Other_Part . . . . .	EST-36
Out_File constant	
Io.Out_File . . . . .	TIO-65
Out_File enumeration	
Window_Io.File_Mode type . . . . .	DIO-120
output	
compressed . . . . .	LM-178
current	
File_Uilities.Current_Output constant . . . . .	LM-175
uncompressed . . . . .	LM-178
output file . . . . .	DIO-3, TIO-3
current	
Io.Current_Output function . . . . .	TIO-24
Io.Pop_Output procedure . . . . .	TIO-70
Io.Reset_Output procedure . . . . .	TIO-83
Io.Set_Output procedure . . . . .	TIO-94
Log.Pop_Output renamed procedure . . . . .	SJM-46
Log.Reset_Log procedure . . . . .	SJM-54
Log.Reset_Output renamed procedure . . . . .	SJM-55
Log.Set_Log procedure . . . . .	SJM-59
Log.Set_Output renamed procedure . . . . .	SJM-60
Text_Io.Current_Output function . . . . .	TIO-172
Text_Io.Set_Output procedure . . . . .	TIO-209
standard	
Io.Standard_Output function . . . . .	TIO-101
Text_Io.Standard_Output function . . . . .	TIO-214
output rate	
Terminal.Set_Output_Rate procedure . . . . .	SMU-316
output type	
terminal device characteristic . . . . .	SMU-299

output window	
Io.Current_Output function . . . . .	TIO-24
Io.Standard_Output function . . . . .	TIO-101
Text_Io.Current_Output function . . . . .	TIO-172
Text_Io.Standard_Output function . . . . .	TIO-214
Output_Count function	
System_Uilities.Output_Count . . . . .	SMU-245
Input_Count function . . . . .	SMU-229
Output_Name function	
System_Uilities.Output_Name . . . . .	SMU-246
Output_Rate function	
System_Uilities.Output_Rate . . . . .	SMU-247
Output_Type_Error	
Io_Exceptions.Data_Error exception . . . . .	DIO-30, TIO-156
Output_Value_Error	
Io_Exceptions.Data_Error exception . . . . .	DIO-30, TIO-156
overload resolution nickname . . . . .	DEB-23
overwrite mode . . . . .	EI-5, EI-11, EI-59
Editor.Set.Insert_Mode procedure . . . . .	EI-62
Overwrite procedure	
Window_Io.Overwrite . . . . .	DIO-155
owner access . . . . .	LM-21
Library.Freeze procedure . . . . .	LM-244
Library.Unfreeze procedure . . . . .	LM-268
Owner constant	
Access_List.Owner . . . . .	LM-42
Access_List_Tools.Owner . . . . .	LM-79

**P**

P.M.	
Time_Uilities.Sun_Positions type . . . . .	PT-202
Pack_Body subclass . . . . .	LM-16, SJM-14
Pack_Inst subclass . . . . .	LM-16, SJM-14
Pack_Ren subclass . . . . .	LM-16, SJM-14
Pack_Spec subclass . . . . .	LM-16, SJM-14
package completed execution message (Debug.Task_Display) . . . . .	DEB-137
page . . . . .	TIO-7
count	
System_Uilities.Get_Page_Counts procedure . . . . .	SMU-220
current number	
Io.Page function . . . . .	TIO-66
Text_Io.Page function . . . . .	TIO-195
default	
System_Uilities.Set_Page_Limit procedure . . . . .	SMU-261
end of	
Io.End_Of_Page function . . . . .	TIO-35
Text_Io.End_Of_Page function . . . . .	TIO-176
faults	
Scheduler.Disk_Waits function . . . . .	SMU-143

page (continued)

length	TIO-8
error, <i>see</i> Line_Page_Length_Error	
Io.Set_Page_Length procedure	TIO-96
Io.Unbounded constant	TIO-103
Text_Io.Set_Page_Length procedure	TIO-210
Text_Io.Unbounded constant	TIO-216
limits	SMU-197
System_Uilities.Set_Page_Limit procedure	SMU-261
new	
Io.New_Page procedure	TIO-61
Text_Io.New_Page procedure	TIO-191
skip	
Io.Skip_Page procedure	TIO-98
Text_Io.Skip_Page procedure	TIO-212
terminator (Ascii.Ff)	DIO-6, TIO-5
withdrawal	SMU-140
Page function	
Io.Page	TIO-66
Text_Io.Page	TIO-195
Page_Length function	
Io.Page_Length	TIO-67
Text_Io.Page_Length	TIO-196
Page_Limit	
library switch	LM-312
System_Uilities.Get_Page_Counts procedure	SMU-220
System_Uilities.Set_Page_Limit procedure	SMU-261
pragma	
System_Uilities.Get_Page_Counts procedure	SMU-220
System_Uilities.Set_Page_Limit procedure	SMU-261
Page_Nonexistent_Error	
Io_Exceptions.Device_Error exception	DIO-31, TIO-157
Pair type	
Concurrent_Map_Generic.Pair	PT-30
Map_Generic.Pair	PT-88
parallel development	PM-280
within subsystems	PM-13
parameter placeholders	LM-7, LM-8, PM-127, PM-128, SJM-5, SJM-6, SMU-1, SMU-3, SMU-55
parameters	
CPU scheduling	SMU-171, SMU-173
disk scheduling	SMU-172, SMU-176
memory scheduling	SMU-171, SMU-174
parent	
Common.Enclosing procedure	EST-83
object	
Library.Default_Keep_Versions constant	LM-229
unit	LM-11, PM-131, SJM-9, SMU-6
Parent procedure	
Common.Object.Parent	EST-115, PM-138
Ada images	EST-4, EST-17
command images	EST-51
Debugger	DEB-6
Help	EST-127
Library package	LM-207
Links package	LM-278
menu images	EST-135

Parent procedure ( <i>continued</i> )		
Common.Object.Parent ( <i>continued</i> )		
Search_List package . . . . .	SJM-212	
session switches . . . . .	SJM-250	
Switches package . . . . .	LM-317	
text images . . . . .	EST-143	
What package . . . . .	SJM-255	
windows images . . . . .	EST-158	
xref images . . . . .	EST-164	
Editor.Window.Parent . . . . .	EI-64, EI-67	
Parents function		
Cmvc_Hierarchy.Parents . . . . .	PM-335	
parity		
Terminal.Set_Parity procedure . . . . .	SMU-317	
Parity function		
System_Uilities.Parity . . . . .	SMU-248	
Parity_Kind subtype		
Terminal.Parity_Kind . . . . .	SMU-304	
Parity_Kind type		
System_Uilities.Parity_Kind . . . . .	SMU-249	
Parse procedure		
Compilation.Parse . . . . .	LM-155	
parsing text files		
Compilation.Compile constant . . . . .	LM-136	
Compilation.Parse procedure . . . . .	LM-155	
partitioning of projects . . . . .		PM-3
Password library switch . . . . .		LM-312
Password session switch . . . . .		SJM-243
passwords		
change		
Operator.Change_Password procedure . . . . .	SJM-66, SMU-60	
remote . . . . .		SJM-76
Profile.Default_Remote_Passwords function . . . . .	SJM-91	
Profile.Remote_Passwords function . . . . .	SJM-131	
Profile.Remote_Passwords_Type subtype . . . . .	SJM-132	
Profile.Set_Default_Remote_Passwords procedure . . . . .	SJM-149	
Profile.Set_Remote_Passwords procedure . . . . .	SJM-160	
path . . . . .		PM-8, PM-268
creating . . . . .		PM-47
differences between paths and subpaths . . . . .		PM-47
multiple . . . . .		PM-37
replacing model . . . . .		PM-96
setting up . . . . .		PM-326
setting up multiple development paths . . . . .		PM-47
Path_Name subtype		
Debug.Path_Name . . . . .	DEB-10, DEB-90	
pathname . . . . .		DEB-17, LM-7, PM-127, PM-139, PM-166, PM-170, PM-171, PM-172, PM-224, PM-227, SJM-5, SMU-1
display		
What.Object procedure . . . . .	SJM-268	
display for an object		
Library.Resolve procedure . . . . .	LM-258	
full . . . . .	DEB-90	

pathname (*continued*)

- patterns in . . . . . LM-8, PM-129, SJM-6, SMU-3
- prefix . . . . . PM-34, PM-47
- relative . . . . . DEB-90

pattern

- Links.Source\_Pattern subtype . . . . . LM-301

pattern matching . . . . . LM-8, LM-169, LM-276, LM-297, SJM-6

- File\_Uilities.Equal function . . . . . LM-181
- File\_Uilities.Find procedure . . . . . LM-184
- File\_Uilities.Found procedure . . . . . LM-187
- metacharacters . . . . . EI-56

peek, *see* Memory\_Display

percent (%)

- file utilities wildcard . . . . . LM-172, LM-181, LM-184, LM-187
- metacharacter . . . . . EI-56
- special character . . . . . DEB-18, DEB-19, LM-10, LM-11, SJM-8, SJM-9, SMU-6

period (.)

- special character . . . . . DEB-18, DEB-19, LM-10, LM-12, PM-132, SJM-8, SJM-10, SMU-7

period, double (..), symbol . . . . . LM-18, SJM-16, SMU-9

permanent breakpoint . . . . . DEB-11

permanent, make changes

- Io.Save procedure . . . . . TIO-84

Permanent\_Breakpoints enumeration

- Debug.Option type . . . . . DEB-88

Persevere enumeration

- Profile.Error\_Reaction type . . . . . SJM-96

Persevere function

- Profile.Persevere . . . . . *SJM-124*

Pipe class . . . . . LM-14, SJM-12

Pipe object manager . . . . . SMU-11, SMU-58

placeholders, parameter . . . . . LM-7, LM-8, PM-127, PM-128, SJM-5, SJM-6, SMU-1, SMU-3, SMU-55

Plain constant

- Window\_Io.Plain . . . . . *DIO-157*

planar cursor movement . . . . . EI-2

pointer

- Hash.Ptr generic formal type . . . . . PT-41, PT-45

Pointer\_Level enumeration

- Debug.Numeric type . . . . . DEB-85

Pointer\_To\_Integer function

- Hash.Pointer\_To\_Integer . . . . . *PT-40*

Pointer\_To\_Integer generic function

- Hash.Pointer\_To\_Integer . . . . . *PT-39*

Pointer\_To\_Long\_Integer function

- Hash.Pointer\_To\_Long\_Integer . . . . . *PT-44*

Pointer\_To\_Long\_Integer generic function

- Hash.Pointer\_To\_Long\_Integer . . . . . *PT-43*

poke, *see* Modify

policy . . . . . PM-15

polymorphic	DIO-1
file	DIO-7
Polymorphic_Sequential_Io package	DIO-39
pop	TIO-8
Pop procedure	
Stack_Generic.Pop	PT-154
Pop_Error procedure	
Io.Pop_Error	TIO-68
Pop_Error renamed procedure	
Log.Pop_Error	SJM-44
Pop_Input procedure	
Io.Pop_Input	TIO-69
Pop_Input renamed procedure	
Log.Pop_Input	SJM-45
Pop_Output procedure	
Io.Pop_Output	TIO-70
Pop_Output renamed procedure	
Log.Pop_Output	SJM-46
port	
characteristics	
Terminal package	SMU-299
number	
Terminal.Current renamed function	SMU-303
settings	
Terminal.Settings procedure	SMU-326
<i>see also</i> terminal	
Port subtype	
System_Uilities.Port	SMU-250
Terminal.Port	SMU-305
position, <i>see</i> Column_Number, Index, Set_Index	
Position_Cursor procedure	
Window_Io.Position_Cursor	DIO-158
Insert procedure	DIO-138
Overwrite procedure	DIO-155
Position_Msg enumeration	
Profile.Msg_Kind type	SJM-118
positions	
Time_Uilities.Sun_Positions type	PT-202
Positive subtype	
Standard.Positive	PT-161
Positive_Count subtype	
Direct_Io.Positive_Count	DIO-23
Io.Positive_Count	TIO-7, TIO-71
Text_Io.Positive_Count	TIO-165, TIO-197
Window_Io.Positive_Count	DIO-160
Positive_Msg enumeration	
Profile.Msg_Kind type	SJM-118
pound sign (#)	
library wildcard	LM-8, LM-109, LM-113, LM-297, LM-301, PM-129, SJM-6, SMU-3
substitution character	LM-10, PM-130, SJM-7, SMU-4
symbol in window banner	PM-135

Pragma subclass . . . . .	LM-16, SJM-14
pragmas	
Disable_Deallocation	
Unchecked_Deallocation generic procedure . . . . .	PT-241
Unchecked_Deallocation.Unchecked_Deallocation procedure . . . . .	PT-247
Enable_Deallocation	
Unchecked_Deallocation generic procedure . . . . .	PT-241, PT-242
Unchecked_Deallocation.Unchecked_Deallocation procedure . . . . .	PT-247
Inline . . . . .	PM-116
Main . . . . .	PM-114
Nickname . . . . .	LM-15, SJM-13
Page_Limit	
System_Uilities.Get_Page_Counts procedure . . . . .	SMU-220
System_Uilities.Set_Page_Limit procedure . . . . .	SMU-261
Private_Eyes_Only . . . . .	PM-57, PM-89, PM-114
prefix	
Editor.Set.Argument_Prefix procedure . . . . .	EI-60
Profile.Default_Prefixes constant . . . . .	SJM-87
Profile.Log_Prefix type . . . . .	SJM-114
Profile.Log_Prefixes type . . . . .	SJM-116
Profile.No_Prefixes constant . . . . .	SJM-123
Profile.Prefixes function . . . . .	SJM-125
Profile.Set_Default_Prefixes procedure . . . . .	SJM-147
Profile.Set_Prefixes procedure . . . . .	SJM-158
Prefixes function	
Profile.Prefixes . . . . .	SJM-125
Log.Put_Line procedure . . . . .	SJM-49
pretty-print	
Common.Format procedure . . . . .	EST-88
Library.Reformat_Image procedure . . . . .	LM-255
prevent from running, <i>see</i> Hold	
previous	
Editor.Char.Delete_Previous procedure . . . . .	EI-13
[Previous Item] key	
Editor.Cursor.Previous procedure . . . . .	EI-19
Previous procedure	
Common.Object.Previous . . . . .	EST-117, PM-138
Ada images . . . . .	EST-4, EST-17
command images . . . . .	EST-51
Debugger . . . . .	DEB-6
Help . . . . .	EST-127
Library package . . . . .	LM-207
Links package . . . . .	LM-278
menu images . . . . .	EST-135
Search_List package . . . . .	SJM-212
session switches . . . . .	SJM-251
Switches package . . . . .	LM-317
text images . . . . .	EST-143
What package . . . . .	SJM-255
windows images . . . . .	EST-158
xref images . . . . .	EST-164
Editor.Cursor.Previous . . . . .	EI-3, EI-17, EI-19, PM-190
Ada.Show_Usage procedure . . . . .	EST-39
Editor.Hold_Stack.Previous . . . . .	EI-21, EI-22
Editor.Line.Previous . . . . .	EI-34
Editor.Mark.Previous . . . . .	EI-4, EI-41, EI-42
Editor.Screen.Previous . . . . .	EI-7, EI-51, EI-53

Previous procedure ( <i>continued</i> )	
Editor.Search.Previous . . . . .	EI-55, EI-56, EI-57
Editor.Window.Previous . . . . .	EI-64, EI-67
Editor.Word.Previous . . . . .	EI-71
[Previous Prompt] key	
Editor.Cursor.Previous procedure . . . . .	EI-19
[Previous Underline] key	
Editor.Cursor.Previous procedure . . . . .	EI-19
primary	
backup . . . . .	SMU-191
subsystem . . . . .	LM-90, PM-2, PM-16, PM-101, PM-339, PM-351, PM-354
copying view into secondary . . . . .	PM-103
setting up . . . . .	PM-103
Primary enumeration	
System_Backup.Kind type . . . . .	SMU-196
print	
default	
Queue.Default procedure . . . . .	SMU-102
display classes	
Queue.Classes procedure . . . . .	SJM-199
display devices	
Queue.Devices procedure . . . . .	SJM-200
display queue	
Queue.Display procedure . . . . .	SJM-201
operator capability . . . . .	SMU-92
queue . . . . .	SMU-92
Queue.Print procedure . . . . .	SJM-203
register	
Queue.Register procedure . . . . .	SMU-123
remove device from print spooler	
Queue.Remove procedure . . . . .	SMU-125
request . . . . .	SMU-91
session switches . . . . .	SJM-230
spooler . . . . .	SMU-91
Queue package . . . . .	SJM-197
Queue.Kill_Print_Spooler procedure . . . . .	SMU-112
Queue.Restart_Print_Spooler procedure . . . . .	SMU-127
stop	
Queue.Cancel procedure . . . . .	SJM-198, SMU-98
unregister	
Queue.Unregister procedure . . . . .	SMU-128
version	
Queue.Print_Version procedure . . . . .	SMU-118
view entries in queue	
Queue.Display procedure . . . . .	SMU-109
[Print] key	
Queue.Print procedure . . . . .	SMU-113
Print procedure	
Queue.Print . . . . .	SJM-203, SMU-91, SMU-113
Print_Version procedure	
Queue.Print_Version . . . . .	SMU-118
printing error messages	
Simple_Status.Display_Message function . . . . .	PT-134
printing status	
Simple_Status.Display_Message function . . . . .	PT-134



priority	
Daemon.Collection_Priority subtype	SMU-15
Daemon.Set_Priority procedure	SMU-38
Scheduler.Cpu_Priority subtype	SMU-141
Scheduler.Get_Cpu_Priority function	SMU-150
Priority function	
System_Uilities.Priority	SMU-251
Priority subtype	
System.Priority	PT-166
private part	PM-87, PM-89
Ada.Create_Private procedure	EST-24
closed	PM-87, PM-113
open	PM-89
Private_Eyes_Only pragma	PM-57, PM-89, PM-114
Privileged group	LM-20, SMU-54
privileged mode	SMU-54
Operator.Enable_Privileges procedure	SMU-72
Privileged_Mode function	
Operator.Privileged_Mode	SMU-81
Problem enumeration	
Daemon.Condition_Class type	SMU-16
Simple_Status.Condition_Class type	PT-130
Proc_Body subclass	LM-16, SJM-14
Proc_Inst subclass	LM-16, SJM-14
Proc_Ren subclass	LM-16, SJM-14
Proc_Spec subclass	LM-16, SJM-14
Procedure_Entry enumeration	
Debug.Stop_Event type	DEB-131
Process generic formal procedure	
Io.Process	TIO-107
Io.Wildcard_Iterator generic procedure	TIO-105
Io.Wildcard_Iterator procedure	TIO-108
processes	SMU-132
processors	LM-322
profile	PM-128, SJM-1, SJM-33, SJM-74, SMU-2
conversion	
Profile.Convert procedure	SJM-83
default	
Profile.Default_Profile function	SJM-88
default response	SJM-75
job response	SJM-75
response	
Profile.Response function	SJM-135
Profile.Response_Profile type	SJM-137
session response	SJM-75
session switches	SJM-229
set activity	
Profile.Set_Activity procedure	SJM-140
Profile package	SJM-4, SJM-33, SJM-73
<PROFILE> special value	DIO-6, LM-5, PM-128, SJM-3, SJM-33, SJM-75, SMU-2, SMU-55, TIO-6

program . . . . .	DEB-2
execution . . . . .	PM-12, PM-84
multiple subsystems . . . . .	PM-51
library . . . . .	PM-9
testing . . . . .	PM-85
Program package . . . . .	SJM-173
Program_Error exception	
Debug package . . . . .	DEB-57
Standard.Program_Error . . . . .	PT-161
programmatic breakpoint	
Debug_Tools.User_Break procedure . . . . .	DEB-185
progress	
Daemon.In_Progress function . . . . .	SMU-23
message . . . . .	SJM-3
<PROGRESS> special value . . . . .	SJM-76
project	
management	
defined . . . . .	PM-1
issues . . . . .	PM-4
partitioning . . . . .	PM-3
reporting . . . . .	PM-15
promote	
Ada.Code_Unit procedure . . . . .	EST-20
Ada.Install_Unit procedure . . . . .	EST-31
effort	
Compilation.Make renamed procedure . . . . .	LM-151
[Promote] key	
Common.Promote procedure . . . . .	EST-91
Promote procedure	
Common.Promote . . . . .	EST-59, EST-91, PM-192
Ada images . . . . .	EST-15
command images . . . . .	EST-49
Command.Spawn procedure . . . . .	EST-56
Library package . . . . .	LM-205
menu images . . . . .	EST-134
session switches . . . . .	SJM-249
Switches package . . . . .	LM-316
text images . . . . .	EST-141
windows images . . . . .	EST-157
xref images . . . . .	EST-163
Compilation.Promote . . . . .	LM-157
Demote procedure . . . . .	LM-141
Make renamed procedure . . . . .	LM-151
Editor.Window.Promote . . . . .	EI-63, EI-67
Demote procedure . . . . .	EI-65
Promote_Scope type	
Compilation.Promote_Scope . . . . .	LM-160
prompt . . . . .	DIO-87
next	
Editor.Cursor.Next procedure . . . . .	EI-19
previous	
Editor.Cursor.Previous procedure . . . . .	EI-19
Prompt enumeration	
Window_Io.Designation type . . . . .	DIO-116

Prompt field	
Window_Io package . . . . .	DIO-83
[Prompt For] key	
Editor.Key.Prompt procedure . . . . .	EI-29
Prompt procedure	
Editor.Key.Prompt . . . . .	EI-29
Prompt_Delimiters session switch . . . . .	SJM-243
Prompt_For_Account library switch . . . . .	LM-312
Prompt_For_Account session switch . . . . .	SJM-78, SJM-243
Prompt_For_Password library switch . . . . .	LM-312
Prompt_For_Password session switch . . . . .	SJM-243
propagate	
changes, <i>see</i> Merge	
request . . . . .	DEB-12
Propagate enumeration	
Profile.Error_Reaction type . . . . .	SJM-96
Propagate function	
Profile.Propagate . . . . .	SJM-126
[Propagate] key	
Debug.Propagate procedure . . . . .	DEB-96
Propagate procedure	
Debug.Propagate . . . . .	DEB-12, DEB-13, DEB-96
Catch procedure . . . . .	DEB-36
Context procedure . . . . .	DEB-44, DEB-45
Debug_Save_Exceptions session switch . . . . .	SJM-235
Option type . . . . .	DEB-88
Propagate_Exception enumeration	
Debug.Trace_Event type . . . . .	DEB-146
Protected enumeration	
Window_Io.Designation type . . . . .	DIO-116
Protected field	
Window_Io package . . . . .	DIO-83
protecting information, <i>see</i> access control	
Ps subclass . . . . .	LM-17, SJM-15
Ptr generic formal type	
Hash.Ptr . . . . .	PT-41, PT-45
Public group . . . . .	LM-20, SMU-54
Push procedure	
Editor.Hold_Stack.Push . . . . .	EI-5, EI-21, EI-22
Editor.Mark.Push . . . . .	EI-41, EI-42
Editor.Screen.Push . . . . .	EI-7, EI-51, EI-53
Stack_Generic.Push . . . . .	PT-155
put, <i>see</i> Write	
Put generic formal procedure	
Scheduler.Put . . . . .	SMU-186
Traverse_Job_Descriptors procedure . . . . .	SMU-187
[Put] key	
Debug.Put procedure . . . . .	DEB-100

