

 **Sablime[®] v6.0u1**
Administrator's Guide



Copyright© 2003 Lucent Technologies
All Rights Reserved

Sablime is a registered trademark and the six delta design is a trademark of Lucent Technologies.

Contents

1	Introduction	1-1
	n Purpose	1-1
	n Scope	1-1
	n Intended Audience	1-1
	n Organization	1-2
	n Conventions	1-3
	n Icons	1-4
2	Installing Sablime	2-1
	n Preparing to Install Sablime	2-2
	n Running the Installation Script	2-6
	n Setting Up the Sablime Environment	2-26
	n Using Sablime On Multiple Machines	2-37
	n Configuring Sablime for External MR Communications	2-44
	n Installing the End User Web Interface	2-62
	n Pre- and Post-Trigger Routines	2-63
3	Customizing the User Interface	3-1
	n Essential Product Customization	3-2
	n Customizing Commands and Command Screens	3-16
	n Customizing Pop-Up Selection Windows	3-29
	n Customizing Templates	3-38
	n Defining New Fields	3-40

Contents

4	Other Administrative Procedures	4-1
n	Managing a Product	4-1
n	Managing a Generic	4-4
n	Managing Files and Directories	4-12
n	Managing the Databases	4-27
n	Changing Machines	4-35
n	Customizing Sablime Mail	4-36
n	Customizing Command Executors and Email Recipients	4-39

5	The Sablime Databases	5-1
n	Relations, Tuple Files, and Records	5-1
n	Commands and Relations	5-2
n	Global Database	5-5
n	Active/Inactive Databases	5-14
n	Source Database	5-62

6	The Administrative Commands	6-1
n	Overview	6-1
n	Command Descriptions	6-3
n	Host-Only Commands	6-4
n	addgen Command	6-5
n	closegen Command	6-10
n	dbcross Command	6-12
n	dbdelta Command	6-13
n	dbedit Command	6-14
n	dbhash Command	16
n	dbstart Command	6-17

Contents

n dbstop Command	6-18
n dbxcross Command	6-20
n ftd Command	6-21
n hotline.ck Script	6-26
n initsab Script	6-28
n mhist/nohist Commands	6-29
n mmail/nomail Commands	6-31
n mredit Command	6-32
n mrgedit Command	6-38
n mtrace/notrace Commands	6-45
n mvgen Script	6-47
n newgen Script	6-48
n primsdb Command	6-49
n screate Command	6-50
n setperm Command	6-54
n setrel Command	6-55
n ADM Subcommand (setrel)	6-56
n CAS Subcommand (setrel)	6-59
n CP Subcommand (setrel)	6-60
n CRIT Subcommand (setrel)	6-62
n ES Subcommand (setrel)	6-65
n PR Subcommand (setrel)	6-68
n PRX Subcommand (setrel)	6-69
n source Command	6-71
n spacecheck Script	6-74
n sreport Command	6-75

Contents

A	Sablime-Level Reserved Groups	A-1
----------	--------------------------------------	-----

B	Product-Level Reserved Groups	B-1
----------	--------------------------------------	-----

C	Initialization Program Diagnostic Messages	C-1
n	Organization	C-1
n	IProgram Diagnostic Messages	C-2
n	Uid Diagnostic Messages	C-15
n	Other Diagnostic Messages	C-17

D	Audit Program Messages	D-1
n	Messages from dbcross	D-1
n	Messages from dbxcross	D-13
n	Messages from dbdelta	D-15
n	Messages from hotline.ck	D-18
n	Messages from spacecheck	D-23

E	Sablime Installation Worksheets	E-1
n	Worksheet 1: Pre-Installation Checklist	E-2
n	Worksheet 2: Installation Information (initsab)	E-3
n	Worksheet 3: Installation Information (sablime.sh and xsablime.sh)	E-10

Contents

- n Worksheet 4:
primsdb ChecklistE-11
- n Worksheet 5:
Installation Information (primsdb)E-12
- n Worksheet 6:
Products ListE-13
- n Worksheet 7:
Product ProfileE-14
- n Worksheet 8:
Host AssignmentsE-15
- n Worksheet 9:
Directory StructureE-16

GL

Glossary

GL-1

Contents

Figures

1 Introduction

2 Installing Sablime

2-1	Sample Directory Structure	2-7
2-2	Sample Directory Structure File v5.0	2-8
2-3	initsab Welcome Screen	2-10
2-4	initsab List of Prompts	2-11
2-5	initsab Summary of Responses	2-16
2-6	initsab Installation Complete Message	2-25
2-7	Multiple Machines for Sablime	2-38
2-8	Sample TCP/IP Setup	2-43
2-9	Environment for IPC Example 1	2-45
2-10	IPC Example 1: setrel ES Screen for Product A	2-47
2-11	IPC Example 1: setrel ES Screen for Product B	2-48
2-12	Environment for IPC Example 2	2-49
2-13	IPC Example 2: setrel ES Screen for Product C	2-50
2-14	IPC Example 2: setrel ES Screen for Product D	2-51
2-15	Environment for IPC Example 3	2-52
2-16	IPC Example 3: setrel ES Screen for Product E	2-53
2-17	IPC Example 3: setrel ES Screen for Product F	2-54
2-18	Environment for TCP/IP Example	2-55
2-19	TCP/IP Example: setrel ES Screen for Product A	2-56
2-20	TCP/IP Example: setrel ES Screen for Product B	2-57
2-21	Environment for UUCP Example	2-58
2-22	UUCP Example : setrel ES Screen for Product A	2-59
2-23	UUCP Example: setrel ES Screen for Product B	2-60

Figures

3	Customizing the User Interface	
3-1	setgroup Command for _SYS	3-5
3-2	setgroup Command for _RELS	3-6
3-3	setgroup Command for _SITES	3-8
3-4	setgroup Command for _ICAT	3-9
3-5	setgroup Command for _LIBSUB	3-13
3-6	CAS Subcommand for lib/_LIBSUB	3-14
3-7	CAS subcommand for planets:Mars/marsmoons	3-15
3-8	ftd Screen for Generic Field in accept Command	3-28
3-9	Sablime-Level Reserved Group Menus	3-30
3-10	Product-Level Reserved Group Menus	3-30
3-11	setgroup Command for __CSEV	3-32
3-12	setgroup Command for __SEND	3-34
3-13	setgroup Command for __CSEV (comments)	3-35
3-14	setgroup Command for _ATYPE	3-37
3-15	Default create Screen	3-47
3-16	Default mrudf1 FTD Data for the create Command	3-48
3-17	Customized mrudf1 FTD Data for the create Command	3-49
3-18	Default mrudf2 FTD Data for the create Command	3-50
3-19	Customized mrudf2 FTD Data for the create Command	3-51
3-20	Customized create Command Screen	3-52
3-21	Default screen for query Command	3-53
3-22	Default mrudf1 FTD Data for the query Command	3-54
3-23	Customized mrudf1 FTD data for the query Command	3-55
3-24	Default mrudf2 FTD data for the query Command	3-56
3-25	Customized mrudf2 FTD data for the query Command	3-57
3-26	Customized query Command	3-58

Figures

4 Other Administrative Procedures

4-1	newgen Screen	4-5
4-2	mvgen Screen	4-10
4-3	Mail Consolidation Message	4-38

5 The Sablime Databases

5-1	Global Database Directories and Relations	5-5
5-2	Active Database Directories and Relations	5-14
5-3	Inactive Database Directories and Relations	5-15
5-4	Active Database DBLOCK Directory	5-16
5-5	Inactive Database DBLOCK Directory	5-17
5-6	Active Database FILES Directory	5-18
5-7	Inactive Database FILES Directory	5-19
5-8	Directory Structure for the Sablime Instance	5-62
5-9	Delta Specifiers	5-63

Figures

6 The Administrative Commands

A Sablime-Level Reserved Groups

B Product-Level Reserved Groups

C Initialization Program Diagnostic Messages

D Audit Program Messages

E Sablime Installation Worksheets

Tables

1 Introduction

2 Installing Sablime

2-1	Initial Space Requirements	2-2
2-2	Sablime Variables	2-27
2-3	Multi-Machine Error Messages	2-44