Put procedure	
Debug.Put . . . . .	DEB-4, DEB-14, DEB-15, DEB-18, <i>DEB-100</i>
Context procedure . . . . .	DEB-44, DEB-45
Debug_Display_Level session switch . . . . .	SJM-232
Debug_Element_Count session switch . . . . .	SJM-232
Debug_First_Element session switch . . . . .	SJM-233
Debug_Pointer_Level session switch . . . . .	SJM-234
Debug_Put_Locals session switch . . . . .	SJM-234
Debug_Tools.Register generic procedure . . . . .	DEB-165
Flag procedure . . . . .	DEB-63
Numeric type . . . . .	DEB-84, DEB-85
Option type . . . . .	DEB-88
Io.Enumeration_Io.Put . . . . .	<i>TIO-115, TIO-117</i>
Io.Fixed_Io.Put . . . . .	<i>TIO-127, TIO-129</i>
Io.Float_Io.Put . . . . .	<i>TIO-139, TIO-141</i>
Io.Integer_Io.Put . . . . .	<i>TIO-150, TIO-152</i>
Io.Put . . . . .	<i>TIO-72, TIO-73, TIO-74, TIO-76, TIO-78</i>
Echo_Line procedure . . . . .	TIO-32
Put_Line procedure . . . . .	TIO-79
Text_Io.Enumeration_Io.Put . . . . .	<i>TIO-223, TIO-225</i>
Text_Io.Fixed_Io.Put . . . . .	<i>TIO-235, TIO-237</i>
Text_Io.Float_Io.Put . . . . .	<i>TIO-247, TIO-249</i>
Text_Io.Integer_Io.Put . . . . .	<i>TIO-258, TIO-260</i>
Text_Io.Put . . . . .	<i>TIO-198, TIO-199</i>
Put_Line procedure . . . . .	TIO-200
Put_Condition procedure	
Log.Put_Condition . . . . .	SJM-47
Program.Condition subtype . . . . .	SJM-177
Program.Create_Job procedure . . . . .	SJM-179
Put_Job_Messages procedure	
Log.Put_Job_Messages . . . . .	SJM-48
Put_Line procedure	
Io.Put_Line . . . . .	<i>TIO-79</i>
Note_Error generic formal procedure . . . . .	TIO-106
Log.Put_Line . . . . .	SJM-49
Text_Io.Put_Line . . . . .	<i>TIO-200</i>
Put_Locals enumeration	
Debug.Option type . . . . .	DEB-88
Put_Notes procedure	
Cmvc.Put_Notes . . . . .	PM-292
Cmvc.Append_Notes procedure . . . . .	PM-210
Cmvc.Create_Empty_Note_Window procedure . . . . .	PM-232
Cmvc.Get_Notes procedure . . . . .	PM-244
Put_System_Messages procedure	
Log.Put_System_Messages . . . . .	SJM-51
Profile.Error_Reaction type . . . . .	SJM-96, SJM-97

Q

qualified name, fully . . . . .	DEB-18, PM-131, SJM-8, SMU-5
Qualify_Stack_Names enumeration	
Debug.Option type . . . . .	DEB-88
quarter plane . . . . .	EI-2

question mark (?)	
file utilities wildcard . . . . .	LM-172, LM-181, LM-184, LM-187
library wildcard . . . . .	LM-8, LM-9, LM-109, LM-113, LM-297, LM-301, PM-129, SJM-6, SJM-7, SMU-3, SMU-4
metacharacter . . . . .	EI-56
substitution character . . . . .	LM-10, PM-130, SJM-8, SMU-5
question mark, double (??)	
library wildcard . . . . .	LM-8, LM-9, PM-129, SJM-6, SJM-7, SMU-3, SMU-4
Queue format (Debug.Memory_Display) . . . . .	DEB-79
Queue package . . . . .	SJM-197, SMU-91
Queue type	
Queue_Generic.Queue . . . . .	PT-109
Queue_Generic generic package . . . . .	PT-95
Queued enumeration	
Scheduler.Job_State type . . . . .	SMU-167
Queued state . . . . .	SMU-134
Quiesce procedure	
Daemon.Quiesce . . . . .	SMU-13, SMU-29
Quiet constant	
Profile.Quiet . . . . .	SJM-127
<QUIET> special value . . . . .	SJM-76
quiet startup . . . . .	DEB-16
Debug.Reset_Defaults procedure . . . . .	DEB-108
Quit enumeration	
Profile.Error_Reaction type . . . . .	SJM-96
Quit procedure	
Editor.Quit . . . . .	EI-10
Quote procedure	
Editor.Char.Quote . . . . .	EI-12, EI-14

R

raise	
Profile.Raise_Exception renamed function . . . . .	SJM-128
Raise_Error enumeration	
Profile.Error_Reaction type . . . . .	SJM-97
Raise_Exception renamed function	
Profile.Raise_Exception . . . . .	SJM-128
<RAISE_EXCEPTION> special value . . . . .	SJM-76
range	
System_Uilities.Character_Bits_Range subtype . . . . .	SMU-203
System_Uilities.Stop_Bits_Range subtype . . . . .	SMU-264
Terminal.Character_Bits_Range subtype . . . . .	SMU-302
Terminal.Stop_Bits_Range subtype . . . . .	SMU-327
type . . . . .	PT-9, PT-67, ST-25
Range_Type generic formal type	
Concurrent_Map_Generic.Range_Type . . . . .	PT-31
Map_Generic.Range_Type . . . . .	PT-89
String_Map_Generic.Range_Type . . . . .	ST-44

Rapid_Blink character attribute . . . . .	DIO-102
rate	
System_Uilities.Input_Rate function . . . . .	SMU-230
System_Uilities.Output_Rate function . . . . .	SMU-247
Terminal.Set_Input_Rate procedure . . . . .	SMU-312
Terminal.Set_Output_Rate procedure . . . . .	SMU-316
Raw package	
Window_Io.Raw . . . . .	DIO-171
reaction	
Profile.Default_Reaction constant . . . . .	SJM-90
Profile.Error_Reaction type . . . . .	SJM-96
Profile.Reaction function . . . . .	SJM-130
Profile.Set_Default_Reaction procedure . . . . .	SJM-148
Profile.Set_Reaction procedure . . . . .	SJM-159
Reaction function	
Profile.Reaction . . . . .	SJM-130
Reaction session switch . . . . .	SJM-243
read	
access	
file/Ada unit . . . . .	LM-21
world . . . . .	LM-21
files with different types of data	
Polymorphic_Sequential_Io package . . . . .	DIO-39
from tapes	
Tape package . . . . .	SMU-283
open to	
Io.In_File constant . . . . .	TIO-53
raw keystrokes typed by users	
Window_Io.Raw package . . . . .	DIO-171
see also Get, Get_Line	
Read constant	
Access_List.Read . . . . .	LM-43
Access_List_Tools.Read . . . . .	LM-80
Read procedure	
Direct_Io.Read . . . . .	DIO-24
Polymorphic_Sequential_Io.Operations.Read . . . . .	DIO-57
Sequential_Io.Read . . . . .	DIO-76
Tape.Read . . . . .	SMU-283, SMU-289
read-only	
access	
Direct_Io.File_Mode type . . . . .	DIO-15
Io.File_Mode subtype . . . . .	TIO-37
Polymorphic_Sequential_Io.File_Mode type . . . . .	DIO-46
Sequential_Io.File_Mode type . . . . .	DIO-68
Text_Io.File_Mode type . . . . .	TIO-178
Window_Io.File_Mode type . . . . .	DIO-120
lock . . . . .	EST-59
read/write	
access	
Direct_Io.File_Mode type . . . . .	DIO-15
to windows	
Window_Io package . . . . .	DIO-79
Read_Banner function	
Window_Io.Read_Banner . . . . .	DIO-161
Job_Number function . . . . .	DIO-141
Job_Time function . . . . .	DIO-142

Read_Mt procedure	
Tape.Read_Mt . . . . .	SMU-291
Read_Time enumeration	
Library.Field type . . . . .	LM-239
Reader enumeration	
Library.Field type . . . . .	LM-240
rebinding keys	
Editor.Key package . . . . .	EI-27
receive flow control . . . . .	SMU-318
Receive_Flow_Control function	
System_Uilities.Receive_Flow_Control . . . . .	SMU-252
Receive_Xon_Xoff_Bytes function	
System_Uilities.Receive_Xon_Xoff_Bytes . . . . .	SMU-254
Receive_Xon_Xoff_Characters function	
System_Uilities.Receive_Xon_Xoff_Characters . . . . .	SMU-255
reclaim, <i>see</i> Free	
storage, <i>see</i> Allows_Deallocation, Unchecked_Deallocation	
recombinant testing . . . . .	PM-85
recompile	
Compilation.Make renamed procedure . . . . .	LM-151
Compilation.Promote procedure . . . . .	LM-157
Recovery_Locality session switch . . . . .	SJM-244
Redirect procedure	
Text.Redirect . . . . .	EST-151
Redo procedure	
Common.Redo . . . . .	EST-47, EST-59, EST-93, PM-193
command images . . . . .	EST-49
Undo procedure . . . . .	EST-99
redraw error underlines, <i>see</i> Get_Errors	
Redraw procedure	
Editor.Screen.Redraw . . . . .	EI-51, EI-54
reduce library storage space	
Library.Compact_Library procedure . . . . .	LM-212
referencers . . . . .	PM-63
Reformat_Image procedure	
Library.Reformat_Image . . . . .	LM-255
refresh	
screen	
Editor.Screen.Redraw procedure . . . . .	EI-54
Window Directory . . . . .	EST-156
region . . . . .	EI-2
Region package	
Editor.Region . . . . .	EI-6, EI-45
<REGION> special name . . . . .	DEB-3, EST-58, EST-131, EST-155, LM-8, LM-130, LM-170, LM-199, LM-308, PM-128, SJM-5, SMU-2
register, stop	
Debug_Tools.Un_Register generic procedure . . . . .	DEB-181
Debug_Tools.Un_Register procedure . . . . .	DEB-183

Register generic procedure	
Debug_Tools.Register	DEB-165
Debug.Put procedure	DEB-101
Register procedure	
Debug_Tools.Register	DEB-175
Queue.Register	SMU-123
Add procedure	SMU-93
Remove procedure	SMU-125
Unregister procedure	SMU-128
relative	
cursor movement	EI-2, EI-3
pathname	DEB-90
release	PM-7, PM-21, PM-30, PM-187, PM-188, PM-268, PM-294
activity	PM-16, PM-187, PM-325
configuration	PM-31
copying in multihost development	PM-104
defined	PM-8
full view	PM-30
implications of upward-compatible changes	PM-91
integrating subpaths	PM-46
level number	PM-34, PM-230
names	PM-34
of configurations	PM-30
representation of	PM-32
<i>see also</i> Enable, Version	
Release procedure	
Cmvc.Release	PM-30, PM-46, PM-92, PM-194, PM-294, PM-327
Cmvc.Build procedure	PM-212
Common.Release	EST-59, EST-94, PM-136
Abandon procedure	EST-62
Ada images	EST-16
command images	EST-50
Debugger	DEB-5
Help	EST-126
Library package	LM-205
Links package	LM-277
menu images	EST-135
Search_List package	SJM-211
session switches	SJM-249
Switches package	LM-316
text images	EST-142
What package	SJM-254
windows images	EST-157
xref images	EST-163
Debug.Release	DEB-9, DEB-106
Context procedure	DEB-44
Execute procedure	DEB-61
Hold procedure	DEB-71
Xecute procedure	DEB-148
Search_List.Release	SJM-218
released view	PM-8, PM-30
relocation	PM-93
remaining disk capacity	
Operator.Disk_Space procedure	SMU-69
remote	
passwords	SJM-76
file	SJM-76, SJM-77



remote (continued)

passwords (continued)

Profile.Default_Remote_Passwords function . . . . .	SJM-91
Profile.Remote_Passwords function . . . . .	SJM-131
Profile.Remote_Passwords_Type subtype . . . . .	SJM-132
Profile.Set_Default_Remote_Passwords procedure . . . . .	SJM-149
Profile.Set_Remote_Passwords procedure . . . . .	SJM-160
sessions . . . . .	SJM-76, SJM-78
file . . . . .	SJM-78
Profile.Default_Remote_Sessions function . . . . .	SJM-92
Profile.Remote_Sessions function . . . . .	SJM-133
Profile.Remote_Sessions_Type subtype . . . . .	SJM-134
Profile.Set_Default_Remote_Sessions procedure . . . . .	SJM-150
Profile.Set_Remote_Sessions procedure . . . . .	SJM-161
Remote_Directory library switch . . . . .	LM-312
Remote_Directory session switch . . . . .	SJM-244
Remote_Machine library switch . . . . .	LM-313
Remote_Machine session switch . . . . .	SJM-244
Remote_Passwords function	
Profile.Remote_Passwords . . . . .	SJM-131
Remote_Passwords profile attribute . . . . .	SJM-73
Remote_Passwords session switch . . . . .	SJM-244
Remote_Passwords_Type subtype	
Profile.Remote_Passwords_Type . . . . .	SJM-132
Remote_Roof library switch . . . . .	LM-313
Remote_Roof session switch . . . . .	SJM-244
Remote_Sessions function	
Profile.Remote_Sessions . . . . .	SJM-133
Remote_Sessions profile attribute . . . . .	SJM-73
Remote_Sessions session switch . . . . .	SJM-78, SJM-244
Remote_Sessions_Type subtype	
Profile.Remote_Sessions_Type . . . . .	SJM-134
Remote_Type library switch . . . . .	LM-313
Remote_Type session switch . . . . .	SJM-244

remove

comment

Editor.Region.Uncomment procedure . . . . .	EI-49
---	-------

leading characters, *see* Strip, Strip\_Leading

map entry, *see* Undefine

part of, *see* Extract

spaces

Editor.Char.Delete_Spaces procedure . . . . .	EI-13
---	-------

stub

Ada.Withdraw procedure . . . . .	EST-43
----------------------------------	--------

trailing characters, *see* Strip, Strip\_Trailing

underlines

Common.Clear_Underlining procedure . . . . .	EST-64
--	--------

*see also* Cancel, Delete, Delete\_Group, Delete\_User, Pop

[Remove Breaks] key

Debug.Remove procedure . . . . .	DEB-107
----------------------------------	---------

Remove procedure	
Activity.Remove . . . . .	PM-158
Debug.Remove . . . . .	DEB-11, DEB-107
Queue.Remove . . . . .	SMU-125
Unregister procedure . . . . .	SMU-128
Remove_Child procedure	
Cmvc_Hierarchy.Remove_Child . . . . .	PM-336
Remove_From_Group procedure	
Operator.Remove_From_Group . . . . .	SMU-82
Remove_From_List procedure	
Work_Order.Remove_From_List . . . . .	PM-389
Remove_Import procedure	
Cmvc.Remove_Import . . . . .	PM-64, PM-299
Cmvc.Remove_Unused_Imports procedure . . . . .	PM-301
Remove_Unused_Imports procedure	
Cmvc.Remove_Unused_Imports . . . . .	PM-301
rename	
Ada units . . . . .	PM-36
view . . . . .	PM-50
Rename procedure	
Library.Rename . . . . .	LM-256
Rendezvous enumeration	
Debug.Information_Type type . . . . .	DEB-75
Debug.Trace_Event type . . . . .	DEB-146
[Rendezvous Info] key	
Debug.Information procedure . . . . .	DEB-73
repaint screen	
Editor.Screen.Redraw procedure . . . . .	EI-54
Repair_Cdb procedure	
Cmvc_Maintenance.Repair_Cdb . . . . .	PM-356
replace	
backward	
Editor.Search.Replace_Previous procedure . . . . .	EI-58
forward	
Editor.Search.Replace_Next procedure . . . . .	EI-57
window state . . . . .	EI-63
Replace procedure	
Bounded_String.Replace . . . . .	ST-17
Links.Replace . . . . .	LM-298
Link_Name subtype . . . . .	LM-297
Update procedure . . . . .	LM-302
Search_List.Replace . . . . .	SJM-219
Unbounded_String.Replace . . . . .	ST-120
Replace_Id procedure	
Ada.Replace_Id . . . . .	EST-37
Replace_Model procedure	
Cmvc.Replace_Model . . . . .	PM-23, PM-64, PM-96, PM-303
Replace_Next procedure	
Editor.Search.Replace_Next . . . . .	EI-55, EI-56, EI-57
Replace_Previous procedure	
Editor.Search.Replace_Previous . . . . .	EI-55, EI-56, EI-58

report writer	
Table_Formatter package	ST-91
Report_Cursor procedure	
Window_Io.Report_Cursor	DIO-163
Report_Location procedure	
Window_Io.Report_Location	DIO-164
Report_Origin procedure	
Window_Io.Report_Origin	DIO-165
Report_Size procedure	
Window_Io.Report_Size	DIO-166
Require_Comment_Lines enumeration	
Work_Order.Venture_Policy_Switch	PM-400
Require_Comments_At_Check_In enumeration	
Work_Order.Venture_Policy_Switch	PM-400
Require_Current_Work_Order enumeration	
Work_Order.Venture_Policy_Switch	PM-401
Require_Debug_Off enumeration	
Debug.Option type	DEB-88
Require_Internal_Links library switch	LM-313
reservation token	PM-6, PM-14, PM-25, PM-44, PM-226, PM-227, PM-264, PM-308, PM-317
reset	TIO-8
time	
Operator.Set_System_Time procedure	SMU-84
Reset procedure	
Direct_Io.Reset	DIO-25
Io.Reset	TIO-80
Polymorphic_Sequential_Io.Reset	DIO-53
Sequential_Io.Reset	DIO-77
Text_Io.Reset	TIO-201
Reset_Defaults procedure	
Debug.Reset_Defaults	DEB-16, DEB-108
Reset_Error	
Io.Exceptions.Use_Error exception	DIO-37, TIO-163
Reset_Error procedure	
Io.Reset_Error	TIO-81
Reset_Error renamed procedure	
Log.Reset_Error	SJM-52
Reset_Input procedure	
Io.Reset_Input	TIO-82
Reset_Input renamed procedure	
Log.Reset_Input	SJM-53
Reset_Log procedure	
Log.Reset_Log	SJM-54
Reset_Output procedure	
Io.Reset_Output	TIO-83
Reset_Output renamed procedure	
Log.Reset_Output	SJM-55
Reset_To_System_Default procedure	
Search_List.Reset_To_System_Default	SJM-220

resolution		
Profile.Get_Cached_Resolution procedure	.....	SJM-102
Resolve procedure		
Library.Resolve	.....	LM-258
resource limit	.....	SMU-197
response		
Profile.Set_Default_Response procedure	.....	SJM-151
Profile.Set_Response procedure	.....	SJM-162
Response function		
Profile.Response	.....	SJM-135
Response_Profile type		
Profile.Response_Profile	.....	SJM-137
rest		
List_Generic.Set_Rest procedure	.....	PT-65
Rest function		
List_Generic.Rest	.....	PT-63
Restart_Print_Spooler procedure		
Queue.Restart_Print_Spooler	.....	SMU-127
Kill_Print_Spooler procedure	.....	SMU-112
restore, <i>see</i> Archive package		
Restore procedure		
Archive.Restore	..... LM-23, LM-87, LM-88, LM-89, LM-90, LM-112, PM-103, PM-109	
Cmvc_Maintenance.Make_Primary procedure	.....	PM-351
Cmvc_Maintenance.Make_Secondary procedure	.....	PM-354
Editor.Macro.Restore	.....	EI-38
resume output		
Text.Continue procedure	.....	EST-147
Retain enumeration		
Library.Field type	.....	LM-240
Retargeted_Blocks constant		
System_Uilities.Retargeted_Blocks	.....	SMU-256
retention count	.....	EST-58, LM-197
default		
Library.Default_Keep_Versions constant	.....	LM-229
set		
Library.Set_Retention_Count procedure	.....	LM-259
Returned enumeration		
Debug.Stop_Event type	.....	DEB-131
Reverse_Locate function		
String_Uilities.Reverse_Locate	.....	ST-82
revert	.....	PM-7
Revert procedure		
Cmvc.Revert	.....	PM-28, PM-305
Cmvc.Accept_Changes procedure	.....	PM-207
Common.Revert	.....	EST-59, EST-96, PM-189
Ada images	.....	EST-16
command images	.....	EST-50
Library package	.....	LM-206
Links package	.....	LM-277
Redo procedure	.....	EST-49
Search_List package	.....	SJM-211

Revert procedure ( <i>continued</i> )	
Common.Revert ( <i>continued</i> )	
session switches . . . . .	SJM-249
Switches package . . . . .	LM-316
text images . . . . .	EST-142
What package . . . . .	SJM-254
windows images . . . . .	EST-157
Search_List.Revert . . . . .	SJM-221
Save procedure . . . . .	SJM-222
review intervals . . . . .	SMU-134
Rewind procedure	
Tape.Rewind . . . . .	SMU-292
right brace ( )	
file utilities wildcard . . . . .	LM-172, LM-181, LM-184, LM-187
metacharacter . . . . .	EI-56
Right enumeration	
Table_Formatter.Adjust type . . . . .	ST-93
Right procedure	
Editor.Cursor.Right . . . . .	EI-17, EI-19
Editor.Image.Right . . . . .	EI-25, EI-26
Window_Shift_Overlap session switch . . . . .	SJM-248
Editor.Screen.Right . . . . .	EI-54
root	
of library system . . . . .	DEB-18, LM-11, PM-131, SJM-8, SMU-5
task . . . . .	DEB-7
Rotate procedure	
Editor.Hold_Stack.Rotate . . . . .	EI-21, EI-22
Editor.Mark.Rotate . . . . .	EI-4, EI-41, EI-42
Editor.Screen.Rotate . . . . .	EI-7, EI-52, EI-54
rplaca, <i>see</i> Set_First	
rplacd, <i>see</i> Set_Rest	
run	
at scheduled time	
Daemon.Schedule procedure . . . . .	SMU-33
last	
Daemon.Last_Run function . . . . .	SMU-25
load . . . . .	SMU-135
queue load	
Scheduler.Get_Run_Queue_Load procedure . . . . .	SMU-157
Run enumeration	
Scheduler.Job_State type . . . . .	SMU-167
[Run] key	
Debug.Run procedure . . . . .	DEB-109
[Run Local] key	
Debug.Run(Local) procedure . . . . .	DEB-109
Run procedure	
Daemon.Run . . . . .	SMU-13, SMU-31
Debug.Run . . . . .	DEB-109
Clear_Stepping procedure . . . . .	DEB-42
Context procedure . . . . .	DEB-45
Hold procedure . . . . .	DEB-71
Show procedure . . . . .	DEB-119
Stop procedure . . . . .	DEB-128
Stop_Event type . . . . .	DEB-130