3 Customizing the User Interface

3-1	Fields, Commands, and Groups	3-3
3-2	Cascading Fields	3-12
3-3	The Fields of the FTD Record	3-18
3-4	Flags (Field 3)	3-19
3-5	Text (Field 4)	3-20
3-6	Values (Field 5)	3-21
3-7	HMI (Field 7)	3-23
3-8	Run-Time Expansion Keywords	3-26
3-9	Command Templates	3-38
3-10	Template File Names	3-39
3-11	User-Definable Fields	3-41

4 Other Administrative Procedures

4-1	MR Branches and Version Retrieved	4-21
4-2	The Audit Programs and their Functions	4-29
4-3	The Variables in spacecheck	4-34
4-4	Definition of Command Permissions Database Relation	4-39

Tables

4-5	Valid Executor Values	4-40
4-6	Valid Email Recipients	4-41
4-7	CP Relation Fields	4-41

5 The Sablime Databases

5-1	Commands and the Relations They Affect	5-2
5-2	Global Database Relations	5-7
5-3	ES Relation Fields	5-8
5-4	PR Relation Fields	5-9
5-5	PRX Relation Fields	5-10
5-6	PTS Relation Fields	5-11
5-7	TR Relation Fields	5-13
5-8	Relations vs. Commands	5-23
5-9	ADM Relation Fields	5-29
5-10	CAS Relation Fields	5-30
5-11	COM Relation Fields	5-31
5-12	CP Relation Fields	5-32
5-13	CRIT Relation Fields	5-33
5-14	DEP Relation Fields	5-34
5-15	EMG Relation Fields	5-35
5-16	EMR Relation Fields	5-36
5-17	FTD Relation Fields	5-38
5-18	FZ Relation Fields	5-40
5-19	G Relation Fields	5-41
5-20	GRP Relation Fields	5-42
5-21	GRPM Relation Fields	5-43
5-22	GS Relation Fields	5-44
5-23	GT Relation Fields	5-45
5-24	HC Relation Fields	5-46
5-25	MD Relation Fields	5-47
5-26	MG Relation Fields	5-48

Tables

5-27	MRG States and Corresponding System Codes	5-50
5-28	MR Relation Fields	5-52
5-29	MRS Relation Fields	5-53
5-30	MRX Relation Fields	5-54
5-31	MS Relation Fields	5-55
5-32	ORG Relation Fields	5-56
5-33	PDEP Relation Fields	5-57
5-34	PDI Relation Fields	5-58
5-35	SNAP Relation Fields	5-60
5-36	UMS Relation Fields	5-61

6 The Administrative Commands

6-1	The Sablime Administrative Commands	6-1
6-2	Command Availability	6-4
6-1	ADM Quick Commands	6-56
6-2	Executor Field Keywords	6-62
6-3	Email Recipient Field Keywords	6-62
6-4	Network Types	6-67
6-5	Machine Modes	6-69

A Sablime-Level Reserved Groups

A-1	Sablime-Level Reserved Groups	A-2
-----	-------------------------------	-----

B Product-Level Reserved Groups

B-1	Product-Level Reserved Groups	B-2
-----	-------------------------------	-----

Tables

C Initialization Program Diagnostic Messages

D Audit Program Messages

D-1	dbcross Messages	D-1
D-2	dbxcross Messages	D-13
D-3	dbdelta Messages	D-15
D-4	hotline.ck Messages	D-18
D-5	spacecheck Messages	D-23

E Sablime Installation Worksheets

Contents

1	Introduction	1
	n Purpose	1
	n Scope	1
	n Intended Audience	1
	n Organization	2
	n Conventions	3
	n Icons	4

Contents

Introduction

1

Purpose

This guide provides the information necessary to install, customize, and administer the Sablime Configuration Management System. It also describes the Sablime databases and contains manual pages for the administrative commands.

Scope

This issue of the *Administrator's Guide* applies to version 6.0 of Sablime.

Intended Audience

This guide is intended for anyone who has an administrative role in the Sablime Configuration Management System; Database Administrators (DBAs), Source Administrators (SAs), Modification Request Administrators (MRAs), and Generic Administrators (GAs). Each of these roles is briefly described below.

The Database Administrator is a person or group of people who owns and is responsible for the *sablime* login and the Sablime databases and commands. Typically, the Database Administrator installs and customizes Sablime for the project, sets up and executes the audit programs and resolves any problems they reveal, acts as a liaison between users and the hotline, answers users questions, and adds and maintains the project PTS IDs.

The Source Administrator is a person or group of people who controls the source database (SDB) and who is responsible for the maintenance of the product

directory structure and the administration of the files associated with a product. Typical tasks of the Source Administrator are; moving files, removing files, changing file names, changing file attributes, and making files common or uncommon.

The Modification Request Administrator decides whether a Modification Request (MR) should be killed, studied, deferred, or accepted for certain generics, and closes an MR after all the work it requests has been completed.

The Generic Administrator decides if and when a Modification Request in a Generic (MRG) is to be processed in the generic. The Generic Administrator also chooses an Assigned Developer to study the MRG or make changes in response to it.

Organization

This guide comprises an introduction (Chapter 1), installation information (Chapter 2), instructions on customization (Chapter 3), administrative procedures (Chapter 4), a description of the Sablime databases (Chapter 5), manual pages for the administrative commands (Chapter 6), five appendices (Appendix A – Appendix E), a Glossary, and an *Index*.

A summary of the contents of the chapters and appendices follows.

- n Chapter 1, *Introduction*, describes the purpose, scope, intended audience and scope of this guide. It also lists the typographical conventions and product safety labels used in the guide.
- n Chapter 2, *Installing Sablime*, describes installation requirements, how to establish an instance of Sablime, how to configure Sablime, how to set up the Sablime environment, multi-machine installation, and how to configure Sablime for external communication.
- n Chapter 3, *Customizing the User Interface*, describes essential customization, how to customize commands, command screens, pop-up selection windows, and templates, and how to define new fields.
- n Chapter 4, *Other Administrative Procedures*, tells how to manage a product, a generic, files, directories, and databases, and how to change machines and to customize Sablime mail.
- n Chapter 5, *The Sablime Databases*, describes the structure of the Sablime databases, and lists all the fields and files in the Global, Active, and Inactive Databases.
- n Chapter 6, *The Administrative Commands*, contains manual pages for all the administrative commands that may only be run by those with administrative privileges.

- n Appendix A, *Sablime-Level Reserved Groups*, lists all the Sablime-Level Reserved Groups, their members, functions, and default values.
- n Appendix B, *Product-Level Reserved Groups*, lists all the Product-Level Reserved Groups, their members, functions, and default values.
- n Appendix C, *Initialization Program Diagnostic Messages*, describes all the diagnostic messages and message numbers generated by the Sablime initialization program and suggests responses to them.
- n Appendix D, *Audit Program Messages*, lists all the messages generated by the Sablime audit programs and indicates the appropriate response to each.
- n Appendix E, *Sablime Installation Worksheets*, provides worksheets for writing information needed during installation and other operations. These sheets may be faxed to the Sablime hotline if you encounter a problem.
- n The *Glossary* contains definitions of terms used in this guide.
- n The *Index* is a comprehensive index that provides a quick and easy way of locating information.

Conventions

The following conventions are used throughout this guide:

- n Command Syntax
 - Words or symbols in **this type** are to be entered literally, exactly as shown.
 - Words in *italics* stand for variables for which you should make an appropriate substitution (usually a file name).
 - Square brackets ([]) indicate that the enclosed word (which can be a variable or the actual word to enter) is optional. If you use an option, do not enter the brackets.
 - A pipe symbol (|) indicates a choice of options, i.e., **y | n** indicates a choice between entering **y** or **n**.
 - Output generated in response to a command example is shown immediately following the command and is shown in this type.
- n File names and directory names are shown in this type. This type is also used when referencing executable programs, such as `sget`. In diagrams, directories may be indicated by a slash (/); executables may be indicated by an asterisk (*).
- n Computer output and file listings are shown in this type.
- n If a command extends over multiple lines, each line ends with a backslash (\). (One or more whitespace characters should either precede the backslash or start the next line.)