Run procedure ( <i>continued</i> )	
Program.Run . . . . .	SJM-187
Create_Job procedure . . . . .	SJM-181
Current function . . . . .	SJM-183, SJM-184
Run_Job procedure . . . . .	SJM-193
[Run Returned] key	
Debug.Run(Returned) procedure . . . . .	DEB-109
Run state . . . . .	SMU-133
Run_Job procedure	
Program.Run_Job . . . . .	LM-19, SJM-190
Create_Job procedure . . . . .	SJM-178, SJM-181
Current function . . . . .	SJM-183, SJM-184
Run procedure . . . . .	SJM-188
running	
task state . . . . .	DEB-9
<i>see also</i> In_Progress	
Running enumeration	
Debug.Task_Category type . . . . .	DEB-135

**S**

safe type . . . . .	DIO-4
Direct_Io package . . . . .	DIO-7
Polymorphic_Sequential_Io package . . . . .	DIO-39
Sequential_Io package . . . . .	DIO-61
same, <i>see</i> Equal	
Same_Directories constant	
Compilation.Same_Directories . . . . .	LM-162
Same_World constant	
Compilation.Same_World . . . . .	LM-163
Same_Worlds constant	
Compilation.Same_Worlds . . . . .	LM-164
save	
Common.Commit procedure . . . . .	EST-65
screen	
Editor.Screen.Push procedure . . . . .	EI-53
<i>see also</i> Archive package	
Save procedure	
Archive.Save . . . . .	LM-87, LM-88, LM-90, LM-122, PM-103, PM-109
Editor.Key.Save . . . . .	EI-29
Editor.Macro.Save . . . . .	EI-37, EI-39
Io.Save . . . . .	TIO-84
Log.Save . . . . .	SJM-56
Search_List.Save . . . . .	SJM-222
Revert procedure . . . . .	SJM-221
Save_Exceptions enumeration	
Debug.Option type . . . . .	DEB-88
saving images, <i>see</i> committing images	
Schedule procedure	
Daemon.Schedule . . . . .	SMU-33
Quiesce procedure . . . . .	SMU-29

scheduled	
Daemon.Next_Scheduled function . . . . .	SMU-28
Scheduler package . . . . .	SMU-131
scheduler parameters . . . . .	SMU-171
scheduling	
CPU . . . . .	SMU-134, SMU-171
disk . . . . .	SMU-140, SMU-172
memory . . . . .	SMU-139, SMU-171
scope	
Compilation.Promote_Scope type . . . . .	LM-160
screen	
control of . . . . .	DIO-91
copy to file	
Editor.Screen.Dump procedure . . . . .	EI-53
down	
Editor.Screen.Down procedure . . . . .	EI-53
editing . . . . .	DIO-83
erase and repaint	
Editor.Screen.Redraw procedure . . . . .	EI-54
erase contents	
Editor.Screen.Clear procedure . . . . .	EI-52
input/output . . . . .	DIO-3, TIO-3
left	
Editor.Screen.Left procedure . . . . .	EI-53
management . . . . .	EI-7
Editor.Screen package . . . . .	EI-51
move from bottom to top	
Editor.Screen.Rotate procedure . . . . .	EI-54
next	
Editor.Screen.Next procedure . . . . .	EI-53
previous	
Editor.Screen.Previous procedure . . . . .	EI-53
refresh	
Editor.Screen.Redraw procedure . . . . .	EI-54
retrieve from top of stack	
Editor.Screen.Top procedure . . . . .	EI-54
right	
Editor.Screen.Right procedure . . . . .	EI-54
save	
Editor.Screen.Push procedure . . . . .	EI-53
stack . . . . .	EI-7, EI-51
top	
Editor.Screen.Copy_Top procedure . . . . .	EI-52
Editor.Screen.Delete_Top procedure . . . . .	EI-52
Editor.Screen.Top procedure . . . . .	EI-54
transpose top two items	
Editor.Screen.Swap procedure . . . . .	EI-54
up	
Editor.Screen.Up procedure . . . . .	EI-54
Screen package	
Editor.Screen . . . . .	EI-7, EI-51
Screen_Dump_File session switch . . . . .	SJM-244
scroll	
bottom of image	
Editor.Image.End_Of procedure . . . . .	EI-26
bottom of window	
Editor.Window.End_Of procedure . . . . .	EI-66

scroll (continued)	
down	
Editor.Image.Down procedure . . . . .	EI-25
find image	
Editor.Image.Find procedure . . . . .	EI-26
left	
Editor.Image.Left procedure . . . . .	EI-26
right	
Editor.Image.Right procedure . . . . .	EI-26
top of image	
Editor.Image.Beginning_Of procedure . . . . .	EI-25
top of window	
Editor.Window.Beginning_Of procedure . . . . .	EI-64
up	
Editor.Image.Up procedure . . . . .	EI-26
search . . . . .	EI-4
and replace backward	
Editor.Search.Replace_Previous procedure . . . . .	EI-58
and replace forward	
Editor.Search.Replace_Next procedure . . . . .	EI-57
backward	
Editor.Search.Previous procedure . . . . .	EI-57
forward	
Editor.Search.Next procedure . . . . .	EI-57
strings	
String_Uilities package . . . . .	ST-69
<i>see also</i> Find, Locate	
Search package	
Editor.Search . . . . .	EI-4, EI-55
Search subclass . . . . .	LM-17, SJM-15
Search_Ignore_Case session switch . . . . .	SJM-245
Search_List package . . . . .	SJM-209
Search_Preserve_Case session switch . . . . .	SJM-245
Search_Regular_Expr session switch . . . . .	SJM-245
searchlist . . . . .	LM-6, SJM-1, SJM-4
access control . . . . .	LM-23
add component	
Search_List.Add procedure . . . . .	SJM-213
delete component	
Search_List.Delete procedure . . . . .	SJM-215
display	
Search_List.Display procedure . . . . .	SJM-216
display definition	
Search_List.Show_Item procedure . . . . .	SJM-224
display libraries	
Search_List.Display_Libraries procedure . . . . .	SJM-217
edit	
Search_List.Show_List procedure . . . . .	SJM-225
end editing	
Search_List.Release procedure . . . . .	SJM-218
images . . . . .	EST-1
name resolution mode . . . . .	LM-12, PM-132, SJM-10, SMU-7
replace components	
Search_List.Replace procedure . . . . .	SJM-219
replace searchlist	
Search_List.Revert procedure . . . . .	SJM-221
Search_List.Set_Up procedure . . . . .	SJM-223



searchlist ( <i>continued</i> )	
reset to default	
Search_List.Reset_To_System_Default procedure . . . . .	SJM-220
save	
Search_List.Save procedure . . . . .	SJM-222
secondary	
backup . . . . .	SMU-191
subsystem . . . . .	LM-90, PM-2, PM-16, PM-101, PM-339, PM-351, PM-354
setting up . . . . .	PM-103
Secondary enumeration	
System_Backup.Kind type . . . . .	SMU-196
seconds	
Time_Uilities.Milliseconds type . . . . .	PT-196
Seconds function	
Calendar.Seconds . . . . .	PT-6
Seconds type	
Time_Uilities.Seconds . . . . .	PT-201
security, <i>see</i> access control	
Seg_Listing library switch . . . . .	LM-313
selection	
Ada images . . . . .	EST-4
add comment	
Editor.Region.Comment procedure . . . . .	EI-46
beginning of current selection	
Editor.Region.Beginning_Of procedure . . . . .	EI-45
case conversion	
Editor.Region.Capitalize procedure . . . . .	EI-46
Editor.Region.Lower_Case procedure . . . . .	EI-48
Editor.Region.Upper_Case procedure . . . . .	EI-49
copy current selection	
Editor.Region.Copy procedure . . . . .	EI-46
delete current and copy at cursor	
Editor.Region.Move procedure . . . . .	EI-48
delete current selection	
Editor.Region.Delete procedure . . . . .	EI-46
end of current selection	
Editor.Region.End_Of procedure . . . . .	EI-46
fill	
Editor.Region.Fill procedure . . . . .	EI-47
getting help with . . . . .	EST-123
in Debugger window . . . . .	DEB-3
justify	
Editor.Region.Justify procedure . . . . .	EI-48
make comment	
Editor.Region.Comment procedure . . . . .	EI-46
mark end of	
Editor.Region.Finish procedure . . . . .	EI-47
mark start of	
Editor.Region.Start procedure . . . . .	EI-49
remove comment	
Editor.Region.Uncomment procedure . . . . .	EI-49
reselect	
Editor.Region.On procedure . . . . .	EI-49
unselect	
Editor.Region.Off procedure . . . . .	EI-49

<SELECTION> special name . . . . .	DEB-3, EST-58, EST-131, EST-155, LM-8, LM-130, LM-170, LM-199, LM-308, PM-128, SJM-5, SMU-2
semantic consistency . . . . .	PM-9
[Semanticize] key	
Common.Semanticize procedure . . . . .	EST-97
Semanticize procedure	
Common.Semanticize . . . . .	EST-97
Ada images . . . . .	EST-16
command images . . . . .	EST-50
xref images . . . . .	EST-163
semicolon (;)	
in set notation . . . . .	LM-13, PM-133, SJM-11, SMU-8
separator . . . . .	LM-18, PM-133, SJM-16, SMU-8
Send procedure	
Message.Send . . . . .	SMU-50
Send_All procedure	
Message.Send_All . . . . .	SMU-51
Send_Port_Enabled library switch . . . . .	LM-313
Send_Port_Enabled session switch . . . . .	SJM-245
separate	
Ada.Make_Separate procedure . . . . .	EST-35
separator, subitem	
Table_Formatter.Subitem_Separator generic formal object . . . . .	ST-102
Sequential_Io generic package . . . . .	DIO-61
Server enumeration	
Scheduler.Job_Kind type . . . . .	SMU-166
server job . . . . .	SMU-133
session . . . . .	SMU-197
create	
Operator.Create_Session procedure . . . . .	SMU-63
edit session switches	
Switches.Edit_Session_Attributes procedure . . . . .	LM-330
edit switch files . . . . .	SJM-227
error reactions . . . . .	SJM-3
get	
System_Uilities.Get_Session function . . . . .	SMU-222
identifier	
System_Uilities.Value function . . . . .	SMU-277
kill	
Operator.Force_Logoff procedure . . . . .	SJM-70
log files . . . . .	SJM-3
remote . . . . .	SJM-76
Profile.Default_Remote_Sessions function . . . . .	SJM-92
Profile.Remote_Sessions function . . . . .	SJM-133
Profile.Remote_Sessions_Type subtype . . . . .	SJM-134
Profile.Set_Default_Remote_Sessions procedure . . . . .	SJM-150
Profile.Set_Remote_Sessions procedure . . . . .	SJM-161
response profile . . . . .	DIO-6, LM-5, PM-128, SJM-3, SJM-75, SMU-2, SMU-55, TIO-6
searchlists . . . . .	SJM-4
stepping	
System_Uilities.Done function . . . . .	SMU-211
switch filename	
Switches.Of_Session constant . . . . .	LM-334

session ( <i>continued</i> )	
switches . . . . .	DEB-4, EI-7, EST-140, LM-5, LM-200, LM-308, PM-362, SJM-1, SJM-3, SJM-227, SJM-228
<i>see also</i> switches, session (for specific switch names)	
tailoring . . . . .	SJM-3
terminate	
Operator.Force_Logoff procedure . . . . .	SMU-75
Session class . . . . .	LM-14, SJM-12
Session function	
System_Uilities.Session . . . . .	SMU-257
Session object manager . . . . .	SMU-11, SMU-58
<SESSION> special value . . . . .	DIO-6, PM-128, SJM-76, SMU-2, SMU-55, TIO-6
Session_Id subtype	
System_Uilities.Session_Id . . . . .	SMU-258
Session_Iterator type	
System_Uilities.Session_Iterator . . . . .	SMU-259
Session_Name function	
System_Uilities.Session_Name . . . . .	SMU-260
<SESSION_PROFILE> special value . . . . .	LM-5, SJM-3, SJM-75
set	
break, <i>see</i> Break, User_Break	
break/exception, <i>see</i> Catch	
library context	
Library.Context procedure . . . . .	LM-214
mark	
Editor.Mark.Push procedure . . . . .	EI-42
notation . . . . .	LM-13, PM-133, SMU-8
position, <i>see</i> Set_Index	
trace, <i>see</i> Trace	
[Set Element Count] key	
Debug.Set_Value procedure . . . . .	DEB-114
[Set First Element] key	
Debug.Set_Value procedure . . . . .	DEB-114
Set package	
Editor.Set . . . . .	EI-6, EI-59
[Set Pointer Level] key	
Debug.Set_Value procedure . . . . .	DEB-114
Set procedure	
Access_List.Set . . . . .	LM-44
Access_List_Tools.Set . . . . .	LM-81
Activity.Set . . . . .	PM-159
Activity.Current procedure . . . . .	PM-147
Profile.Set . . . . .	SJM-138, SJM-139
Scheduler.Set . . . . .	SMU-131, SMU-133, SMU-170
Display procedure . . . . .	SMU-144
Get function . . . . .	SMU-149
Get_Wsl_Limits procedure . . . . .	SMU-159
Job_Descriptor type . . . . .	SMU-163
Job_Kind type . . . . .	SMU-165
Set_Wsl_Limits procedure . . . . .	SMU-180
Use_Default_Wsl_Limits procedure . . . . .	SMU-188
Switches.Set . . . . .	LM-335
Change procedure . . . . .	LM-321

Set type	
Set_Generic.Set . . . . .	PT-124
Initialize procedure . . . . .	PT-118
set/use information	
Ada.Show_Usage procedure . . . . .	EST-39
Common.Definition procedure . . . . .	EST-71
xref images . . . . .	EST-159
<i>see also</i> Definition	
Set_Access_List_Compaction procedure	
Daemon.Set_Access_List_Compaction . . . . .	SMU-35
Set_Activity procedure	
Profile.Set_Activity . . . . .	SJM-140
Set_Banner procedure	
Window_Io.Set_Banner . . . . .	DIO-168
Job_Number function . . . . .	DIO-141
Job_Time function . . . . .	DIO-142
Set_Character_Size procedure	
Terminal.Set_Character_Size . . . . .	SMU-306
System_Uilities.Character_Size function . . . . .	SMU-204
Set_Col procedure	
Io.Set_Col . . . . .	TIO-85
Text_Io.Set_Col . . . . .	TIO-203
Set_Consistency_Checking procedure	
Daemon.Set_Consistency_Checking . . . . .	SMU-36
Set_Default procedure	
Access_List.Set_Default . . . . .	LM-46
Access_List_Tools.Set_Default . . . . .	LM-83
Activity.Set_Default . . . . .	PM-67, PM-82, PM-161
Activity.Current procedure . . . . .	PM-147
Activity.Set procedure . . . . .	PM-159
Profile.Set_Default . . . . .	SJM-141
Get_Default function . . . . .	SJM-103, SJM-104
Work_Order.Set_Default . . . . .	PM-390
Set_Default_Activity procedure	
Profile.Set_Default_Activity . . . . .	SJM-142
Set_Default_Filter procedure	
Profile.Set_Default_Filter . . . . .	SJM-143, SJM-145
Set_Default_List procedure	
Work_Order.Set_Default_List . . . . .	PM-392
Work_Order.Venture_Editor.Set_Default_List . . . . .	PM-419
Set_Default_Log_File procedure	
Profile.Set_Default_Log_File . . . . .	SJM-146
Set_Default_Order procedure	
Work_Order.Venture_Editor.Set_Default_Order . . . . .	PM-421
Set_Default_Prefixes procedure	
Profile.Set_Default_Prefixes . . . . .	SJM-147
Set_Default_Reaction procedure	
Profile.Set_Default_Reaction . . . . .	SJM-148
Set_Default_Remote_Passwords procedure	
Profile.Set_Default_Remote_Passwords . . . . .	SJM-149
Set_Default_Remote_Sessions procedure	
Profile.Set_Default_Remote_Sessions . . . . .	SJM-150

Set_Default_Response procedure	
Profile.Set_Default_Response . . . . .	<i>SJM-151</i>
Set_Default_Venture procedure	
Work_Order.Set_Default_Venture . . . . .	<i>PM-394</i>
Set_Default_Width procedure	
Profile.Set_Default_Width . . . . .	<i>SJM-153</i>
Set_Detach_On_Disconnect procedure	
Terminal.Set_Detach_On_Disconnect . . . . .	<i>SMU-307</i>
Set_Disconnect_On_Disconnect procedure	
Terminal.Set_Disconnect_On_Disconnect . . . . .	<i>SMU-308</i>
Set_Disconnect_On_Failed_Login procedure	
Terminal.Set_Disconnect_On_Failed_Login . . . . .	<i>SMU-309</i>
Set_Disconnect_On_Logoff procedure	
Terminal.Set_Disconnect_On_Logoff . . . . .	<i>SMU-310</i>
Set_Error procedure	
Io.Set_Error . . . . .	<i>TIO-8, TIO-87</i>
Pop_Error procedure . . . . .	<i>TIO-68</i>
Reset_Error procedure . . . . .	<i>TIO-81</i>
Set_Error renamed procedure	
Log.Set_Error . . . . .	<i>SJM-57</i>
Pop_Error renamed procedure . . . . .	<i>SJM-44</i>
Reset_Error renamed procedure . . . . .	<i>SJM-52</i>
Set_Field procedure	
Work_Order.Editor.Set_Field . . . . .	<i>PM-409, PM-410, PM-411</i>
Set_Field_Info procedure	
Work_Order.Venture_Editor.Set_Field_Info . . . . .	<i>PM-423</i>
Set_Filter procedure	
Profile.Set_Filter . . . . .	<i>SJM-154, SJM-156</i>
Set_First procedure	
List_Generic.Set_First . . . . .	<i>PT-64</i>
Set_Flow_Control procedure	
Terminal.Set_Flow_Control . . . . .	<i>SMU-311</i>
System_Uilities.Flow_Control function . . . . .	<i>SMU-216</i>
Set_Generic generic package . . . . .	<i>PT-111</i>
Set_Index procedure	
Direct_Io.Set_Index . . . . .	<i>DIO-26</i>
Set_Input procedure	
Io.Set_Input . . . . .	<i>TIO-8, TIO-89</i>
Pop_Input procedure . . . . .	<i>TIO-69</i>
Reset_Input procedure . . . . .	<i>TIO-82</i>
Text_Io.Set_Input . . . . .	<i>TIO-205</i>
Set_Input renamed procedure	
Log.Set_Input . . . . .	<i>SJM-58</i>
Pop_Input renamed procedure . . . . .	<i>SJM-45</i>
Reset_Input renamed procedure . . . . .	<i>SJM-53</i>
Set_Input_Rate procedure	
Terminal.Set_Input_Rate . . . . .	<i>SMU-312</i>
System_Uilities.Input_Rate function . . . . .	<i>SMU-230</i>
Set_Job_Attribute procedure	
Scheduler.Set_Job_Attribute . . . . .	<i>SMU-133, SMU-179</i>
Get_Job_Attribute function . . . . .	<i>SMU-153</i>

Set_Length procedure	
Bounded_String.Set_Length . . . . .	ST-19
Unbounded_String.Set_Length . . . . .	ST-122
Set_Line procedure	
Io.Set_Line . . . . .	TIO-91
Text_Io.Set_Line . . . . .	TIO-206
Set_Line_Length procedure	
Io.Set_Line_Length . . . . .	TIO-93
Text_Io.Set_Line_Length . . . . .	TIO-208
Set_Load_View procedure	
Activity.Set_Load_View . . . . .	PM-162
Set_Log procedure	
Log.Set_Log . . . . .	SJM-59
Set_Log_Failed_Logins procedure	
Terminal.Set_Log_Failed_Logins . . . . .	SMU-313
Set_Log_File procedure	
Profile.Set_Log_File . . . . .	SJM-157
Set_Log_Threshold procedure	
Daemon.Set_Log_Threshold . . . . .	SMU-12, SMU-37
Condition_Class type . . . . .	SMU-16
Log_Threshold type . . . . .	SMU-26
Set_Login_Disabled procedure	
Terminal.Set_Login_Disabled . . . . .	SMU-314
Set_Logoff_On_Disconnect procedure	
Terminal.Set_Logoff_On_Disconnect . . . . .	SMU-315
Set_Notes procedure	
Work_Order.Editor.Set_Notes . . . . .	PM-412
Work_Order.List_Editor.Set_Notes . . . . .	PM-415
Work_Order.Set_Notes . . . . .	PM-395
Work_Order.Venture_Editor.Set_Notes . . . . .	PM-425
Set_Notes_List procedure	
Work_Order.Set_Notes_List . . . . .	PM-396
Set_Notes_Venture procedure	
Work_Order.Set_Notes_Venture . . . . .	PM-397
Set_Output procedure	
Io.Set_Output . . . . .	TIO-8, TIO-94
Pop_Output procedure . . . . .	TIO-70
Reset_Output procedure . . . . .	TIO-83
Text_Io.Set_Output . . . . .	TIO-209
Set_Output_renamed procedure	
Log.Set_Output . . . . .	SJM-60
Pop_Output_renamed procedure . . . . .	SJM-46
Reset_Log procedure . . . . .	SJM-54
Reset_Output_renamed procedure . . . . .	SJM-55
Set_Output_Rate procedure	
Terminal.Set_Output_Rate . . . . .	SMU-316
System_Uilities.Output_Rate function . . . . .	SMU-247
Set_Page_Length procedure	
Io.Set_Page_Length . . . . .	TIO-96
Text_Io.Set_Page_Length . . . . .	TIO-210