- n Input and output lines that wrap to the following line due to the margin constraints of this guide contain a backslash (\) at the end of each line.
- n UNIX system commands are referenced as *name*(*n*), where *name* is the name of the UNIX system command and *n* identifies the section of the UNIX system manual in which the manual page for the command is found.
- n When you are instructed to enter a series of characters, type the characters and then press the RETURN key. That key may be labeled RETURN or ENTER, or may show an arrow (↵).



NOTE:

All the menus and defaults listed in this guide are standard in the software as provided to your project; however, many of them may be customized. (See Chapter 3, *Customizing the User Interface*.) If you do make changes to the menus or defaults, remember to note the changes in this guide so that your documentation will accurately reflect your customized system.

Icons

This document uses two icons, *Caution* and *Note*.



CAUTION:

The Caution icon is used to mark activities that could affect the proper functioning of the Sablime system.



NOTE:

The Note icon is used to call particular attention to something in the text.

Contents

2	Installing Sablime	1
n	Preparing to Install Sablime	2
	Space Requirements	2
	Software and Other Requirements	3
	Downloading the Sablime Software	3
	The License Manager	4
	The License File	4
n	Running the Installation Script	5
	What the Installation Script Does	5
	Setting Up a Directory Structure File	6
	Gathering the Information You Will Need	8
	Running the Script	9
n	Setting Up the Sablime Environment	26
	Defining Environment Variables	27
	Customizing the dot sablime Script	33
	Making Changes to sablime.sh	34
	Making Changes to xsablime.sh	34
	Changing xsablime.sh and the sabVAR File When Communication Over TCP/IP is Needed	37
	Changing xsablime.sh When Adding a New Generic	38
	Creating Executable Versions of sablime.sh and xsablime.sh	39
n	Using Sablime On Multiple Machines	43
	Installing Sablime On Multiple Machines	45
	Notes for All Multi-Machine Configurations	45
	Using Sablime over NFS/RFS	45
	Using Sablime over TCP/IP	46
n	Configuring Sablime for External MR Communications	50
	Setting Up an Inter-Process Communication Link	51
	Setting Up a TCP/IP Communication Link	59
	Setting Up a UUCP Communication Link	63

Contents

n	Installing the Web Interface	67
n	Pre- and Post-Trigger Routines	68

This chapter provides detailed instructions for installing a Sablime instance or product on your machine and network. An instance, which may contain one or more products, consists of a Global Database (GDB) and, for each product in the instance, a set of three databases: an Active Database (ADB), an Inactive Database (IDB), and a Source Database (SDB).



NOTE:

You can use a name other than *sablime* to identify the login that owns the Sablime databases and executables; however, this guide and all the error messages generated by Sablime refer to this login as the *sablime* login.

The login that owns the databases and the Sablime executables is called the *sablime* login. A single *sablime* login can own more than one Global Database and thus be associated with more than one instance.

If you have any questions about installing Sablime or run into any problems while installing it, you can call the Sablime helpdesk for technical assistance.



NOTE:

Worksheet 1: Pre-Installation Checklist in Appendix E has a pre-installation checklist; please have your completed worksheet ready when you call the Sablime hotline.

Preparing to Install Sablime

Before you install Sablime, you must be sure that you have adequate disk space available and that you have access to appropriate software. Then you must download the Sablime software and make sure the license file is properly installed. The following sections deal with these matters.

Space Requirements

The first thing to check before you try to install Sablime is that you have a file system available with enough free space for the Sablime executables and databases. Be sure there is enough space for each product or each Sablime instance you will be installing.

The Sablime executable set (i.e. the Sablime bin directory and sub-directories) ranges in size from about 100 MB to 250 MB depending upon the architecture of the system. The precise size of the distribution can be found on the Sablime Web Site download page.

Estimates of the space required for three of the four databases are given in the following table:

Table 2-1. Initial Space Requirements

Database	Size
Active Database	3 MB
Inactive Database	.4 MB
Global Database	.3 MB

These databases grow with activity. A mature product can easily build active/inactive databases of 100 MB or more. The inactive database stores records that have been moved from the active database when MRs or generics get closed. Thus all growth happens first in the active database.