Set_Page_Limit procedure	
System_Utilities.Set_Page_Limit . . . . .	SMU-261
Get_Page_Counts procedure . . . . .	SMU-220
Set_Parity procedure	
Terminal.Set_Parity . . . . .	SMU-317
System_Utilities.Parity function . . . . .	SMU-248
System_Utilities.Parity_Kind type . . . . .	SMU-249
Set_Policy procedure	
Work_Order.Venture_Editor.Set_Policy . . . . .	PM-426
Set_Prefixes procedure	
Profile.Set_Prefixes . . . . .	SJM-158
Set_Priority procedure	
Daemon.Set_Priority . . . . .	SMU-38
Set_Reaction procedure	
Profile.Set_Reaction . . . . .	SJM-159
Set_Receive_Flow_Control procedure	
Terminal.Set_Receive_Flow_Control . . . . .	SMU-318
System_Utilities.Receive_Flow_Control function . . . . .	SMU-252
Set_Receive_Xon_Xoff_Bytes procedure	
Terminal.Set_Receive_Xon_Xoff_Bytes . . . . .	SMU-320
System_Utilities.Receive_Xon_Xoff_Bytes function . . . . .	SMU-254
Set_Receive_Xon_Xoff_Characters procedure	
Terminal.Set_Receive_Xon_Xoff_Characters . . . . .	SMU-321
System_Utilities.Receive_Xon_Xoff_Characters function . . . . .	SMU-255
Set_Remote_Passwords procedure	
Profile.Set_Remote_Passwords . . . . .	SJM-160
Set_Remote_Sessions procedure	
Profile.Set_Remote_Sessions . . . . .	SJM-161
Set_Response procedure	
Profile.Set_Response . . . . .	SJM-162
Errors constant . . . . .	SJM-95
Full constant . . . . .	SJM-99
Quiet constant . . . . .	SJM-127
Summary constant . . . . .	SJM-165
Terse constant . . . . .	SJM-166
Set_Rest procedure	
List_Generic.Set_Rest . . . . .	PT-65
Set_Retention_Count procedure	
Library.Set_Retention_Count . . . . .	LM-259
Set_Spec_View procedure	
Activity.Set_Spec_View . . . . .	PM-164
Set_Stop_Bits procedure	
Terminal.Set_Stop_Bits . . . . .	SMU-322
System_Utilities.Stop_Bits function . . . . .	SMU-263
Set_Subclass procedure	
Library.Set_Subclass . . . . .	LM-261
Set_System_Time procedure	
Operator.Set_System_Time . . . . .	SMU-84

Set_Task_Name procedure	
Debug.Set_Task_Name . . . . .	DEB-8, DEB-19, <i>DEB-112</i> , LM-11, SJM-9, SMU-6
Catch procedure . . . . .	DEB-39
Context procedure . . . . .	DEB-46
Debug_Tools.Set_Task_Name procedure . . . . .	DEB-179
Task_Display procedure . . . . .	DEB-136
Task_Name subtype . . . . .	DEB-140
Debug_Tools.Set_Task_Name . . . . .	DEB-8, <i>DEB-179</i> , LM-11, SJM-9, SMU-6
Debug.Catch procedure . . . . .	DEB-39
Debug.Context procedure . . . . .	DEB-46
Debug.Set_Task_Name procedure . . . . .	DEB-112
Debug.Task_Display procedure . . . . .	DEB-136
Debug.Task_Name subtype . . . . .	DEB-140
Get_Task_Name function . . . . .	DEB-161
Set_Terminal_Type procedure	
Terminal.Set_Terminal_Type . . . . .	SMU-323
System_Uilities.Terminal_Type function . . . . .	SMU-273
Set_Termination_Message procedure	
Job.Set_Termination_Message . . . . .	<i>SJM-31</i>
Set_Up procedure	
Search_List.Set_Up . . . . .	<i>SJM-223</i>
Set_Value procedure	
Debug.Set_Value . . . . .	DEB-16, <i>DEB-114</i>
Flag procedure . . . . .	DEB-63
Numeric type . . . . .	DEB-84
Set_Venture_Policy procedure	
Work_Order.Set_Venture_Policy . . . . .	<i>PM-398</i>
Set_Width procedure	
Profile.Set_Width . . . . .	<i>SJM-164</i>
Set_Wsl_Limits procedure	
Scheduler.Set_Wsl_Limits . . . . .	SMU-180
Get_Wsl_Limits procedure . . . . .	SMU-159
Job_Descriptor type . . . . .	SMU-163
Use_Default_Wsl_Limits procedure . . . . .	SMU-188
Set_Xon_Xoff_Bytes procedure	
Terminal.Set_Xon_Xoff_Bytes . . . . .	SMU-324
System_Uilities.Xon_Xoff_Bytes function . . . . .	SMU-280
Set_Xon_Xoff_Characters procedure	
Terminal.Set_Xon_Xoff_Characters . . . . .	SMU-325
System_Uilities.Xon_Xoff_Characters function . . . . .	SMU-281
sets, in names . . . . .	LM-7, SJM-5
setting, default	
Io.Enumeration_Io.Default_Setting constant . . . . .	TIO-110
Text_Io.Enumeration_Io.Default_Setting constant . . . . .	TIO-218
Settings procedure	
Terminal.Settings . . . . .	SMU-326
sever . . . . .	PM-14, PM-43, PM-308
Sever procedure	
Cmvc.Sever . . . . .	PM-43, PM-44, PM-48, <i>PM-308</i>
Cmvc.Copy procedure . . . . .	PM-222
Cmvc.Make_Path procedure . . . . .	PM-268



severed objects	
merging changes . . . . .	PM-45
rejoining . . . . .	PM-45
Severity function	
Simple_Status.Severity . . . . .	PT-142
Sharp_Msg enumeration	
Profile.Msg_Kind type . . . . .	SJM-118
Short enumeration	
Time_Uilities.Time_Format type . . . . .	PT-204
show	
break, <i>see</i> Information	
calls, <i>see</i> Stack	
defining occurrences	
Common.Definition procedure . . . . .	EST-71
symbol, <i>see</i> Display, Source	
task, <i>see</i> Task_Display	
<i>see also</i> Display, Display_Group, Display_Tape	
[Show Access List] key . . . . .	LM-38
[Show Breaks] key	
Debug.Show procedure . . . . .	DEB-115
[Show Errors] key	
Ada.Get_Errors procedure . . . . .	EST-28
[Show Exceptions] key	
Debug.Show(Exceptions) procedure . . . . .	DEB-115
Show procedure	
Cmvc.Show . . . . .	PM-40, PM-310
Cmvc.Check_Out procedure . . . . .	PM-218
Debug.Show . . . . .	DEB-115
State_Type type . . . . .	DEB-126
[Show Source] key	
Debug.Source procedure . . . . .	DEB-121
[Show Unused (Unit)] key	
Ada.Show_Unused procedure . . . . .	EST-41
[Show Unused] key	
Ada.Show_Unused procedure . . . . .	EST-41
[Show Usage (Indirect)] key	
Ada.Show_Usage procedure . . . . .	EST-39
[Show Usage (Unit)] key	
Ada.Show_Usage procedure . . . . .	EST-39
[Show Usage] key	
Ada.Show_Usage procedure . . . . .	EST-39
Show_All_Checked_Out procedure	
Cmvc.Show_All_Checked_Out . . . . .	PM-312
Cmvc.Check_Out procedure . . . . .	PM-218
Show_All_Controlled procedure	
Cmvc.Show_All_Controlled . . . . .	PM-313
Show_All_Uncontrolled procedure	
Cmvc.Show_All_Uncontrolled . . . . .	PM-314
Cmvc.Check_Out procedure . . . . .	PM-218

Show_Checked_Out_By_User procedure	
Cmvc.Show_Checked_Out_By_User . . . . .	PM-315
Cmvc.Check_Out procedure . . . . .	PM-218
Show_Checked_Out_In_View procedure	
Cmvc.Show_Checked_Out_In_View . . . . .	PM-316
Cmvc.Check_Out procedure . . . . .	PM-218
Show_History procedure	
Cmvc.Show_History . . . . .	PM-317
Show_History_By_Generation procedure	
Cmvc.Show_History_By_Generation . . . . .	PM-29, PM-319
Cmvc.Check_In procedure . . . . .	PM-216
Cmvc.Merge_Changes procedure . . . . .	PM-289
Show_Image_Of_Generation procedure	
Cmvc.Show_Image_Of_Generation . . . . .	PM-321
Show_Item procedure	
Search_List.Show_Item . . . . .	SJM-224
Show_List procedure	
Search_List.Show_List . . . . .	SJM-225
Search_List package . . . . .	SJM-210
Show_Location enumeration	
Debug.Option type . . . . .	DEB-88
Show_Log_Thresholds procedure	
Daemon.Show_Log_Thresholds . . . . .	SMU-39
Show_Login_Limit procedure	
Operator.Show_Login_Limit . . . . .	SMU-86
Show_Out_Of_Date_Objects procedure	
Cmvc.Show_Out_Of_Date_Objects . . . . .	PM-323
Cmvc.Accept_Changes procedure . . . . .	PM-205
Show_Shutdown_Settings procedure	
Operator.Show_Shutdown_Settings . . . . .	SMU-87
Show_Snapshot_Settings procedure	
Daemon.Show_Snapshot_Settings . . . . .	SMU-40
Show_Unused procedure	
Ada.Show_Unused . . . . .	EST-41
Show_Usage procedure	
Ada.Show_Usage . . . . .	EST-39
shutdown	
Operator.Archive_On_Shutdown procedure . . . . .	SMU-58
Operator.Cancel_Shutdown procedure . . . . .	SMU-59
Operator.Explain_Crash procedure . . . . .	SMU-74
Operator.Get_Archive_On_Shutdown function . . . . .	SMU-76
Operator.Get_Shutdown_Interval function . . . . .	SMU-78
Operator.Show_Shutdown_Settings procedure . . . . .	SMU-87
Shutdown procedure	
Operator.Shutdown . . . . .	SMU-88
Cancel_Shutdown procedure . . . . .	SMU-59
Explain_Crash procedure . . . . .	SMU-74
Get_Shutdown_Interval function . . . . .	SMU-78
Shutdown_Warning procedure . . . . .	SMU-90
Shutdown_Warning procedure	
Operator.Shutdown_Warning . . . . .	SMU-90
Get_Shutdown_Interval function . . . . .	SMU-78

Shutdown_Warning procedure ( <i>continued</i> )	
Operator.Shutdown_Warning ( <i>continued</i> )	
Show_Shutdown_Settings procedure . . . . .	SMU-87
Shutdown procedure . . . . .	SMU-88
sibling	
Common.Object.Next procedure . . . . .	EST-113
Simple_Key subtype	
Window_Io.Raw.Simple_Key . . . . .	DIO-184
Simple_Name subtype	
Library.Simple_Name . . . . .	LM-262
Simple_Status package . . . . .	
	PT-127
single-library application . . . . .	
	PM-15
Single_Unit enumeration	
Compilation.Promote_Scope type . . . . .	LM-160
size	
byte	
System.Byte_Size constant . . . . .	PT-164
character	
System_Uilities.Character_Size function . . . . .	SMU-204
Terminal.Set_Character_Size procedure . . . . .	SMU-306
get	
Daemon.Get_Size procedure . . . . .	SMU-20
memory	
System.Memory_Size constant . . . . .	PT-165
report	
Window_Io.Report_Size procedure . . . . .	DIO-166
word	
System.Word_Size constant . . . . .	PT-167
working set	
Scheduler.Working_Set_Size function . . . . .	SMU-189
Size enumeration	
Library.Field type . . . . .	LM-240
Size function	
Direct_Io.Size . . . . .	DIO-27
Size generic formal object	
Concurrent_Map_Generic.Size . . . . .	PT-32
Map_Generic.Size . . . . .	PT-90
String_Map_Generic.Size . . . . .	ST-45
Skip_Line procedure	
Io.Skip_Line . . . . .	TIO-97
Get_Line procedure . . . . .	TIO-51
Set_Line procedure . . . . .	TIO-91
Text_Io.Skip_Line . . . . .	TIO-211
Get_Line procedure . . . . .	TIO-183
Set_Line procedure . . . . .	TIO-206
Skip_Page procedure	
Io.Skip_Page . . . . .	TIO-98
Text_Io.Skip_Page . . . . .	TIO-212
Slow_Blink character attribute . . . . .	
	DIO-102
snapshot . . . . .	
	SMU-12
settings	
Daemon.Get_Snapshot_Settings procedure . . . . .	SMU-21
Daemon.Show_Snapshot_Settings procedure . . . . .	SMU-40

Snapshot client . . . . .	SMU-12, SMU-13, SMU-27, SMU-31
Snapshot_Finish_Message procedure	
Daemon.Snapshot_Finish_Message . . . . .	SMU-41
Snapshot_Start_Message procedure	
Daemon.Snapshot_Start_Message . . . . .	SMU-42
Snapshot_Warning_Message procedure	
Daemon.Snapshot_Warning_Message . . . . .	SMU-43
software flow control . . . . .	SMU-311
sort format . . . . .	PM-136
Sort procedure	
Table_Formatter.Sort . . . . .	ST-100
Sort_Image procedure	
Common.Sort_Image . . . . .	EST-98, PM-136
Links package . . . . .	LM-277
sorting	
table	
Table_Formatter.Field_List type . . . . .	ST-95
Table_Sort_Generic generic procedure . . . . .	PT-169
Table_Sort_Generic.Table_Sort_Generic procedure . . . . .	PT-174
[Source (All Worlds)] key	
Compilation.Demote procedure . . . . .	LM-141
[Source (This World)] key	
Compilation.Demote procedure . . . . .	LM-141
source	
configuration . . . . .	PM-9
display . . . . .	DEB-3
objects	
Compilation.Demote procedure . . . . .	LM-141
Compilation.Make renamed procedure . . . . .	LM-151
Compilation.Promote procedure . . . . .	LM-157
unit state . . . . .	EST-5
Source enumeration	
Compilation.Unit_State type . . . . .	LM-167
Source generic formal type	
Unchecked_Conversion.Source . . . . .	PT-210
Unchecked_Conversions.Source . . . . .	PT-240
Unchecked_Conversions.Unchecked_Conversion_Package.Source . . . . .	PT-229
Source procedure	
Debug.Source . . . . .	DEB-121
[Source Unit] key	
Ada.Source_Unit procedure . . . . .	EST-42
Source_Name subtype	
Links.Source_Name . . . . .	LM-275, LM-300
Source_Pattern subtype	
Links.Source_Pattern . . . . .	LM-276, LM-301
Source_Unit procedure	
Ada.Source_Unit . . . . .	EST-42
space	
Editor.Char.Delete_Spaces procedure . . . . .	EI-13
Operator.Disk_Space procedure . . . . .	SJM-67, SMU-69

Space enumeration	
Debug.Information_Type type	DEB-75
Space procedure	
Library.Space	LM-263
spawn job, <i>see</i> Create_Job, Run_Job	
Spawn procedure	
Command.Spawn	EST-56
Scheduler.Job_Kind type	SMU-165
spec view	PM-10, PM-11, PM-29, PM-52, PM-136, PM-137, PM-187
adding or removing units from	PM-95
compilation	PM-61
controlled units	PM-61
creating	PM-58
names and level numbers	PM-59
spec/load subsystems	PM-187
Spec_Load subclass	LM-15, SJM-13
Spec_Load_Subsystem enumeration	
Cmvc.System_Object_Enum type	PM-324
Spec_View subclass	LM-15, SJM-13
special characters	DEB-18, LM-7, LM-10, PM-131, SJM-5, SJM-8, SMU-5
backslash (\)	DEB-20, LM-10, LM-12, PM-132, SJM-8, SJM-10, SJM-210, SMU-7
braces ({} )	LM-13, PM-133, SJM-11, SMU-8
brackets ([ ])	LM-13, LM-109, LM-113, LM-297, LM-301, PM-133, SJM-11, SMU-8
caret (^)	DEB-18, LM-10, LM-11, PM-131, SJM-8, SMU-6
dollar sign (\$)	DEB-18, LM-10, LM-11, PM-131, SJM-8, SJM-9, SJM-209, SMU-6
double dollar sign (\$\$)	DEB-18, DEB-19, LM-10, LM-11, PM-132, SJM-8, SJM-9, SMU-6
exclamation mark (!)	DEB-18, LM-10, LM-11, PM-131, SJM-8, SMU-5
grave (`)	DEB-20, LM-10, LM-12, PM-132, SJM-4, SJM-8, SJM-10, SJM-209, SMU-7
percent (%)	DEB-18, DEB-19, LM-10, LM-11, SJM-8, SJM-9, SMU-6
period (.)	DEB-18, DEB-19, LM-10, LM-12, PM-132, SJM-8, SJM-10, SMU-7
underscore (_)	DEB-18, DEB-19, LM-10, LM-12, PM-132, SJM-8, SJM-9, SMU-7
special characters, insert	
Editor.Char.Quote procedure	EI-12, EI-14
special display	
Debug.Put procedure	DEB-101
Debug_Tools.Register generic procedure	DEB-165
Debug_Tools.Register procedure	DEB-175
special groups	SMU-54
special names	DEB-57, DEB-90, DIO-5, EST-58, LM-7, PM-127, SJM-5, SJM-34, SMU-1, SMU-2, TIO-4
<ACTIVITY>	EST-58, EST-131, EST-155, LM-8, LM-130, LM-170, LM-199, LM-308, PM-128, SJM-5, SMU-2
Ada images	EST-5
<CURSOR>	DEB-3, EST-58, EST-131, EST-155, LM-8, LM-130, LM-170, LM-199, LM-308, PM-128, SJM-5, SMU-2
<IMAGE>	DEB-3, EST-58, EST-131, EST-155, LM-8, LM-130, LM-170, LM-199, LM-308, PM-128, SJM-5, SMU-2
menu images	EST-131
online help facility	EST-123
<REGION>	DEB-3, EST-58, EST-131, EST-155, LM-8, LM-130, LM-170, LM-199, LM-308, PM-128, SJM-5, SMU-2
<SELECTION>	DEB-3, EST-58, EST-131, EST-155, LM-8, LM-130, LM-170, LM-199, LM-308, PM-128, SJM-5, SMU-2
<TEXT>	EST-58, EST-131, EST-155, LM-8, LM-130, LM-170, LM-199, LM-308, PM-128, SJM-5, SMU-2

special values	LM-5, PM-128, SJM-6, SMU-2
<ALL_WORLDS>	LM-129, LM-134, LM-199
<DEFAULT>	DIO-6, LM-5, PM-128, SJM-3, SJM-75, SMU-55, TIO-6
<DIRECTORIES>	LM-129, LM-134, LM-138, LM-162, LM-199
<ERRORS>	SJM-75
<IGNORE>	SJM-75
<NIL>	SJM-75
<PROFILE>	DIO-6, LM-5, PM-128, SJM-3, SJM-33, SJM-75, SMU-55, TIO-6
<PROGRESS>	SJM-76
<QUIET>	SJM-76
<RAISE_EXCEPTION>	SJM-76
<SESSION>	DIO-6, PM-128, SJM-76, SMU-55, TIO-6
<SESSION_PROFILE>	LM-5, SJM-3, SJM-75
<SUBUNITS>	LM-129, LM-134, LM-199
<UNITS>	LM-129, LM-134, LM-199
<VERBOSE>	SJM-76
<WARN>	SJM-76
<WARNINGS>	SJM-76
<WORLDS>	LM-129, LM-134, LM-163, LM-164, LM-199
Special_Types enumeration	
Debug.State_Type type	DEB-126
Specification subtype	
Switches.Specification	LM-336
Value_Image subtype	LM-337
Split procedure	
Calendar.Split	PT-7
spooler	
devices	
Queue.All_Spooler_Devices constant	SMU-97
print	
Queue.Kill_Print_Spooler procedure	SMU-112
Queue.Restart_Print_Spooler procedure	SMU-127
Queue package	SJM-197
Spread_Fields procedure	
Work_Order.Venture_Editor.Spread_Fields	PM-427
stable-storage error log	SMU-12
stack	TIO-8
file	
Io.Pop_Error procedure	TIO-68
Io.Pop_Input procedure	TIO-69
Io.Pop_Output procedure	TIO-70
Io.Reset_Error procedure	TIO-81
Io.Reset_Input procedure	TIO-82
Io.Reset_Output procedure	TIO-83
Io.Set_Error procedure	TIO-87
Io.Set_Input procedure	TIO-89
Io.Set_Output procedure	TIO-94
frame	DEB-8
prefix	SMU-7
Stack_Generic.Empty_Stack constant	PT-148
[Stack] key	
Debug.Stack procedure	DEB-123
Stack procedure	
Debug.Stack	DEB-4, DEB-7, DEB-8, DEB-123
Context procedure	DEB-45
Debug_Addresses session switcl.	SJM-232

Stack procedure ( <i>continued</i> )	
Debug.Stack ( <i>continued</i> )	
Debug_Qualify_Stack_Names session switch . . . . .	SJM-234
Debug_Stack_Count session switch . . . . .	SJM-235
Numeric type . . . . .	DEB-85
Option type . . . . .	DEB-86, DEB-88
Stack type	
Stack_Generic.Stack . . . . .	PT-156
Stack_Count enumeration	
Debug.Numeric type . . . . .	DEB-85
Stack_Generic generic package . . . . .	
	PT-143
Stack_Start enumeration	
Debug.Numeric type . . . . .	DEB-85
standard	
error file . . . . .	DIO-3, TIO-3, TIO-7
System_Uilities.Error_Name function . . . . .	SMU-215
input file . . . . .	DIO-3, TIO-3, TIO-7, TIO-165
System_Uilities.Input_Name function . . . . .	SMU-228
output file . . . . .	DIO-3, TIO-3, TIO-7, TIO-165
System_Uilities.Output_Name function . . . . .	SMU-246
value . . . . .	DEB-84, DEB-86
Standard package . . . . .	
	PT-161
Standard_Error function	
Io.Standard_Error . . . . .	TIO-99
Standard_Input function	
Io.Standard_Input . . . . .	TIO-100
Text_Io.Standard_Input . . . . .	TIO-213
Standard_Output function	
Io.Standard_Output . . . . .	TIO-101
Text_Io.Standard_Output . . . . .	TIO-214
start	
at scheduled time	
Daemon.Schedule procedure . . . . .	SMU-33
generator	
Ada.Create_Body procedure . . . . .	EST-22
<i>see also</i> Run	
Start procedure	
Editor.Macro.Start . . . . .	EI-7, EI-37, EI-39
Editor.Region.Start . . . . .	EI-45, EI-49
Started_Successfully function	
Program.Started_Successfully . . . . .	SJM-195
Condition subtype . . . . .	SJM-177
Create_Job procedure . . . . .	SJM-178
state . . . . .	
	DEB-1
attribute . . . . .	LM-17, SJM-13
description directory . . . . .	PM-32
job . . . . .	SMU-133
Scheduler.Get_Job_State function . . . . .	SMU-156
Scheduler.Job_State type . . . . .	SMU-167
presently unknown execution message (Debug.Task_Display) . . . . .	DEB-137
unit	
Compilation.Unit_State type . . . . .	LM-166
State procedure	
Scheduler.State . . . . .	SMU-131, SMU-182
Job_Descriptor type . . . . .	SMU-161

State_Type type	
Debug.State_Type . . . . .	DEB-126
Statement enumeration	
Debug.Stop_Event type . . . . .	DEB-131
Debug.Trace_Event type . . . . .	DEB-146
Statement subclass . . . . .	LM-16, SJM-14
Statement_Indentation library switch . . . . .	LM-313
Statement_Length library switch . . . . .	LM-313
status	
handling	
Simple_Status package . . . . .	PT-127
Program.Condition subtype . . . . .	SJM-177
Status enumeration	
Library.Field type . . . . .	LM-240
Status procedure	
Daemon.Status . . . . .	SMU-13, SMU-44
Major_Clients constant . . . . .	SMU-27
Run procedure . . . . .	SMU-31
Status type	
Check.Status . . . . .	PM-179
Status_Error exception	
Direct_Io generic package	
Close procedure . . . . .	DIO-8
Create procedure . . . . .	DIO-11
Delete procedure . . . . .	DIO-12
Form function . . . . .	DIO-17
Mode function . . . . .	DIO-20
Name function . . . . .	DIO-21
Open procedure . . . . .	DIO-22
Reset procedure . . . . .	DIO-25
Io package	
Append procedure . . . . .	TIO-13
Close procedure . . . . .	TIO-14
Col function . . . . .	TIO-15
Create procedure . . . . .	TIO-21
Delete procedure . . . . .	TIO-25
End_Of_File function . . . . .	TIO-33
End_Of_Line function . . . . .	TIO-34
End_Of_Page function . . . . .	TIO-35
Form function . . . . .	TIO-40
Get procedure . . . . .	TIO-41, TIO-42, TIO-44, TIO-46, TIO-48, TIO-49
Get_Line procedure . . . . .	TIO-50, TIO-52
Line function . . . . .	TIO-55
Line_Length function . . . . .	TIO-56
Mode function . . . . .	TIO-58
Name function . . . . .	TIO-59
New_Line procedure . . . . .	TIO-60
New_Page procedure . . . . .	TIO-61
Open procedure . . . . .	TIO-64
Page function . . . . .	TIO-66
Page_Length function . . . . .	TIO-67
Put procedure . . . . .	TIO-72, TIO-73, TIO-75, TIO-77, TIO-78
Put_Line procedure . . . . .	TIO-79
Reset procedure . . . . .	TIO-80
Set_Col procedure . . . . .	TIO-86
Set_Error procedure . . . . .	TIO-88



Status\_Error exception (continued)

Io package (continued)

Set_Input procedure . . . . .	TIO-90
Set_Line procedure . . . . .	TIO-92
Set_Line_Length procedure . . . . .	TIO-93
Set_Output procedure . . . . .	TIO-95
Set_Page_Length procedure . . . . .	TIO-96
Skip_Line procedure . . . . .	TIO-97
Skip_Page procedure . . . . .	TIO-98
Io.Enumeration_Io generic package	
Get procedure . . . . .	TIO-113
Put procedure . . . . .	TIO-116
Io.Fixed_Io generic package	
Get procedure . . . . .	TIO-124
Put procedure . . . . .	TIO-128
Io.Float_Io generic package	
Get procedure . . . . .	TIO-136
Put procedure . . . . .	TIO-140
Io.Integer_Io generic package	
Get procedure . . . . .	TIO-147
Put procedure . . . . .	TIO-151
Io.Exceptions.Status_Error . . . . .	DIO-36, TIO-162
Polymorphic_Sequential_Io package	
Append procedure . . . . .	DIO-40
Close procedure . . . . .	DIO-41
Create procedure . . . . .	DIO-43
Delete procedure . . . . .	DIO-44
End_Of_File function . . . . .	DIO-45
Form function . . . . .	DIO-48
Mode function . . . . .	DIO-50
Name function . . . . .	DIO-51
Open procedure . . . . .	DIO-52
Reset procedure . . . . .	DIO-53
Polymorphic_Sequential_Io.Operations package	
Read procedure . . . . .	DIO-57
Sequential_Io package	
Close procedure . . . . .	DIO-62
Create procedure . . . . .	DIO-64
Delete procedure . . . . .	DIO-65
End_Of_File function . . . . .	DIO-67
Form function . . . . .	DIO-70
Mode function . . . . .	DIO-72
Name function . . . . .	DIO-73
Open procedure . . . . .	DIO-75
Read procedure . . . . .	DIO-76
Reset procedure . . . . .	DIO-77
Text_Io package	
Close procedure . . . . .	TIO-166
Col function . . . . .	TIO-167
Create procedure . . . . .	TIO-170
Delete procedure . . . . .	TIO-173
End_Of_File function . . . . .	TIO-174
End_Of_Line function . . . . .	TIO-175
End_Of_Page function . . . . .	TIO-176
Get procedure . . . . .	TIO-181, TIO-182
Get_Line procedure . . . . .	TIO-184
Line function . . . . .	TIO-186
Line_Length function . . . . .	TIO-187
Mode function . . . . .	TIO-188
Name function . . . . .	TIO-189
New_Line procedure . . . . .	TIO-190
New_Page procedure . . . . .	TIO-191

Status\_Error exception (continued)

Text\_Io package (continued)

Open procedure . . . . .	TIO-194
Page function . . . . .	TIO-195
Page_Length function . . . . .	TIO-196
Put procedure . . . . .	TIO-198, TIO-199
Put_Line procedure . . . . .	TIO-200
Reset procedure . . . . .	TIO-202
Set_Col procedure . . . . .	TIO-204
Set_Input procedure . . . . .	TIO-205
Set_Line procedure . . . . .	TIO-207
Set_Line_Length procedure . . . . .	TIO-208
Set_Output procedure . . . . .	TIO-209
Set_Page_Length procedure . . . . .	TIO-210
Skip_Line procedure . . . . .	TIO-211
Skip_Page procedure . . . . .	TIO-212

Text\_Io.Enumeration\_Io generic package

Get procedure . . . . .	TIO-221
Put procedure . . . . .	TIO-224

Text\_Io.Fixed\_Io generic package

Get procedure . . . . .	TIO-232
Put procedure . . . . .	TIO-236

Text\_Io.Float\_Io generic package

Get procedure . . . . .	TIO-244
Put procedure . . . . .	TIO-248

Text\_Io.Integer\_Io generic package

Get procedure . . . . .	TIO-255
Put procedure . . . . .	TIO-259

Window\_Io package

Char_At function . . . . .	DIO-106
Close procedure . . . . .	DIO-107
Create procedure . . . . .	DIO-110
Delete procedure . . . . .	DIO-113, DIO-114
Delete_Lines procedure . . . . .	DIO-115
End_Of_File function . . . . .	DIO-118
End_Of_Line function . . . . .	DIO-119
Font_At function . . . . .	DIO-123
Form function . . . . .	DIO-124
Get procedure . . . . .	DIO-126, DIO-128
Get_Line function . . . . .	DIO-132
Get_Line procedure . . . . .	DIO-135
Insert procedure . . . . .	DIO-139
Last_Line function . . . . .	DIO-143
Line_Image function . . . . .	DIO-144
Line_Length function . . . . .	DIO-145
Mode function . . . . .	DIO-147
Move_Cursor procedure . . . . .	DIO-149
Name function . . . . .	DIO-150
New_Line procedure . . . . .	DIO-151
Overwrite procedure . . . . .	DIO-156
Position_Cursor procedure . . . . .	DIO-159
Read_Banner function . . . . .	DIO-161
Report_Cursor procedure . . . . .	DIO-163
Report_Location procedure . . . . .	DIO-164
Report_Origin procedure . . . . .	DIO-165
Report_Size procedure . . . . .	DIO-166
Set_Banner procedure . . . . .	DIO-168

Window\_Io.Raw package

Close procedure . . . . .	DIO-172
Get procedure . . . . .	DIO-177
Open procedure . . . . .	DIO-183

step, <i>see</i> Next, Run	
stepping . . . . .	DEB-13
jobs	
System_Uilities.Done function . . . . .	SMU-210
remove	
Debug.Clear_Stepping procedure . . . . .	DEB-42
sessions	
System_Uilities.Done function . . . . .	SMU-211
terminals	
System_Uilities.Done function . . . . .	SMU-212
<i>see also</i> Next	
Steps enumeration	
Debug.State_Type type . . . . .	DEB-127
stop	
bits	
Terminal.Set_Stop_Bits procedure . . . . .	SMU-322
output	
Text.Block procedure . . . . .	EST-146
shutdown	
Operator.Cancel_Shutdown procedure . . . . .	SMU-59
<i>see also</i> Cancel_Shutdown, Debug_Off, Kill, Quiesce, Un_Register	
[Stop] key	
Debug.Stop procedure . . . . .	DEB-128
Stop procedure	
Debug.Stop . . . . .	DEB-9, DEB-128
Context procedure . . . . .	DEB-45, DEB-47
Hold procedure . . . . .	DEB-71
Release procedure . . . . .	DEB-106
Xecute procedure . . . . .	DEB-148
Stop_Bits function	
System_Uilities.Stop_Bits . . . . .	SMU-263
Stop_Bits_Range subtype	
System_Uilities.Stop_Bits_Range . . . . .	SMU-264
Terminal.Stop_Bits_Range . . . . .	SMU-327
Stop_Event type	
Debug.Stop_Event . . . . .	DEB-130
Stopped enumeration	
Debug.Task_Category type . . . . .	DEB-135
stopped task state . . . . .	DEB-9
Stops_And_Holds enumeration	
Debug.State_Type type . . . . .	DEB-127
storage	
default	
Unbounded_String.Default_Maximum_Length generic formal object . . . . .	ST-109
management, <i>see</i> Allocate, Allows_Deallocation, Free, Unchecked_Deallocation	
reclaiming, <i>see</i> Allows_Deallocation, Unchecked_Deallocation	
Storage_Error exception	
Concurrent_Map_Generic generic package	
Initialize procedure . . . . .	PT-21
Debug package . . . . .	DEB-57
Map_Generic generic package	
Initialize procedure . . . . .	PT-79
Standard.Storage_Error . . . . .	PT-161

Storage_Error exception ( <i>continued</i> )	
System_Utilities package	
Get_Page_Counts procedure . . . . .	SMU-221
Set_Page_Limit procedure . . . . .	SMU-262
Storage_Unit constant	
System.Storage_Unit . . . . .	PT-166
stream operations . . . . .	EI-3
Stream_Type type	
Window_Io.Raw.Stream_Type . . . . .	DIO-185
strict stream policy . . . . .	SMU-138
String type	
Standard.String . . . . .	PT-161
String_Length subtype	
Bounded_String.String_Length . . . . .	ST-21
Unbounded_String.String_Length . . . . .	ST-123
String_Map_Generic generic package . . . . .	ST-25
String_Table package . . . . .	ST-49
String_To_Number procedure	
String_Utilities.String_To_Number . . . . .	ST-84
String_Utilities package . . . . .	ST-69
strings	
Bounded_String.Variable_String type . . . . .	ST-23
byte	
System_Utilities.Byte_String subtype . . . . .	SMU-202
case conversion of	
String_Utilities package . . . . .	ST-69
comparison of	
String_Utilities package . . . . .	ST-69
convert from byte	
Unchecked_Conversions.Convert_From_Byte_String function . . . . .	PT-232
Unchecked_Conversions.Convert_From_Byte_String generic function . . . . .	PT-231
convert to byte	
Unchecked_Conversions.Convert_To_Byte_String function . . . . .	PT-238
Unchecked_Conversions.Convert_To_Byte_String generic function . . . . .	PT-237
fill	
Bounded_String.Set_Length procedure . . . . .	ST-19
Unbounded_String.Set_Length procedure . . . . .	ST-122
insert	
Editor.Char.Insert_String procedure . . . . .	EI-13
name . . . . .	DEB-8, LM-7, PM-127, SJM-5, SMU-1
read enumeration value from	
Io.Enumeration_Io.Get procedure . . . . .	TIO-114
Text_Io.Enumeration_Io.Get procedure . . . . .	TIO-222
read fixed-point value from	
Io.Fixed_Io.Get procedure . . . . .	TIO-125
Text_Io.Fixed_Io.Get procedure . . . . .	TIO-233
read floating-point value from	
Io.Float_Io.Get procedure . . . . .	TIO-137
Text_Io.Float_Io.Get procedure . . . . .	TIO-245
read from file	
Io.Get procedure . . . . .	TIO-42
Text_Io.Get procedure . . . . .	TIO-182
read integer value from	
Io.Integer_Io.Get procedure . . . . .	TIO-148
Text_Io.Integer_Io.Get procedure . . . . .	TIO-256

strings (*continued*)

read single line except terminator	
Io.Get_Line procedure . . . . .	TIO-51
Text_Io.Get_Line procedure . . . . .	TIO-183
searching	
String_Uilities package . . . . .	ST-69
String_Uilities.Hash_String function . . . . .	ST-74
String_Uilities.Number_To_String function . . . . .	ST-81
System.Byte_String type . . . . .	PT-164
Unbounded_String.Variable_String type . . . . .	ST-125
variable length	
Bounded_String package . . . . .	ST-1
Unbounded_String package . . . . .	ST-103
Window_Io.Raw_Key_String type . . . . .	DIO-182
write enumeration value to	
Io.Enumeration_Io.Put procedure . . . . .	TIO-117
Text_Io.Enumeration_Io.Put procedure . . . . .	TIO-225
write fixed-point value to	
Io.Fixed_Io.Put procedure . . . . .	TIO-129
Text_Io.Fixed_Io.Put procedure . . . . .	TIO-237
write floating-point value to	
Io.Float_Io.Put procedure . . . . .	TIO-141
Text_Io.Float_Io.Put procedure . . . . .	TIO-249
write integer value to	
Io.Integer_Io.Put procedure . . . . .	TIO-152
Text_Io.Integer_Io.Put procedure . . . . .	TIO-260
write to current error file (Message window)	
Io.Echo procedure . . . . .	TIO-27
write to current error file (Message window)/advance line	
Io.Echo_Line procedure . . . . .	TIO-32
write to file	
Io.Put procedure . . . . .	TIO-73
Text_Io.Put procedure . . . . .	TIO-199
write to file/advance line	
Io.Put_Line procedure . . . . .	TIO-79
Text_Io.Put_Line procedure . . . . .	TIO-200
Strip function	
String_Uilities.Strip . . . . .	ST-86
Strip procedure	
File_Uilities.Strip . . . . .	LM-193
Strip_Leading function	
String_Uilities.Strip_Leading . . . . .	ST-87
Strip_Trailing function	
String_Uilities.Strip_Trailing . . . . .	ST-88
structural editing . . . . .	DIO-89, DIO-93
subclass . . . . .	LM-15
attributes . . . . .	LM-15, LM-16, LM-17
Ada . . . . .	LM-16, SJM-14
file . . . . .	LM-17, SJM-15
library . . . . .	LM-15, SJM-13
set	
Library.Set_Subclass procedure . . . . .	LM-261
Subclass enumeration	
Library.Field type . . . . .	LM-240
Subitem procedure	
Table_Formatter.Subitem . . . . .	ST-91, ST-101

Subitem_Separator generic formal object	
Table_Formatter.Subitem_Separator . . . . .	ST-102
Subp_Body subclass . . . . .	LM-16, SJM-14
Subp_Inst subclass . . . . .	LM-16, SJM-14
Subp_Ren subclass . . . . .	LM-16, SJM-14
Subp_Spec subclass . . . . .	LM-16, SJM-14
Subpackage enumeration	
Library.Kind type . . . . .	LM-246
subpath . . . . .	PM-13, PM-37, PM-224, PM-227, PM-280
creating . . . . .	PM-37
differences between paths and subpaths . . . . .	PM-47
integrating into a single release . . . . .	PM-46
name extension . . . . .	PM-37, PM-47
substitute, <i>see</i> Replace	
substitution characters . . . . .	LM-9, PM-130, SJM-7, SMU-4
at sign (@) . . . . .	LM-10, PM-130, SJM-8, SMU-4
pound sign (#) . . . . .	LM-10, PM-130, SJM-7, SMU-4
question mark (?) . . . . .	LM-10, PM-130, SJM-8, SMU-5
substitution, in names . . . . .	LM-7, SJM-5
subsystem . . . . .	PM-3, PM-4, PM-5, PM-136, PM-137, PM-187
access control . . . . .	LM-23
child . . . . .	PM-16
compatibility database . . . . .	LM-90
compilation . . . . .	LM-130
compiling units . . . . .	PM-29
copying identification number . . . . .	PM-104
copying releases and code views . . . . .	PM-104
copying views among hosts . . . . .	PM-103, PM-109
creating . . . . .	PM-17, PM-20, PM-98
sample program . . . . .	PM-17
defined . . . . .	PM-3, PM-4
developing applications using multiple . . . . .	PM-51
developing with joined objects . . . . .	PM-38
development paths . . . . .	PM-33
editing controlled objects . . . . .	PM-26
executing an entire application . . . . .	PM-84
exports . . . . .	PM-10
identification number . . . . .	PM-104
imports . . . . .	PM-10
interfaces . . . . .	PM-10, PM-11
internal structure . . . . .	PM-20, PM-24
making design changes . . . . .	PM-89
making implementation changes . . . . .	PM-86
managing CMVC information interactively . . . . .	PM-188
managing views . . . . .	PM-48
moving a primary to another host . . . . .	PM-108
primary . . . . .	LM-90, PM-2, PM-16, PM-339, PM-351, PM-354
program development within . . . . .	PM-13
propagating changes across hosts . . . . .	PM-105
releasing configurations . . . . .	PM-30
secondary . . . . .	LM-90, PM-2, PM-16, PM-339, PM-351, PM-354
setting up . . . . .	PM-96
for cross-development . . . . .	PM-115
multiple development paths . . . . .	PM-47
primary and secondary . . . . .	PM-103
Units directory . . . . .	PM-23
testing an application . . . . .	PM-85

subsystem ( <i>continued</i> )	
using CDFs with . . . . .	PM-111
using CMVC . . . . .	PM-17
working view . . . . .	PM-21
predefined library characteristics . . . . .	PM-22
putting objects under CMVC . . . . .	PM-25
Subsystem subclass . . . . .	LM-15, SJM-13
Subsystem_Interface library switch . . . . .	LM-314
Subsystem_Name subtype	
Activity.Subsystem_Name . . . . .	PM-166
<SUBUNITS> special value . . . . .	LM-129, LM-134, LM-199
Subunits_Too enumeration	
Compilation.Promote_Scope type . . . . .	LM-161
successful start	
Program.Started_Successfully function . . . . .	SJM-195
sum	
Calendar.+ function . . . . .	PT-8
Summarize renamed procedure	
Log.Summarize . . . . .	SJM-61
Summary constant	
Profile.Summary . . . . .	SJM-165
Sun_Positions type	
Time_Utilities.Sun_Positions . . . . .	PT-202
super user, <i>see</i> Change_Identity, Enable_Privileges, privileged mode	
supplier . . . . .	PM-70
swap, <i>see</i> Transpose	
Swap procedure	
Editor.Hold_Stack.Swap . . . . .	EI-21, EI-23
Editor.Mark.Swap . . . . .	EI-4, EI-41, EI-43
Editor.Screen.Swap . . . . .	EI-7, EI-52, EI-54
switch file	
association	
Switches.Associate procedure . . . . .	LM-318
Switches.Associated function . . . . .	LM-320
Switches.Dissociate procedure . . . . .	LM-328
constant	
Switches.Of_Library constant . . . . .	LM-333
Switches.Of_Session constant . . . . .	LM-334
create	
Switches.Create procedure . . . . .	LM-323
Switches.Define procedure . . . . .	LM-325
default	
Switches.Default_File constant . . . . .	LM-324
display	
Switches.Display . . . . .	LM-326
edit	
Switches.Edit procedure . . . . .	LM-329
Switches.Visit procedure . . . . .	LM-338
name	
Switches.File_Name subtype . . . . .	LM-331
set switches	
Switches.Set procedure . . . . .	LM-335

switch images . . . . .	EST-1
Switch subclass . . . . .	LM-17, LM-308, SJM-15, SJM-227
switches	
commentary messages . . . . .	LM-5
Common package . . . . .	LM-315
composite name	
Switches.Composite_Name subtype . . . . .	LM-322
edit session switches	
Switches.Edit_Session_Attributes procedure . . . . .	LM-330
error messages . . . . .	LM-5
exception messages . . . . .	LM-5
insert	
Switches.Insert procedure . . . . .	LM-332
library . . . . .	EST-60, LM-5, LM-309, SJM-227
Account . . . . .	LM-309
Ada images . . . . .	EST-9
Ada units . . . . .	LM-309
Alignment_Threshold . . . . .	LM-309
Asm_Listing . . . . .	LM-309
Auto_Login . . . . .	LM-310
Closed_Private_Part . . . . .	LM-310
command images . . . . .	EST-47
Comment_Column . . . . .	EI-15, LM-310
Configuration . . . . .	LM-310
Consistent_Breaking . . . . .	LM-310
Create_Internal_Links . . . . .	LM-7, LM-310
Create_Subprogram_Specs . . . . .	LM-310
Enable_Deallocation . . . . .	LM-311, PT-241
Id_Case . . . . .	LM-311
Ignore_Interface_Pragmas . . . . .	LM-311
Ignore_Minor_Errors . . . . .	LM-311
Ignore_Unsupported_Rep_Specs . . . . .	LM-311
Keyword_Case . . . . .	EST-9, EST-14, LM-311
Line_Length . . . . .	LM-311
links . . . . .	LM-309
listings . . . . .	LM-309
Major_Indentation . . . . .	LM-312
Minor_Indentation . . . . .	LM-312
networking . . . . .	LM-309
Number_Case . . . . .	LM-312
Page_Limit . . . . .	LM-312, SMU-220, SMU-261
Password . . . . .	LM-312
Prompt_For_Account . . . . .	LM-312
Prompt_For_Password . . . . .	LM-312
Remote_Directory . . . . .	LM-312
Remote_Machine . . . . .	LM-313
Remote_Root . . . . .	LM-313
Remote_Type . . . . .	LM-313
Require_Internal_Links . . . . .	LM-313
Seg_Listing . . . . .	LM-313
Send_Port_Enabled . . . . .	LM-313
Statement_Indentation . . . . .	LM-313
Statement_Length . . . . .	LM-313
Subsystem_Interface . . . . .	LM-314
Target_Key . . . . .	LM-314
Terminal_Echo . . . . .	LM-314
Transfer_Mode . . . . .	LM-314
Transfer_Structure . . . . .	LM-314
Transfer_Type . . . . .	LM-314
Username . . . . .	LM-314
Wrap_Indentation . . . . .	LM-314



switches (continued)

overview	LM-308
parameter placeholders	LM-308
progress messages	LM-5
session	DEB-4, EI-7, EST-140, LM-5, LM-200, PM-362, SJM-1, SJM-3, SJM-227, SJM-228, SJM-230
Account	SJM-230
Activity_File	PM-161, SJM-230
Ada units	SJM-228
Auto_Login	SJM-230
Banner	SJM-230
Beep_On_Errors	SJM-230
Beep_On_Interrupt	SJM-230
Beep_On_Messages	SJM-231
Cmvc_Break_Long_Lines	PM-362
Cmvc_Capitalize	PM-362
Cmvc_Comment_Extent	PM-362
Cmvc_Configuration_Extent	PM-362
Cmvc_Enable_Relocation	PM-196
Cmvc_Field_Extent	PM-362
Cmvc_Indentation	PM-362
Cmvc_Line_Length	PM-362
Cmvc_Shorten_Name	PM-362
Cmvc_Shorten_Unit_State	PM-363
Cmvc_Show_Add_Date	PM-363
Cmvc_Show_Add_Time	PM-363
Cmvc_Show_All_Default_Lists	PM-363
Cmvc_Show_All_Default_Orders	PM-363
Cmvc_Show_Boolean	PM-364
Cmvc_Show_Deleted_Objects	PM-363
Cmvc_Show_Deleted_Versions	PM-363
Cmvc_Show_Display_Position	PM-363
Cmvc_Show_Edit_Info	PM-363
Cmvc_Show_Field_Default	PM-363
Cmvc_Show_Field_Max_Index	PM-363
Cmvc_Show_Field_Type	PM-363
Cmvc_Show_Frozen	PM-364
Cmvc_Show_Hidden_Fields	PM-364
Cmvc_Show_Retention	PM-364
Cmvc_Show_Unit_State	PM-364
Cmvc_Show_Users	PM-364
Cmvc_Show_Version_Number	PM-364
Cmvc_Uppercase	PM-364
Cmvc_Version_Extent	PM-364
Cursor_Bottom_Offset	SJM-231
Cursor_Left_Offset	SJM-231
Cursor_Right_Offset	SJM-231
Cursor_Top_Offset	SJM-231
Cursor_Transpose_Moves	EI-15, EI-34, EI-68, EI-71, SJM-231
Debug_Addresses	SJM-232
Debug_Break_At_Creation	SJM-232
Debug_Declaration_Display	SJM-232
Debug_Delete_Temporary_Breaks	SJM-232
Debug_Display_Count	SJM-232
Debug_Display_Creation	SJM-232
Debug_Display_Level	SJM-232
Debug_Echo_Commands	SJM-232
Debug_Element_Count	SJM-232
Debug_First_Element	SJM-233
Debug_Freeze_Tasks	SJM-233
Debug_History_Count	SJM-233
Debug_History_Entries	SJM-233
Debug_History_Start	SJM-233