The Global database grows at a much slower pace, and is unlikely to ever grow beyond a few MB in size.

The size of the Source Database (SDB) depends on the number and size of the files being controlled. To estimate the initial space requirements for your Source Database, populate an empty node with all the source files for your project. Execute the UNIX system command "du -sk ." at the top of the node. This command reports the amount of disk space used by the files. Use this estimated number with an additional 10 per cent to allow for Source Code Control System (SCCS) header information that will be added to the beginning of each file; if you use the Source and Binary Control System (SBCS) for storing all files, the actual

storage space needed will be less than that used by the uncompact files. You should also plan for the growth in the Source Database as a result of any expected modifications or additions to the database. Obviously, delta (change) activity increases the space requirements. Tripling the figures obtained from the `du` command will provide a reasonable estimate of the space required for approximately three years of development.

The following guidelines will help you estimate the space your product will need as development proceeds:

- n Each MR (including a description file one page long) adds a minimum of 5 KB to the Active Database. If you add User-Definable Fields (UDFs) to collect more data, this number may increase.
- n The addition of each file adds 4 KB blocks to the Active Database.

Software and Other Requirements

Before running the Sablime installation script, check the following items:

- n A minimum process limit size of six megabytes (6 MB) is available for the *sablime* login and for all other logins using Sablime.
- n A *sablime* login has been established on the system.
- n The Korn shell (`ksh`) is the default shell for the *sablime* login and all other logins that use Sablime.
- n SCCS is available on your machine (if you want to use SCCS to control non-binary files).
- n Networking (NFS or TCP/IP) is established between machines (if you want to use the multi-machine or the External MR Communications features).
- n The `tbl`, `pic`, and `grap` preprocessors and the `troff` processor are available (if you want to use pie and bar charts as management reports).



NOTE:

These tools are available within Lucent through the "exptools" facility. See <http://exptools.web.lucent.com>. They are part of the "dwb" (Documenter's Workbench) package.

If these are not on your system and you do not have access to "exptools", they can also be found in the "Gnu" set (<http://www.gnu.org>). A public-domain version of `grap` is also maintained at <http://www.lunabase.org/~faber/Vault/software/grap/>.

Sablime does not provide support for these tools.

- n All Sablime users' directories have permissions of at least 755 and all files that are stored in Sablime have permissions of at least 444.

⇒ **NOTE:**
If you are installing Sablime on a machine running UNIX system V release 3.2 or later, the user ID (uid) number for the *sablime* login and all Sablime users must be greater than 100 because this version of the operating system turns off the set user ID (suid) bit for logins with uid numbers less than 100. The uid number is the third colon-separated field in the */etc/passwd* file. For more information, see your UNIX system administrator.

⇒ **NOTE:**
If you are using Curses Forms interface on a machine or terminal that allows windows, be sure that the window in which Sablime will run is at least 24 rows by 80 columns. Otherwise, the Sablime screens will not be properly displayed.

Downloading the Sablime Software

Detailed information about downloading the Sablime software is available in the Sablime Release Notes.

The License Manager

Every year projects using Sablime must obtain a Sablime user license from the Sablime organization. Because the license tracks each Sablime user, a project must determine how many different project members will be using Sablime each year. For example, if a project has 20 members who will be using Sablime, then a 20-user Sablime license should be obtained from the Sablime organization. If, during the year, additional project members need to use Sablime, replacement license files can be obtained.

⇒ **NOTE:**
As of release v5.2, update 1, the *create* and *report* commands do not require the user to be licensed. The users must still have a PTS record, but they need not take up a license.

The licensing of an individual Sablime user can occur in one of two ways:

1. The Sablime Database Administrator can explicitly license a Sablime user by executing the *pts* command and setting the *Licensed:* field on the PTS screen to *y*. (Only the Sablime Database Administrator can set this field.)
2. If a Sablime user who is not licensed executes a Sablime command, the user will automatically be licensed if the project has not already reached the maximum number of Sablime licenses that it is allowed.