## switches (continued)

## session (continued)

Debug_Include_Packages	SJM-233
Debug_Interpret_Control_Words	SJM-233
Debug_Kill_Old_Jobs	SJM-233
Debug_Machine_Level	SJM-233
Debug_Memory_Count	SJM-234
Debug_No_History_Timestamps	SJM-234
Debug_Optimize_Generic_History	SJM-234
Debug_Permanent_Breakpoints	SJM-234
Debug_Pointer_Level	SJM-234
Debug_Put_Locals	SJM-234
Debug_Qualify_Stack_Names	SJM-234
Debug_Require_Debug_Off	SJM-234
Debug_Save_Exceptions	SJM-235
Debug_Show_Location	SJM-235
Debug_Stack_Count	SJM-235
Debug_Stack_Start	SJM-235
Debug_Timestamps	SJM-235
debugging	SJM-228
Default_Job_Page_Limit	SJM-235, SMU-220, SMU-261
Default_Venture	PM-364, PM-394
editing_images	SJM-228
Escape	SJM-235
Escape_On_Break	SJM-235
Footer	SJM-235
Header	SJM-236
Image_Fill_Column	EI-47, EI-48, SJM-236
Image_Fill_Extra_Space	EI-48, SJM-236
Image_Fill_Indent	EI-48, SJM-236
Image_Fill_Mode	SJM-236
Image_Fill_Prefix	EI-47, EI-48, SJM-236
Image_Insert_Mode	SJM-236
Image_Tab_Stops	SJM-237
Job_Context_First	SJM-237
Job_Context_Length	SJM-237
Job_Name_Length	SJM-237
Job_Name_Separator	SJM-237
Key_Directory	SJM-237
library display	SJM-229
Library_Break_Long_Lines	LM-200, SJM-237
Library_Capitalize	LM-200, SJM-237
Library_Indentation	LM-201, SJM-238
Library_Lazy_Realignment	LM-201, SJM-238
Library_Line_Length	LM-201, SJM-238
Library_Misc_Show_Edit_Info	LM-201, SJM-238
Library_Misc_Show_Frozen	LM-201, SJM-238
Library_Misc_Show_Retention	LM-201, SJM-238
Library_Misc_Show_Size	LM-201, SJM-238
Library_Misc_Show_Subclass	LM-201, SJM-238
Library_Misc_Show_Unit_State	LM-201, SJM-238
Library_Misc_Show_Volume	LM-201, SJM-239
Library_Shorten_Names	LM-201, SJM-239
Library_Shorten_Subclass	LM-202, SJM-239
Library_Shorten_Unit_State	LM-202, SJM-239
Library_Show_Deleted_Objects	LM-202, SJM-239
Library_Show_Deleted_Versions	LM-202, SJM-239
Library_Show_Miscellaneous	LM-202, SJM-239
Library_Show_Standard	LM-202, SJM-239
Library_Show_Subunits	LM-202, SJM-239
Library_Show_Version_Number	LM-202, SJM-240
Library_Std_Show_Class	LM-202, SJM-240

## switches (continued)

## session (continued)

Library_Std_Show_Subclass . . . . .	LM-202, SJM-240
Library_Std_Show_Unit_State . . . . .	LM-202, SJM-240
Library_Uppercase . . . . .	LM-203, SJM-240
log operations . . . . .	SJM-229
Log_At_Sign_Msgs . . . . .	SJM-241
Log_Auxiliary_Msgs . . . . .	SJM-241
Log_Diagnostic_Msgs . . . . .	SJM-241
Log_Dollar_Msgs . . . . .	SJM-241
Log_Error_Msgs . . . . .	SJM-241
Log_Exception_Msgs . . . . .	SJM-241
Log_File . . . . .	SJM-241
Log_Line_Width . . . . .	SJM-241
Log_Negative_Msgs . . . . .	SJM-242
Log_Note_Msgs . . . . .	SJM-242
Log_Position_Msgs . . . . .	SJM-242
Log_Positive_Msgs . . . . .	SJM-242
Log_Prefix_1 . . . . .	SJM-242
Log_Prefix_2 . . . . .	SJM-242
Log_Prefix_3 . . . . .	SJM-242
Log_Sharp_Msgs . . . . .	SJM-242
Log_Warning_Msgs . . . . .	SJM-243
networking . . . . .	SJM-229
Notify_Warnings . . . . .	SJM-243
Options . . . . .	SJM-243
Password . . . . .	SJM-243
printing . . . . .	SJM-230
profile operations . . . . .	SJM-229
Prompt_Delimiters . . . . .	SJM-243
Prompt_For_Account . . . . .	SJM-243
Prompt_For_Password . . . . .	SJM-243
Reaction . . . . .	SJM-243
Recovery_Locality . . . . .	SJM-244
Remote_Directory . . . . .	SJM-244
Remote_Machine . . . . .	SJM-244
Remote_Passwords . . . . .	SJM-244
Remote_Roof . . . . .	SJM-244
Remote_Sessions . . . . .	SJM-244
Remote_Type . . . . .	SJM-244
Screen_Dump_File . . . . .	SJM-244
Search_Ignore_Case . . . . .	SJM-245
Search_Preserve_Case . . . . .	SJM-245
Search_Regular_Expr . . . . .	SJM-245
Send_Port_Enabled . . . . .	SJM-245
text images . . . . .	EST-140
text input/output . . . . .	SJM-229
Text_Bottom_Stripe . . . . .	SJM-245
Text_Convert_Tabs . . . . .	SJM-245
Text_Header . . . . .	SJM-245
Text_Print_Name . . . . .	SJM-245
Text_Print_Number . . . . .	SJM-246
Text_Print_Time . . . . .	SJM-246
Text_Reuse_Window . . . . .	SJM-246
Text_Scroll_Output . . . . .	SJM-246
Text_Top_Stripe . . . . .	SJM-246
Transfer_Mode . . . . .	SJM-246
Transfer_Structure . . . . .	SJM-246
Transfer_Type . . . . .	SJM-246
Username . . . . .	SJM-246
Window_Command_Size . . . . .	SJM-247
Window_Frames . . . . .	SJM-247

switches (continued)

session (continued)

Window_Frames_Startup . . . . .	SJM-247
Window_Have_Sides . . . . .	SJM-247
Window_Is_Staggered . . . . .	SJM-247
Window_Message_Life . . . . .	SJM-247
Window_Message_Size . . . . .	SJM-247
Window_Scroll_Overlap . . . . .	SJM-247
Window_Shift_Overlap . . . . .	SJM-248
Word_Breaks . . . . .	EI-70, EST-138, SJM-248

set

Switches.Set procedure . . . . .	LM-335
----------------------------------	--------

special names . . . . .	LM-307
-------------------------	--------

value

Switches.Value_Image subtype . . . . .	LM-337
--	--------

warning messages . . . . .	LM-5
----------------------------	------

Switches package . . . . .	LM-307
----------------------------	--------

Swch_Def subclass . . . . .	LM-17, SJM-15
-----------------------------	---------------

symbol table

String_Map_Generic generic package . . . . .	ST-25
--	-------

String_Table package . . . . .	ST-49
--------------------------------	-------

symbolic name

System_Uilities.Job_Name function . . . . .	SMU-233
---	---------

symbolize, *see* Address\_To\_Location

symbols

# . . . . .	PM-135, PM-404
-------------	----------------

* . . . . .	PM-210, PM-232, PM-244, PM-292
-------------	--------------------------------

*, . . . . .	PM-45
--------------	-------

+ . . . . .	PM-404
-------------	--------

= . . . . .	PM-135
-------------	--------

Symbols enumeration

Profile.Log_Prefix type . . . . .	SJM-114
-----------------------------------	---------

synchronization . . . . .	DIO-5, TIO-5
---------------------------	--------------

syntax error, *see* Input\_Syntax\_Error

syntax rules . . . . .	LM-18, SJM-15
------------------------	---------------

system . . . . .	PM-16, PM-187, PM-325
------------------	-----------------------

default

Search_List.Reset_To_System_Default procedure . . . . .	SJM-220
---	---------

default profile . . . . .	SJM-75
---------------------------	--------

diagnosis

Operator.Internal_System_Diagnosis procedure . . . . .	SMU-79
--	--------

object . . . . .	PM-187
------------------	--------

setting up . . . . .	PM-326
----------------------	--------

view . . . . .	PM-187, PM-325
----------------	----------------

releasing . . . . .	PM-327
---------------------	--------

System enumeration

Cmvc.System_Object_Enum type . . . . .	PM-324
--	--------

System format (Debug.Memory_Display) . . . . .	DEB-79
--	--------

System package . . . . .	PT-163
--------------------------	--------

System_Backup package . . . . .	SMU-191
---------------------------------	---------

System\_Boot\_Configuration function

System_Uilities.System_Boot_Configuration . . . . .	SMU-265
---	---------

System_Name constant	
System.System_Name . . . . .	PT-166
System_Object_Enum type	
Cmvc.System_Object_Enum . . . . .	PM-324
System_Up_Time function	
System_Uilities.System_Up_Time . . . . .	SMU-266
System_Uilities package . . . . .	SMU-197

T

T generic formal type	
Debug_Tools.T . . . . .	DEB-177, DEB-182
Hash.T . . . . .	PT-42, PT-46
tab	
backward	
Editor.Char.Tab_Backward procedure . . . . .	EI-14
column width	
Editor.Set.Tab_Width procedure . . . . .	EI-62
forward	
Editor.Char.Tab_Forward procedure . . . . .	EI-14
remove	
Editor.Set.Tab_Off procedure . . . . .	EI-62
set	
Editor.Set.Tab_On procedure . . . . .	EI-62
stops	
What.Tabs procedure . . . . .	SJM-269
to comment	
Editor.Char.Tab_To_Comment procedure . . . . .	EI-15
Tab_Backward procedure	
Editor.Char.Tab_Backward . . . . .	EI-5, EI-6, EI-11, EI-14
Tab_Forward procedure	
Editor.Char.Tab_Forward . . . . .	EI-5, EI-6, EI-11, EI-14
Tab_Off procedure	
Editor.Set.Tab_Off . . . . .	EI-6, EI-62
Image_Tab_Stops session switch . . . . .	SJM-237
Tab_On procedure	
Editor.Set.Tab_On . . . . .	EI-6, EI-62
Image_Tab_Stops session switch . . . . .	SJM-237
Tab_Width procedure . . . . .	EI-62
Tab_To_Comment procedure	
Editor.Char.Tab_To_Comment . . . . .	EI-5, EI-6, EI-15
Tab_Width procedure	
Editor.Set.Tab_Width . . . . .	EI-6, EI-62
Tab_On procedure . . . . .	EI-62
table	
new	
String_Table.New_Table function . . . . .	ST-61
sorting	
Table_Formatter.Field_List type . . . . .	ST-95
symbol	
String_Map_Generic generic package . . . . .	ST-25
String_Table package . . . . .	ST-49

Table type	
String_Table.Table	ST-64
Table_Formatter generic package	ST-91
Table_Full exception	
String_Table.Table_Full	ST-49, ST-65
Table_Sort_Generic generic procedure	PT-169
Table_Sort_Generic procedure	
Table_Sort_Generic.Table_Sort_Generic	PT-174
Tabs procedure	
What.Tabs	SJM-269
Editor.Set.Tab_On procedure	EI-62
Editor.Set.Tab_Width procedure	EI-62
tail, <i>see</i> Rest	
Take_History procedure	
Debug.Take_History	DEB-12, DEB-132
Context procedure	DEB-45
Trace_Event type	DEB-146
tape	DIO-2, TIO-2
blue	SMU-191
data	SMU-191
display	
Tape.Display_Tape procedure	SMU-284
format	
Tape.Format_Tape procedure	SMU-288
label	
Tape.Examine_Labels procedure	SMU-287
read	
Tape.Read procedure	SMU-289
rewind	
Tape.Rewind procedure	SMU-292
unload	
Tape.Unload procedure	SMU-293
write	
Tape.Write procedure	SMU-294
Tape class	LM-14, SJM-12
Tape object manager	SMU-11, SMU-58
Tape package	SMU-283
Tape subtype	
System_Uilities.Tape	SMU-267
Tape_Name function	
System_Uilities.Tape_Name	SMU-268
Target generic formal type	
Unchecked_Conversion.Target	PT-211
Unchecked_Conversions.Target	PT-235
Unchecked_Conversions.Unchecked_Conversion_Package.Target	PT-230
target key	PM-113, PM-303
Target_Key library switch	LM-314
task	PM-15
breakpoints	DEB-10
call stacks	DEB-8
control	DEB-7

task (continued)

current	DEB-7
Debug.Display procedure	DEB-54
display history	
Debug.History_Display procedure	DEB-68
display name	
Debug_Tools.Get_Task_Name function	DEB-161
display stack	
Debug.Stack procedure	DEB-123
exception trapping	DEB-12
history facility	DEB-12
hold	DEB-8
name	
Debug.Set_Task_Name procedure	DEB-112
Debug.Task_Name subtype	DEB-140
Debug_Tools.Set_Task_Name procedure	DEB-179
nickname	
Debug.Context procedure	DEB-46, DEB-47
Debug.Set_Task_Name procedure	DEB-112
Debug_Tools.Set_Task_Name procedure	DEB-179
programmatic access to Debugger facilities	DEB-16
release from held state	
Debug.Release procedure	DEB-106
remove stepping	
Debug.Clear_Stepping procedure	DEB-42
resume execution	
Debug.Execute procedure	DEB-61
Debug.Xecute procedure	DEB-148
root	DEB-7
show	
Debug.Task_Display procedure	DEB-136
state	DEB-7, DEB-9
stepping	DEB-13
stop execution	DEB-8
Debug.Hold procedure	DEB-71
Debug.Stop procedure	DEB-128
trace	DEB-12
Debug.Trace procedure	DEB-142
when to stop	
Debug.Stop_Event type	DEB-130
[Task Display] key	
Debug.Task_Display procedure	DEB-136
Task_Body subclass	LM-16, SJM-14
Task_Category type	
Debug.Task_Category	DEB-135
Task_Display procedure	
Debug.Task_Display	DEB-8, DEB-19, DEB-136, LM-11, SJM-9, SMU-6
Debug_Addresses session switch	SJM-232
Debug_Include_Packages session switch	SJM-233
Option type	DEB-86, DEB-87
Task_Category type	DEB-135
Task_Name subtype	DEB-140
Task_Name subtype	
Debug.Task_Name	DEB-140
Tasking_Error exception	
Debug package	DEB-57
Standard.Tasking_Error	PT-161

temporary breakpoint . . . . .	DEB-11
temporary file	
Direct_Io.Create procedure . . . . .	DIO-10
Polymorphic_Sequential_Io.Create procedure . . . . .	DIO-42
Sequential_Io.Create procedure . . . . .	DIO-63
TERMCAP (creating your own terminal type) . . . . .	SMU-300
terminal	
access . . . . .	DIO-80
bell	
Editor.Alert procedure . . . . .	EI-10
characteristics	
Terminal package . . . . .	SMU-299
connect to job	
Job.Connect procedure . . . . .	SJM-20
control of . . . . .	DIO-91
device characteristics . . . . .	SMU-299
disable	
Operator.Disable_Terminal procedure . . . . .	SMU-68
enable	
Operator.Enable_Terminal procedure . . . . .	SMU-73
input/output . . . . .	DIO-3, TIO-3
keyboard input . . . . .	DIO-80
line	
Terminal.Port subtype . . . . .	SMU-305
login from non-Rational type . . . . .	SMU-300
number	
System_Uilities.Value function . . . . .	SMU-278
options	
Window_Io.Bell procedure . . . . .	DIO-104
ports	
System_Uilities.Port subtype . . . . .	SMU-250
settings	
Terminal.Settings procedure . . . . .	SMU-326
stepping	
System_Uilities.Done function . . . . .	SMU-212
types . . . . .	DIO-85, SMU-299
Terminal.Set_Terminal_Type procedure . . . . .	SMU-323
Terminal class . . . . .	LM-15, SJM-12
Terminal function	
System_Uilities.Terminal . . . . .	SMU-269
Terminal object manager . . . . .	SMU-11, SMU-58
Terminal package . . . . .	SMU-299
Terminal subtype	
Window_Io.Raw.Terminal . . . . .	DIO-186
Terminal_Echo library switch . . . . .	LM-314
Terminal_Iterator type	
System_Uilities.Terminal_Iterator . . . . .	SMU-271
Terminal_Name function	
System_Uilities.Terminal_Name . . . . .	SMU-272
Terminal_Type function	
System_Uilities.Terminal_Type . . . . .	SMU-273
Terminated enumeration	
Scheduler.Job_Kind type . . . . .	SMU-166
terminated execution message (Debug.Task_Display) . . . . .	DEB-137



terminated job . . . . .	SMU-133
termination message	
Job.Set_Termination_Message procedure . . . . .	SJM-31
terminators . . . . .	DIO-6, TIO-5
Terse constant	
Profile.Terse . . . . .	SJM-166
Terse_Format constant	
Library.Terse_Format . . . . .	LM-265
test	
application . . . . .	PM-85
recombinant . . . . .	PM-85
text	
definition . . . . .	EST-137
edit . . . . .	EI-5
image type . . . . .	EST-137
images . . . . .	EST-1, EST-137
commands from package Common . . . . .	EST-140
committing . . . . .	EST-139
designation . . . . .	EST-138
job I/O . . . . .	EST-139
key concepts . . . . .	EST-138
locks . . . . .	EST-139
session switches . . . . .	EST-140
structure . . . . .	EST-137
versions . . . . .	EST-139
normal	
Editor.Set.Designation_Off procedure . . . . .	EI-60
Profile.Cached_Selected_Text function . . . . .	SJM-82
retrieve . . . . .	EI-5
select . . . . .	EI-6
Text enumeration	
Window_Io.Designation type . . . . .	DIO-116
Text field	
Window_Io package . . . . .	DIO-83
text file . . . . .	DIO-1, TIO-1, TIO-7
create	
Io.Create procedure . . . . .	TIO-20
Text.Create procedure . . . . .	EST-148
Text_Io.Create procedure . . . . .	TIO-169
text images . . . . .	EST-137
text input/output	
session switches . . . . .	SJM-229
Text package . . . . .	EST-145
<TEXT> special name . . . . .	EST-58, EST-131, EST-155, LM-8, LM-130, LM-170, LM-199, LM-308, PM-128, SJM-5, SMU-2
Text subclass . . . . .	LM-17, SJM-15
Text_Bottom_Stripe session switch . . . . .	SJM-245
Text_Convert_Tabs session switch . . . . .	SJM-245
Text_Header session switch . . . . .	SJM-245
Text_Io package . . . . .	TIO-165
Text_Print_Name session switch . . . . .	SJM-245

Text_Print_Number session switch . . . . .	SJM-246
Text_Print_Time session switch . . . . .	SJM-246
Text_Reuse_Window session switch . . . . .	SJM-246
Text_Scroll_Output session switch . . . . .	SJM-246
Text_Top_Stripe session switch . . . . .	SJM-246
The_Current_Activity function	
Activity.The_Current_Activity . . . . .	PM-167
The_Enclosing_Subsystem function	
Activity.The_Enclosing_Subsystem . . . . .	PM-168
The_Enclosing_View function	
Activity.The_Enclosing_View . . . . .	PM-169
threshold	
Daemon.Get_Log_Threshold function . . . . .	SMU-19
Daemon.Log_Threshold type . . . . .	SMU-26
Daemon.Set_Log_Threshold procedure . . . . .	SMU-37
Daemon.Show_Log_Thresholds procedure . . . . .	SMU-39
Threshold_Warnings procedure	
Daemon.Threshold_Warnings . . . . .	SMU-46
throw away, <i>see</i> Delete	
Tick constant	
System.Tick . . . . .	PT-167
tilde (~)	
indicating replace window state . . . . .	EI-63, EI-65, EI-67
symbol . . . . .	LM-13, LM-18, LM-109, LM-113, LM-297, LM-301, PM-133, SJM-16, SMU-8, SMU-9
time	
formats	
Operator.Set_System_Time procedure . . . . .	SMU-84
job	
Window_Io.Job_Time function . . . . .	DIO-142
Operator.Set_System_Time procedure . . . . .	SMU-84
Scheduler.Get_Cpu_Time_Used function . . . . .	SMU-151
System_Utilities.Elapsed function . . . . .	SMU-213
System_Utilities.System_Up_Time function . . . . .	SMU-266
Time_Utilities.Convert_Time function . . . . .	PT-179
Time_Utilities.Get_Time function . . . . .	PT-187
Time enumeration	
Profile.Log_Prefix type . . . . .	SJM-115
Time procedure	
What.Time . . . . .	SJM-270
Time type	
Calendar.Time . . . . .	PT-7
Time_Utilities.Time . . . . .	PT-175, PT-203
Image function . . . . .	PT-190
Time_Error exception	
Calendar.Time_Error . . . . .	PT-7
Time_Format type	
Time_Utilities.Time_Format . . . . .	PT-204
Time_Of function	
Calendar.Time_Of . . . . .	PT-7
Time_Only enumeration	
Time_Utilities.Image_Contents type . . . . .	PT-192

Time_Uilities package . . . . .	PT-175
Timestamps enumeration	
Debug.Option type . . . . .	DEB-88
top	
hold stack	
Editor.Hold_Stack.Copy_Top procedure . . . . .	EI-21
Editor.Hold_Stack.Delete_Top procedure . . . . .	EI-22
Editor.Hold_Stack.Top procedure . . . . .	EI-23
image	
Editor.Image.Beginning_Of procedure . . . . .	EI-25
mark	
Editor.Mark.Copy_Top procedure . . . . .	EI-41
Editor.Mark.Delete_Top procedure . . . . .	EI-42
Editor.Mark.Top procedure . . . . .	EI-43
screen	
Editor.Screen.Copy_Top procedure . . . . .	EI-52
Editor.Screen.Delete_Top procedure . . . . .	EI-52
Editor.Screen.Top procedure . . . . .	EI-54
selection	
Editor.Region.Beginning_Of procedure . . . . .	EI-45
window	
Editor.Window.Beginning_Of procedure . . . . .	EI-64
Top function	
Stack_Generic.Top . . . . .	PT-157
Top procedure	
Editor.Hold_Stack.Top . . . . .	EI-21, EI-23
Editor.Mark.Top . . . . .	EI-4, EI-41, EI-43
Editor.Screen.Top . . . . .	EI-7, EI-51, EI-54
trace	
messages	
Debug.Trace procedure . . . . .	DEB-142
<i>see also</i> Input_Logging_To	
Trace procedure	
Debug.Trace . . . . .	DEB-12, DEB-142
Context procedure . . . . .	DEB-45
Debug_Addresses session switch . . . . .	SJM-232
Display procedure . . . . .	DEB-53
Option type . . . . .	DEB-86
Trace_Event type . . . . .	DEB-146
Trace_Event type	
Debug.Trace_Event . . . . .	DEB-146
Trace_To_File procedure	
Debug.Trace_To_File . . . . .	DEB-147
Trace procedure . . . . .	DEB-142
Traces enumeration	
Debug.State_Type type . . . . .	DEB-127
tracing facility . . . . .	DEB-12
trailing characters	
String_Uilities.Strip function . . . . .	ST-86
String_Uilities.Strip_Trailing function . . . . .	ST-88
Transfer_Mode library switch . . . . .	LM-314
Transfer_Mode session switch . . . . .	SJM-246
Transfer_Structure library switch . . . . .	LM-314

Transfer_Structure session switch . . . . .	SJM-246
Transfer_Type library switch . . . . .	LM-314
Transfer_Type session switch . . . . .	SJM-246
transitive closure . . . . .	PM-68
Transpose procedure . . . . .	EI-5
Editor.Char.Transpose . . . . .	EI-15
Editor.Line.Transpose . . . . .	EI-31, EI-34
Editor.Window.Transpose . . . . .	EI-64, EI-68
Editor.Word.Transpose . . . . .	EI-71
Traverse_Job_Descriptors generic procedure	
Scheduler.Traverse_Job_Descriptors . . . . .	SMU-185
Traverse_Job_Descriptors procedure	
Scheduler.Traverse_Job_Descriptors . . . . .	SMU-187
Put generic formal procedure . . . . .	SMU-186
trip count . . . . .	DEB-11
truncate string, <i>see</i> Set_Length	
turn off	
Operator.Shutdown procedure . . . . .	SMU-88
Typ format (Debug.Memory_Display) . . . . .	DEB-79
type	
activity	
Profile.Activity_Type subtype . . . . .	SJM-81
context	
Debug.Context_Type type . . . . .	DEB-48
domain	
Concurrent_Map_Generic.Domain_Type generic formal type . . . . .	PT-13
Map_Generic.Domain_Type generic formal type . . . . .	PT-71
element	
Direct_Io.Element_Type generic formal type . . . . .	DIO-13
Polymorphic_Sequential_Io.Operations.Element_Type generic formal type . . . . .	DIO-56
Sequential_Io.Element_Type generic formal type . . . . .	DIO-66
error	
Simple_Status.Error_Type function . . . . .	PT-138
<i>see also</i> Output_Type_Error	
file	
Direct_Io.File_Type type . . . . .	DIO-16
Io.File_Type type . . . . .	TIO-38
Polymorphic_Sequential_Io.File_Type type . . . . .	DIO-47
Sequential_Io.File_Type type . . . . .	DIO-69
Text_Io.File_Type type . . . . .	TIO-179
Window_Io.File_Type type . . . . .	DIO-122
information	
Debug.Informaion_Type type . . . . .	DEB-75
range	
Concurrent_Map_Generic.Range_Type generic formal type . . . . .	PT-31
Map_Generic.Range_Type generic formal type . . . . .	PT-89
String_Map_Generic.Range_Type generic formal type . . . . .	ST-44
state	
Debug.State_Type type . . . . .	DEB-126
stream	
Window_Io.Raw.Stream_Type type . . . . .	DIO-185
terminal	
System_Uilities.Terminal_Type function . . . . .	SMU-273
Terminal.Set_Terminal_Type procedure . . . . .	SMU-323

Type_Error exception	
System.Type_Error . . . . .	PT-167
Unchecked_Conversion.Source generic formal type . . . . .	PT-210
Unchecked_Conversion.Target generic formal type . . . . .	PT-211
Unchecked_Conversion.Unchecked_Conversion function . . . . .	PT-212, PT-213
Unchecked_Conversions.Convert_From_Byte_String function . . . . .	PT-232, PT-233
Unchecked_Conversions.Convert_To_Byte_String function . . . . .	PT-238
Unchecked_Conversions.Source generic formal type . . . . .	PT-240
Unchecked_Conversions.Target generic formal type . . . . .	PT-235
Unchecked_Conversions.Unchecked_Conversion_Package.Convert function . . . . .	PT-226, PT-227
Unchecked_Conversions.Unchecked_Conversion_Package.Source generic formal type . . . . .	PT-229
Unchecked_Conversions.Unchecked_Conversion_Package.Target generic formal type . . . . .	PT-230
Type_Set subtype	
Io.Type_Set . . . . .	TIO-102
Io.Enumeration_Io generic package . . . . .	TIO-109
Type_Set type	
Text_Io.Type_Set . . . . .	TIO-215
Text_Io.Enumeration_Io generic package . . . . .	TIO-217

U

Un_Register generic procedure		
Debug_Tools.Un_Register . . . . .	DEB-181	
Un_Register procedure		
Debug_Tools.Un_Register . . . . .	DEB-183	
Unbounded constant		
Io.Unbounded . . . . .	TIO-8, TIO-103	
Text_Io.Unbounded . . . . .	TIO-165, TIO-216	
Unbounded_String generic package . . . . .		ST-103
Unchecked_Conversion function		
Unchecked_Conversion.Unchecked_Conversion . . . . .	PT-212, PT-219, PT-225	
Unchecked_Conversion generic function . . . . .		PT-209
Unchecked_Conversion_Package generic package		
Unchecked_Conversions.Unchecked_Conversion_Package . . . . .	PT-219, PT-225	
Unchecked_Conversions package . . . . .		PT-219
Unchecked_Deallocation generic procedure . . . . .		PT-241
Allows_Deallocation generic function . . . . .	PT-1	
Allows_Deallocation.Allows_Deallocation function . . . . .	PT-2	
Unchecked_Deallocation procedure		
Unchecked_Deallocation.Unchecked_Deallocation . . . . .	PT-246	
[Unicode (All Worlds)] key		
Compilation.Demote procedure . . . . .	LM-141	
[Unicode (This World)] key		
Compilation.Demote procedure . . . . .	LM-141	
Uncomment procedure		
Editor.Region.Uncomment . . . . .	EI-49	
uncompressed output . . . . .		LM-178

Undefine procedure	
Concurrent_Map_Generic.Undefine . . . . .	PT-33
Map_Generic.Undefine . . . . .	PT-91
String_Map_Generic.Undefine . . . . .	ST-46
Undefined exception	
Concurrent_Map_Generic.Undefined . . . . .	PT-34
Eval function . . . . .	PT-15
Undefine procedure . . . . .	PT-33
Map_Generic.Undefined . . . . .	PT-92
Eval function . . . . .	PT-73
Undefine procedure . . . . .	PT-91
String_Map_Generic.Undefined . . . . .	ST-47
Eval function . . . . .	ST-30
Undefine procedure . . . . .	ST-46
Undelete procedure	
Library.Undelete . . . . .	LM-266
Compilation.Delete procedure . . . . .	LM-139
Delete renamed procedure . . . . .	LM-231
Underflow exception	
Stack_Generic.Underflow . . . . .	PT-143, PT-158
Pop procedure . . . . .	PT-154
Top function . . . . .	PT-157
underline	
next	
Editor.Cursor.Next procedure . . . . .	EI-19
previous	
Editor.Cursor.Previous procedure . . . . .	EI-19
remove	
Common.Clear_Underlining procedure . . . . .	EST-64
[Underlines Off] key	
Common.Clear_Underlining procedure . . . . .	EST-64
underscore ( _ )	
identifier character . . . . .	LM-9, PM-129, SJM-6, SMU-3
special character . . . . .	DEB-18, DEB-19, LM-10, LM-12, PM-132, SJM-8, SJM-9, SMU-7
Underscore character attribute . . . . .	DIO-102
undo a deletion	
Library.Undelete procedure . . . . .	LM-266
Undo procedure	
Common.Undo . . . . .	EST-47, EST-59, EST-99, PM-193
command images . . . . .	EST-50
Debugger . . . . .	DEB-3
Library package . . . . .	LM-206
Redo procedure . . . . .	EST-49, EST-93
Unfreeze procedure	
Library.Unfreeze . . . . .	LM-268
Freeze procedure . . . . .	LM-244
uninitialized, <i>see</i> Is_Nil, Nil	
Unique function	
String_Table.Unique . . . . .	ST-66
Unique_Index function	
String_Table.Unique_Index . . . . .	ST-67

unit	
create	
Library.Create_Unit renamed procedure . . . . .	LM-224
System.Storage_Unit constant . . . . .	PT-166
unit states	
Ada images . . . . .	EST-5
archived . . . . .	EST-5
coded . . . . .	EST-5
command images . . . . .	EST-46
false usages	
xref images . . . . .	EST-161
installed . . . . .	EST-5
source . . . . .	EST-5
Unit_Name subtype	
Activity.Unit_Name . . . . .	PM-170
Compilation.Unit_Name . . . . .	LM-165
Unit_Only enumeration	
Compilation.Promote_Scope type . . . . .	LM-161
Unit_State type	
Compilation.Unit_State . . . . .	LM-166
Units directory, setting up . . . . .	PM-23
Units procedure	
Check.Units . . . . .	PM-180
<UNITS> special value . . . . .	LM-129, LM-134, LM-199
Unknown_Key exception	
Window_Io.Raw.Unknown_Key . . . . .	DIO-187
Value function . . . . .	DIO-188
Unload procedure	
Tape.Unload . . . . .	SMU-293
unlock . . . . .	PM-136
unqualified name . . . . .	DEB-18, DEB-20
unreferenced	
Ada.Show_Unused procedure . . . . .	EST-41
Unregister procedure	
Queue.Unregister . . . . .	SMU-128
Remove procedure . . . . .	SMU-125
Unsupported_Error	
Io_Exceptions.Use_Error exception . . . . .	DIO-37, TIO-163
until	
Time_Uilities.Duration_Until function . . . . .	PT-185
Time_Uilities.Duration_Until_Next function . . . . .	PT-186
unused	
Ada.Show_Unused procedure . . . . .	EST-41
Up procedure	
Editor.Cursor.Up . . . . .	EI-17, EI-20
Editor.Image.Up . . . . .	EI-25, EI-26
Window_Scroll_Overlap session switch . . . . .	SJM-247
Editor.Screen.Up . . . . .	EI-54
up time	
System_Uilities.System_Up_Time function . . . . .	SMU-266

update	
images	EST-59
joined objects	PM-40
preventing automatic	PM-42
Update procedure	
Links.Update	LM-302
Replace procedure	LM-298
Source_Name subtype	LM-300
Update_Cdb procedure	
Cmvc_Maintenance.Update_Cdb	PM-108, PM-358
Cmvc_Maintenance.Make_Primary procedure	PM-351
Update_Time enumeration	
Library.Field type	LM-240
Updater enumeration	
Library.Field type	LM-240
Upper_Case constant	
Io.Upper_Case	TIO-104
Upper_Case function	
String_Uutilities.Upper_Case	ST-89
Upper_Case procedure	
Editor.Char.Upper_Case	EI-15
Editor.Line.Upper_Case	EI-35
Editor.Region.Upper_Case	EI-49
Editor.Word.Upper_Case	EI-72
String_Uutilities.Upper_Case	ST-90
uppercase, <i>see also</i> Capitalize	
usage	
Ada.Show_Usage procedure	EST-39
Use_Default_Wsl_Limits procedure	
Scheduler.Use_Default_Wsl_Limits	SMU-188
Set_Wsl_Limits procedure	SMU-180
Use_Error enumeration	
Profile.Log_Output_File type	SJM-113
Use_Error exception	
Access_List package	
Read constant	LM-43
Write constant	LM-48
Access_List_Tools package	
Read constant	LM-80
Write constant	LM-85
Direct_Io generic package	DIO-7
Create procedure	DIO-11
Delete procedure	DIO-12
Element_Type type	DIO-13
Open procedure	DIO-22
Reset procedure	DIO-25
Write procedure	DIO-28
Io package	TIO-2, TIO-3, TIO-5
Append procedure	TIO-13
Create procedure	TIO-21
Delete procedure	TIO-25
Open procedure	TIO-64
Reset procedure	TIO-80
Set_Error procedure	TIO-88



Use_Error exception (continued)	
Io package (continued)	
Set_Input procedure . . . . .	TIO-90
Set_Line_Length procedure . . . . .	TIO-93
Set_Output procedure . . . . .	TIO-95
Set_Page_Length procedure . . . . .	TIO-96
Io_Exceptions.Use_Error . . . . .	DIO-2, DIO-3, DIO-5, DIO-37, TIO-163
Polymorphic_Sequential_Io package	
Append procedure . . . . .	DIO-40
Create procedure . . . . .	DIO-43
Delete procedure . . . . .	DIO-44
Open procedure . . . . .	DIO-52
Reset procedure . . . . .	DIO-53
Polymorphic_Sequential_Io.Operations package	
Write procedure . . . . .	DIO-58
Sequential_Io package . . . . .	DIO-61
Create procedure . . . . .	DIO-64
Delete procedure . . . . .	DIO-65
Open procedure . . . . .	DIO-75
Reset procedure . . . . .	DIO-77
Write procedure . . . . .	DIO-78
Text_Io package	
Create procedure . . . . .	TIO-170
Delete procedure . . . . .	TIO-173
Open procedure . . . . .	TIO-194
Reset procedure . . . . .	TIO-202
Set_Line_Length procedure . . . . .	TIO-208
Set_Page_Length procedure . . . . .	TIO-210
Use_Output enumeration	
Profile.Log_Output_File type . . . . .	SJM-113
Use_Standard_Error enumeration	
Profile.Log_Output_File type . . . . .	SJM-113
Use_Standard_Output enumeration	
Profile.Log_Output_File type . . . . .	SJM-113
Used_By procedure	
Xref.Used_By . . . . .	LM-342
user . . . . .	SMU-197
access control . . . . .	LM-19, SMU-53
change job attribute	
Scheduler.Set_Job_Attribute procedure . . . . .	SMU-179
create	
Operator.Create_User procedure . . . . .	SMU-64
delete	
Operator.Delete_User procedure . . . . .	SMU-67
group membership	
Operator.Add_To_Group procedure . . . . .	SMU-56
Operator.Display_Group procedure . . . . .	SMU-70
Operator.Remove_From_Group procedure . . . . .	SMU-82
home library	
System_Uilities.Home_Library function . . . . .	SMU-223
library	
What.Home_Library procedure . . . . .	SJM-259
login	
Operator.Get_Login_Limit function . . . . .	SMU-77
Operator.Limit_Login procedure . . . . .	SMU-80
logoff	
Operator.Force_Logoff procedure . . . . .	SMU-75
name	
System_Uilities.User_Name function . . . . .	SMU-275

user ( <i>continued</i> )	
password	
Operator.Change_Password procedure . . . . .	SMU-60
who	
What.Users procedure . . . . .	SJM-271
world restored . . . . .	LM-89
User class . . . . .	LM-15, SJM-12
User function	
System_Uilities.User . . . . .	SMU-274
user groups, <i>see</i> Group	
User object manager . . . . .	SMU-11, SMU-58
user-defined	
groups . . . . .	SMU-55
messages . . . . .	LM-6, SJM-4
User_Break procedure	
Debug_Tools.User_Break . . . . .	DEB-16, DEB-185
User_Name function	
System_Uilities.User_Name . . . . .	SMU-275
username groups . . . . .	SMU-54
Username library switch . . . . .	LM-314
Username session switch . . . . .	SJM-246
Users procedure	
What.Users . . . . .	SJM-271
Uses procedure	
Xref.Uses . . . . .	LM-347
utilities	
File_Uilities package . . . . .	LM-169
String_Uilities package . . . . .	ST-69
System_Uilities package . . . . .	SMU-197
Time_Uilities package . . . . .	PT-175

V

validity	
Access_List_Tools.Check_Validity procedure . . . . .	LM-62
value	
Debug.Set_Value procedure . . . . .	DEB-114
<i>see also</i> Eval	
value delimiters . . . . .	LM-18, SJM-15, SMU-8
colon equals (:=) . . . . .	LM-18, SJM-15, SMU-8
equals (=) . . . . .	LM-18, SJM-15, SMU-8
equals/greater than (=>) . . . . .	LM-18, SJM-15, SMU-8
value error, <i>see</i> Input_Value_Error, Output_Value_Error	
Value function	
Bounded_String.Value . . . . .	ST-22
Concurrent_Map_Generic.Value . . . . .	PT-35
Init procedure . . . . .	PT-19
Iterator type . . . . .	PT-24
List_Generic.Value . . . . .	PT-66
Init procedure . . . . .	PT-54
Iterator type . . . . .	PT-57

Value function (*continued*)

Map_Generic.Value . . . . .	PT-93
Init procedure . . . . .	PT-77
Iterator type . . . . .	PT-82
Profile.Value . . . . .	SJM-167
Queue_Generic.Value . . . . .	PT-110
Init procedure . . . . .	PT-102
Iterator type . . . . .	PT-106
Set_Generic.Value . . . . .	PT-125
Init procedure . . . . .	PT-117
Iterator type . . . . .	PT-121
Stack_Generic.Value . . . . .	PT-159
Init procedure . . . . .	PT-149
Iterator type . . . . .	PT-151
String_Map_Generic.Value . . . . .	ST-48
Init procedure . . . . .	ST-33
Iterator type . . . . .	ST-38
String_Table.Value . . . . .	ST-68
Init procedure . . . . .	ST-56
Iterator type . . . . .	ST-59
System_Uilities.Value . . . . .	SMU-276, SMU-277, SMU-278
Done function . . . . .	SMU-210
Init procedure . . . . .	SMU-225, SMU-226, SMU-227
Job_Iterator type . . . . .	SMU-232
Next procedure . . . . .	SMU-241
Session_Iterator type . . . . .	SMU-259
Terminal_Iterator type . . . . .	SMU-271
Time_Uilities.Value . . . . .	PT-205
Unbounded_String.Value . . . . .	ST-124
Window_Io.Raw.Value . . . . .	DIO-188
Value procedure	
Window_Io.Raw.Value . . . . .	DIO-190
Value_Copy enumeration	
Activity.Creation_Mode . . . . .	PM-146
Activity.Display procedure . . . . .	PM-148
Value_Image subtype	
Switches.Value_Image . . . . .	LM-337
Vanilla constant	
Window_Io.Vanilla . . . . .	DIO-170
variable-length strings	
Bounded_String package . . . . .	ST-1
Unbounded_String package . . . . .	ST-103
Variable_String type	
Bounded_String.Variable_String . . . . .	ST-1, ST-23
Unbounded_String.Variable_String . . . . .	ST-103, ST-125
venture . . . . .	PM-2, PM-15, PM-361
editing . . . . .	PM-385, PM-417
images . . . . .	EST-1
policy switch . . . . .	PM-361
Venture subclass . . . . .	LM-17, SJM-15
Venture_Editor package	
Work_Order.Venture_Editor . . . . .	PM-417
Venture_Policy_Switch type	
Work_Order.Venture_Policy_Switch . . . . .	PM-400

Verbose renamed function	
Profile.Verbose . . . . .	SJM-168
<VERBOSE> special value . . . . .	SJM-76
Verbose_Format constant	
Library.Verbose_Format . . . . .	LM-270
Verbose_List renamed procedure	
Library.Verbose_List . . . . .	LM-3, LM-271
version . . . . .	EST-58, PM-408
Ada images . . . . .	EST-8
attributes . . . . .	LM-14, SJM-12
-n . . . . .	LM-14, SJM-12
All . . . . .	LM-14, SJM-12
Any . . . . .	LM-14, SJM-12
Max . . . . .	LM-14, SJM-12
Min . . . . .	LM-14, SJM-12
n . . . . .	LM-14, SJM-12
command images . . . . .	EST-46
control . . . . .	PM-3
defined . . . . .	PM-1, PM-6
error, <i>see</i> Nonexistent_Version_Error	
Library.Default_Keep_Versions constant . . . . .	LM-229
pathname	
System_Uilities.Image function . . . . .	SMU-224
print	
Queue.Print_Version procedure . . . . .	SMU-118
text images . . . . .	EST-139
Version enumeration	
Library.Field type . . . . .	LM-240
Version procedure	
What.Version . . . . .	SJM-274
Version subtype	
System_Uilities.Version . . . . .	SMU-279
vertical bar ( ) symbol . . . . .	LM-18, SJM-16, SMU-9
vertical layout . . . . .	DIO-93, DIO-94
view . . . . .	PM-1, PM-3, PM-8
building from a configuration object . . . . .	PM-50
client . . . . .	PM-10
code . . . . .	PM-16, PM-30, PM-262, PM-348
combined . . . . .	PM-53, PM-116
contents of tape	
Tape.Display_Tape procedure . . . . .	SMU-284
defining occurrences	
Common.Definition procedure . . . . .	EST-71
deleting . . . . .	PM-48
load . . . . .	PM-10, PM-11, PM-52, PM-136, PM-137
managing . . . . .	PM-48
names, coordinating level numbers . . . . .	PM-94
port settings	
System_Uilities package . . . . .	SMU-197
print queue	
Queue.Display procedure . . . . .	SMU-109
released . . . . .	PM-30
renaming . . . . .	PM-50
repairing damaged . . . . .	PM-50
spec . . . . .	PM-10, PM-11, PM-52, PM-137

view ( <i>continued</i> )	
system information	
System_Uilities package . . . . .	SMU-197
terminal settings	
System_Uilities package . . . . .	SMU-197
working . . . . .	PM-21, PM-224
View_Name subtype	
Activity.View_Name . . . . .	PM-171
View_Or_Activity_Name subtype	
Activity.View_Or_Activity_Name . . . . .	PM-172
View_Simple_Name subtype	
Activity.View_Simple_Name . . . . .	PM-173
Views procedure	
Check.Views . . . . .	PM-182
visible parts and bodies . . . . .	LM-13, SJM-11
visit other part of Ada unit	
Ada.Other_Part procedure . . . . .	EST-36
Visit procedure	
Activity.Visit . . . . .	PM-174
Links.Visit . . . . .	LM-304
Edit procedure . . . . .	LM-290
Switches.Visit . . . . .	LM-338
Edit procedure . . . . .	LM-329
Volume subtype	
Daemon.Volume . . . . .	SMU-47
Library.Volume . . . . .	LM-274

W

wait	
Scheduler.Disk_Waits function . . . . .	SMU-143
Wait enumeration	
Scheduler.Job_State type . . . . .	SMU-167
Wait state . . . . .	SMU-133
Wait_For procedure	
Program.Wait_For . . . . .	SJM-196
waiting at accept for entry call execution message (Debug.Task_Display) . . . . .	DEB-137
waiting at entry for accept execution message (Debug.Task_Display) . . . . .	DEB-137
waiting at select for entry call execution message (Debug.Task_Display) . . . . .	DEB-137
waiting at select-delay for entry call execution message (Debug.Task_Display) . . . . .	DEB-137
waiting at select-terminate for entry call execution message (Debug.Task_Display) . . . . .	DEB-137
waiting at select-terminate for entry call with dependents execution message (Debug.Task_Display) . . . . .	DEB-137
waiting at timed entry for accept execution message (Debug.Task_Display) . . . . .	DEB-137
waiting for child elaboration execution message (Debug.Task_Display) . . . . .	DEB-137
waiting for children execution message (Debug.Task_Display) . . . . .	DEB-137
waiting for delay execution message (Debug.Task_Display) . . . . .	DEB-137
waiting for parent elaboration execution message (Debug.Task_Display) . . . . .	DEB-137

waiting for task activation execution message (Debug.Task_Display) . . . . .	DEB-137
Warn renamed function	
Profile.Warn . . . . .	SJM-170
<WARN> special value . . . . .	SJM-76
warning	
Daemon.Get_Warning_Interval function . . . . .	SMU-22
Daemon.Threshold_Warnings procedure . . . . .	SMU-46
Operator.Shutdown_Warning procedure . . . . .	SMU-90
Warning enumeration	
Daemon.Condition_Class type . . . . .	SMU-16
Simple_Status.Condition_Class type . . . . .	PT-130
warning message . . . . .	SJM-3
Warning_Interval procedure	
Daemon.Warning_Interval . . . . .	SMU-48
Warning_Msg enumeration	
Profile.Msg_Kind type . . . . .	SJM-119
<WARNINGS> special value . . . . .	SJM-76
week	
Time_Uilities.Day_Of_Week function . . . . .	PT-183
Weekday type	
Time_Uilities.Weekday . . . . .	PT-206
Image function . . . . .	PT-190
Weekly client . . . . .	SMU-12
[What Load] key	
What.Load procedure . . . . .	SJM-264
[What Locks] key	
What.Lock procedure . . . . .	SJM-266
[What Object] key	
What.Object procedure . . . . .	SJM-268
What package . . . . .	SJM-253
[What Time] key	
What.Time procedure . . . . .	SJM-270
[What Users] key	
What.Users procedure . . . . .	SJM-271
who	
What.Users procedure . . . . .	SJM-271
width	
default	
Io.Enumeration_Io.Default_Width constant . . . . .	TIO-111
Io.Integer_Io.Default_Width constant . . . . .	TIO-145
Profile.Default_Width constant . . . . .	SJM-93
Text_Io.Enumeration_Io.Default_Width constant . . . . .	TIO-219
Text_Io.Integer_Io.Default_Width constant . . . . .	TIO-253
set	
Profile.Set_Default_Width procedure . . . . .	SJM-153
Profile.Set_Width procedure . . . . .	SJM-164
Width function	
Profile.Width . . . . .	SJM-172
Log.Put_Line procedure . . . . .	SJM-49
Width profile attribute . . . . .	SJM-73

Wildcard_Iterator generic procedure	
Io.Wildcard_Iterator . . . . .	TIO-105
Wildcard_Iterator procedure	
Io.Wildcard_Iterator . . . . .	TIO-8, TIO-108
Note_Error generic formal procedure . . . . .	TIO-106
Process generic formal procedure . . . . .	TIO-107
wildcards . . . . .	DIO-5, LM-7, PM-127, PM-129, SMU-1, SMU-3, TIO-4
file utilities	
asterisk (*) . . . . .	LM-172, LM-181, LM-184, LM-187
backslash (\) . . . . .	LM-172, LM-181, LM-184, LM-187
brackets ([ ]) . . . . .	LM-172, LM-181, LM-184, LM-187
caret (^) . . . . .	LM-172, LM-181, LM-184, LM-187
dollar sign (\$) . . . . .	LM-172, LM-181, LM-184, LM-187
left brace ({) . . . . .	LM-172, LM-181, LM-184, LM-187
percent (%) . . . . .	LM-172, LM-181, LM-184, LM-187
question mark (?) . . . . .	LM-172, LM-181, LM-184, LM-187
right brace (}) . . . . .	LM-172, LM-181, LM-184, LM-187
library . . . . .	LM-8, SJM-5, SJM-6
at sign (@) . . . . .	LM-8, LM-9, LM-109, LM-113, LM-297, LM-301, PM-129, SJM-6, SMU-3
double question mark (??) . . . . .	LM-8, LM-9, PM-129, SJM-6, SJM-7, SMU-3, SMU-4
pound sign (#) . . . . .	LM-8, LM-109, LM-113, LM-297, LM-301, PM-129, SJM-6, SMU-3
question mark (?) . . . . .	LM-8, LM-9, LM-109, LM-113, LM-297, LM-301, PM-129, SJM-6, SJM-7, SMU-3, SMU-4
. . . . .	
window . . . . .	DIO-2, DIO-79, DIO-81, TIO-2
active	
Editor.Window.Directory procedure . . . . .	EI-66
attributes	
Window_Io.Job_Number function . . . . .	DIO-141
Window_Io.Job_Time function . . . . .	DIO-142
Window_Io.Last_Line function . . . . .	DIO-143
Window_Io.Line_Image function . . . . .	DIO-144
Window_Io.Line_Length function . . . . .	DIO-145
Window_Io.Line_Number subtype . . . . .	DIO-146
Window_Io.Read_Banner function . . . . .	DIO-161
Window_Io.Report_Cursor procedure . . . . .	DIO-163
Window_Io.Report_Location procedure . . . . .	DIO-164
Window_Io.Report-Origin procedure . . . . .	DIO-165
Window_Io.Report_Size procedure . . . . .	DIO-166
Window_Io.Set_Banner procedure . . . . .	DIO-168
banner	
# symbol . . . . .	PM-135, PM-404
* symbol . . . . .	PM-210, PM-232, PM-244, PM-292
= symbol . . . . .	PM-135
bottom of	
Editor.Window.End_Of procedure . . . . .	EI-66
change to next higher state	
Editor.Window.Promote procedure . . . . .	EI-67
change to next lower state	
Editor.Window.Demote procedure . . . . .	EI-65
Editor window	
Io.Current_Error function . . . . .	TIO-22
Io.Current_Input function . . . . .	TIO-23
Io.Current_Output function . . . . .	TIO-24
Io.Standard_Error function . . . . .	TIO-99
Io.Standard_Input function . . . . .	TIO-100
Io.Standard_Output function . . . . .	TIO-101
Text_Io.Current_Input function . . . . .	TIO-171
Text_Io.Current_Output function . . . . .	TIO-172
Text_Io.Standard_Input function . . . . .	TIO-213
Text_Io.Standard_Output function . . . . .	TIO-214

window (continued)

expand	
Editor.Window.Expand procedure . . . . .	EI-66
Editor.Window.Join procedure . . . . .	EI-66
image type . . . . .	EI-6
images . . . . .	EST-1
committing . . . . .	EST-155
demoting . . . . .	EST-155
image structure . . . . .	EST-153
key concepts . . . . .	EST-154
promoting . . . . .	EST-155
refreshing . . . . .	EST-156
releasing images . . . . .	EST-156
traversing . . . . .	EST-155
keep on screen	
Editor.Window.Promote procedure . . . . .	EI-67
management . . . . .	EI-6
Editor.Window package . . . . .	EI-63
move between windows	
Editor.Window.Next procedure . . . . .	EI-67
Editor.Window.Previous procedure . . . . .	EI-67
move to next	
Editor.Window.Child procedure . . . . .	EI-64
move to previous	
Editor.Window.Parent procedure . . . . .	EI-67
remove	
Editor.Window.Delete procedure . . . . .	EI-65
restore frame size	
Editor.Window.Focus procedure . . . . .	EI-66
set number of work windows	
Editor.Window.Frames procedure . . . . .	EI-66
states . . . . .	EI-63
swap locations	
Editor.Window.Transpose procedure . . . . .	EI-68
top of	
Editor.Window.Beginning_Of procedure . . . . .	EI-64
two views	
Editor.Window.Copy procedure . . . . .	EI-65
utilities . . . . .	DIO-86
Window Directory . . . . .	EI-6, EST-153
commands from package Common . . . . .	EST-156
committing . . . . .	EST-155
demoting . . . . .	EST-155
designation . . . . .	EST-154
Editor.Image.Find procedure . . . . .	EI-26
Editor.Window package . . . . .	EI-63
Editor.Window.Directory procedure . . . . .	EI-66
image structure . . . . .	EST-153
key concepts . . . . .	EST-154
promoting . . . . .	EST-155
refreshing . . . . .	EST-156
releasing images . . . . .	EST-156
traversing . . . . .	EST-155
Window package	
Editor.Window . . . . .	EI-6, EI-63
[Window] - [Definition] key combination	
Editor.Window.Directory procedure . . . . .	EI-66
[Window] - [Demote] key combination	
Editor.Window.Demote procedure . . . . .	EI-65



[Window] - [Promote] key combination	
Editor.Window.Promote procedure . . . . .	EI-67
Window_Command_Size session switch . . . . .	SJM-247
Window_Frames session switch . . . . .	SJM-247
Window_Frames_Startup session switch . . . . .	SJM-247
Window_Have_Sides session switch . . . . .	SJM-247
Window_Io package . . . . .	DIO-79
Window_Is_Staggered session switch . . . . .	SJM-247
Window_Message_Life session switch . . . . .	SJM-247
Window_Message_Size session switch . . . . .	SJM-247
Window_Scroll_Overlap session switch . . . . .	SJM-247
Window_Shift_Overlap session switch . . . . .	SJM-248
Withdraw procedure	
Ada.Withdraw . . . . .	EST-43, PM-35, PM-36
[Withdraw Unit] key	
Ada.Withdraw procedure . . . . .	EST-43
withdrawing objects . . . . .	PM-35
withdrawn items . . . . .	LM-196
withheld	
job . . . . .	SMU-133, SMU-136
task	
Scheduler.Get_Withheld_Task_Load procedure . . . . .	SMU-158
Word package	
Editor.Word . . . . .	EI-69
Word_Breaks session switch . . . . .	SJM-248
Editor.Word.Breaks procedure . . . . .	EI-70
text images . . . . .	EST-138
Word_Size constant	
System.Word_Size . . . . .	PT-167
words	
beginning of	
Editor.Word.Beginning_Of procedure . . . . .	EI-69
case conversion	
Editor.Word.Capitalize procedure . . . . .	EI-70
Editor.Word.Lower_Case procedure . . . . .	EI-71
Editor.Word.Upper_Case procedure . . . . .	EI-72
deletion	
Editor.Word.Delete procedure . . . . .	EI-70
Editor.Word.Delete_Backward procedure . . . . .	EI-70
Editor.Word.Delete_Forward procedure . . . . .	EI-70
editing operations	
Editor.Word package . . . . .	EI-69
end of	
Editor.Word.End_Of procedure . . . . .	EI-71
next	
Editor.Word.Next procedure . . . . .	EI-71
previous	
Editor.Word.Previous procedure . . . . .	EI-71
redefine break characters	
Editor.Word.Breaks procedure . . . . .	EI-70

words ( <i>continued</i> )	
swap locations	
Editor.Word.Transpose procedure	EI-71
work-list images	EST-1
work order	PM-2, PM-15, PM-361
editing	PM-403
images	EST-2
list	PM-2, PM-15, PM-361
editing	PM-384
Work subclass	LM-17, SJM-15
Work_List subclass	LM-17, SJM-15
Work_Order package	PM-361
Work_Order_Implementation package	
Implementation.Work_Order_Implementation	
Word_Order.Venture_Policy_Switch type	PM-400
working	
library	PM-7
set limits (WSL)	
Scheduler.Get_Wsl_Limits procedure	SMU-159
Scheduler.Set_Wsl_Limits procedure	SMU-180
Scheduler.Use_Default_Wsl_Limits procedure	SMU-188
set management	SMU-139
view	PM-8, PM-21, PM-188, PM-224
predefined library characteristics	PM-22
putting objects under CMVC	PM-25
releasing configurations	PM-30
Working_Set_Size function	
Scheduler.Working_Set_Size	SMU-189
Job_Descriptor type	SMU-161
world	LM-2, SJM-2
access classes	LM-21
Access_List.Create constant	LM-34
Access_List.Delete constant	LM-37
Access_List.Owner constant	LM-42
Access_List.Read constant	LM-43
Access_List_Tools.Create constant	LM-64
Access_List_Tools.Delete constant	LM-65
Access_List_Tools.Owner constant	LM-79
Access_List_Tools.Read constant	LM-80
all	
Compilation.All_Worlds constant	LM-132
create	
Library.Create_World renamed procedure	LM-226
enclosing	DEB-19, LM-11, PM-132, SJM-9, SMU-6
Library.Enclosing_World procedure	LM-235
links	LM-275
restore	LM-89
same	
Compilation.Same_World constant	LM-163
Compilation.Same_Worlds constant	LM-164
World enumeration	
Library.Kind type	LM-247
World subclass	LM-15, SJM-13
World_Name subtype	
Links.World_Name	LM-305

<WORLDS> special value . . . . .	LM-129, LM-134, LM-163, LM-164, LM-199
Wrap_Indentation library switch . . . . .	LM-314
write	
access . . . . .	LM-21
Boolean value to Message window	
Io.Echo procedure . . . . .	TIO-31
character to Message window	
Io.Echo procedure . . . . .	TIO-26
files with different types of data	
Polymorphic_Sequential_Io package . . . . .	DIO-39
floating-point value to Message window	
Io.Echo procedure . . . . .	TIO-29
integer value to Message window	
Io.Echo procedure . . . . .	TIO-28
lock . . . . .	EST-59
open to	
Out_File constant . . . . .	TIO-65
string to Message window	
Io.Echo procedure . . . . .	TIO-27
string to Message window/advance line	
Io.Echo_Line procedure . . . . .	TIO-32
to tapes	
Tape package . . . . .	SMU-283
<i>see also</i> Put, Put_Line	
Write constant	
Access_List.Write . . . . .	LM-48
Access_List_Tools.Write . . . . .	LM-85
Write procedure	
Activity.Write . . . . .	PM-175
Direct_Io.Write . . . . .	DIO-28
Polymorphic_Sequential_Io.Operations.Write . . . . .	DIO-58
Sequential_Io.Write . . . . .	DIO-78
Switches.Write . . . . .	LM-339
Tape.Write . . . . .	SMU-283, SMU-294
write-only access	
Direct_Io.File_Mode type . . . . .	DIO-15
Io.File_Mode subtype . . . . .	TIO-37
Polymorphic_Sequential_Io.File_Mode type . . . . .	DIO-46
Sequential_Io.File_Mode type . . . . .	DIO-68
Text_Io.File_Mode type . . . . .	TIO-178
Window_Io.File_Mode type . . . . .	DIO-120
Write_File procedure	
Common.Write_File . . . . .	EST-100
command images . . . . .	EST-50
Debugger . . . . .	DEB-3, DEB-5
text images . . . . .	EST-142
Write_Mt procedure	
Tape.Write_Mt . . . . .	SMU-298
Write_To_Read_Only_Page_Error	
Io_Exceptions.Device_Error exception . . . . .	DIO-31, TIO-157
writer, report	
Table_Formatter package . . . . .	ST-91

Wsl	
Scheduler.Get_Wsl_Limits procedure	SMU-159
Scheduler.Set_Wsl_Limits procedure	SMU-180
Scheduler.Use_Default_Wsl_Limits procedure	SMU-188

Xecute procedure	
Debug.Xecute	DEB-148
Execute procedure	DEB-61

Xon/Xoff	
System_Uilities.Receive_Xon_Xoff_Bytes function	SMU-254
System_Uilities.Receive_Xon_Xoff_Characters function	SMU-255
Terminal.Set_Receive_Xon_Xoff_Bytes procedure	SMU-320
Terminal.Set_Receive_Xon_Xoff_Characters procedure	SMU-321
Terminal.Set_Xon_Xoff_Bytes procedure	SMU-324
Terminal.Set_Xon_Xoff_Characters procedure	SMU-325

Xon_Xoff_Bytes function	
System_Uilities.Xon_Xoff_Bytes	SMU-280

Xon_Xoff_Characters function	
System_Uilities.Xon_Xoff_Characters	SMU-281

Xref flag definitions	LM-342
-----------------------	--------

xref images	EST-2, EST-159
commands from package Common	EST-161
designation	EST-160
elision	EST-160
expansion	EST-160
key concepts	EST-160
structure	EST-159
unit states	
false usages	EST-161

Xref package	LM-341
--------------	--------

Y

Year function	
Calendar.Year	PT-7

Year_Month_Day enumeration	
Time_Uilities.Date_Format type	PT-180

Year_Number subtype	
Calendar.Year_Number	PT-7

Years type	
Time_Uilities.Years	PT-207

Yr_Mn_Dy enumeration	
Profile.Log_Prefix type	SJM-115

# RATIONAL

## READER'S COMMENTS

**Note:** This form is for documentation comments only. You can also submit problem reports and comments electronically by using the SIMS problem-reporting system. If you use SIMS to submit documentation comments, please indicate the manual name, book name, and page number.

Did you find this book understandable, usable, and well organized? Please comment and list any suggestions for improvement.

---

---

---

---

If you found errors in this book, please specify the error and the page number. If you prefer, attach a photocopy with the error marked.

---

---

---

Indicate any additions or changes you would like to see in the index.

---

---

---

How much experience have you had with the Rational Environment?

6 months or less \_\_\_\_\_ 1 year \_\_\_\_\_ 3 years or more \_\_\_\_\_

How much experience have you had with the Ada programming language?

6 months or less \_\_\_\_\_ 1 year \_\_\_\_\_ 3 years or more \_\_\_\_\_

Name (optional) \_\_\_\_\_ Date \_\_\_\_\_

Company \_\_\_\_\_

Address \_\_\_\_\_

City \_\_\_\_\_ State \_\_\_\_\_ ZIP Code \_\_\_\_\_

Please return this form to:

**Publications Department  
Rational  
3320 Scott Boulevard  
Santa Clara, CA 95054-3197**

FINAL

COMMENTS

and other related items. You can also submit problem reports and  
other information by using the BMB problem-reporting system. If you see BMB  
in the comments, it indicates the manual name, book name, and page number.

to look under the book name, and well organized. Please comment and let us  
know if you have any suggestions.

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

in this area. If you find an error, please report it to the editor.

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

If you have any suggestions, please let us know.

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

If you have any suggestions, please let us know.

\_\_\_\_\_ years or more \_\_\_\_\_ year \_\_\_\_\_ years

If you have any suggestions, please let us know.

\_\_\_\_\_ years or more \_\_\_\_\_ year \_\_\_\_\_ years

\_\_\_\_\_ Date

\_\_\_\_\_ Date

Publication Department  
Federal  
330 South Broadway  
Fort Collins, CO 80521

# RATIONAL

## READER'S COMMENTS

**Note:** This form is for documentation comments only. You can also submit problem reports and comments electronically by using the SIMS problem-reporting system. If you use SIMS to submit documentation comments, please indicate the manual name, book name, and page number.

Did you find this book understandable, usable, and well organized? Please comment and list any suggestions for improvement.

---

---

---

---

---

If you found errors in this book, please specify the error and the page number. If you prefer, attach a photocopy with the error marked.

---

---

---

---

Indicate any additions or changes you would like to see in the index.

---

---

---

---

How much experience have you had with the Rational Environment?

6 months or less \_\_\_\_\_ 1 year \_\_\_\_\_ 3 years or more \_\_\_\_\_

How much experience have you had with the Ada programming language?

6 months or less \_\_\_\_\_ 1 year \_\_\_\_\_ 3 years or more \_\_\_\_\_

Name (optional) \_\_\_\_\_ Date \_\_\_\_\_

Company \_\_\_\_\_

Address \_\_\_\_\_

City \_\_\_\_\_ State \_\_\_\_\_ ZIP Code \_\_\_\_\_

Please return this form to:

Publications Department  
Rational  
3320 Scott Boulevard  
Santa Clara, CA 95054-3197