The License File



CAUTION:

Do not edit this file. Sublime commands may not function if incorrect changes are made to this file.

Your Sublime license is governed by information stored in the `.usrid` file. The `.usrid` file will most likely be delivered via email. If you are using SBCS, you must copy the `.usrid` file to a file called `sbc`s in the SBCS license directory on both the host and satellite machines, as follows:

```
cp .usrid $Sablime_MCB/SBCS/lib/license/sbc
```

A typical `.usrid` file for v6.0 looks like this:

```
sablime:YcNC7vt4uQwDhG
sbc:s:yEyuLUnGSISkBD
merge:yTYLNxZU5oFCW6
host=mozart|136.4.118.99|191.18.4.*
expire=03/03/05
total=20
mtotal=8
local=1
# *** Changes to this file ... -- DO NOT EDIT! ****
user=sablime|jfk|lbj|rmn|jcc|ghf|rr|ghwb|wjc|gwb
muser=mkp|twh
```

It consists of one or more product-identification lines that contain a product name (e.g., `sablime`, `sbc`s, `merge`) followed by a 14-character license code. Following the product-identification lines, there is a `host` line, which identifies the host for Sublime and may contain multiple host names or IP addresses. The `"expire="` line of the file contains the expiration date of the current license. (Sublime generates an automatic warning seven calendar days before the expiration of the license.) The `"total="` line specifies the maximum number of users that can be licensed with the licensing file for Sublime. The `"mtotal="` line specifies the number of licensed merge tool users. The `"local="` line is always `"local=1"`.

The following line (beginning with `"user="`) will be added to `$sabMCB/.usrid` by the Sublime software as users are licensed. Similarly, an `"muser="` line will be added to `as merge tool users` become licensed. Each line includes one or more PTS IDs of licensed users, separated by vertical bars.



NOTE:

The `sbc`s file must not contain `"user="` lines. Only the `$sabMCB/.usrid` file may contain `"user="` lines.

Running the Installation Script

You establish a Sablime instance or Sablime product by executing the `initsab` script and then modifying the `sablime.sh` and `xsablime.sh` scripts.

If you are installing a new product, you may choose to include it in the same Sablime instance as an existing product (in which case the new product will use of the Global Database associated with that instance), or you may establish a separate Sablime instance for the new product. Using a single instance allows you to use single Global Database and a single set of Personnel Tracking System identifications (PTS IDs), which simplifies maintenance. Using separate instances provides complete isolation of the databases associated with each product.

After you have installed Sablime, you may customize it for your product in a variety of ways. See Chapter 3, *Customizing the User Interface*, for detailed information.



NOTE:

A certain amount of customization is essential to make Sablime meaningful for your product. After running the installation script, be sure to customize the user interface for your product as described in the section *Essential Product Customization* in Chapter 3, *Customizing the User Interface*.

What the Installation Script Does

The installation script `initsab`:

- n Installs the Active, Inactive, Source, and (for a new instance) Global Databases.
- n Establishes data related to input fields. You may change this data to customize Sablime for your product after you have run `initsab`. (See *Customizing Commands and Command Screens* in Chapter 3, *Customizing the User Interface*.)
- n Establishes Sablime-Level and Product-Level Reserved Groups. You may change the defaults and members in some of these groups to customize Sablime for your product after you have run `initsab`. (See *Customizing Pop-Up Selection Windows* in Chapter 3, *Customizing the User Interface*.)
- n Establishes templates for the database files created in the `commit`, `create`, `dbedit`, `fcreate`, `dbstop`, `edput`, `hcode`, `mredit`, `mrgedit`, `mrnote`, `pdi`, `propose`, `reject`, `setgroup`, `spawnmr`, and `submit` commands. You may change the template text to customize Sablime for your product after you have run `initsab`. (See *Customizing Templates* in Chapter 3, *Customizing the User Interface*.)

- n Establishes the initial generic for the product and assigns administrative and testing personnel to groups for that generic. You may rename the generic and change the members of these groups after you have run `initsab`.

Setting Up a Directory Structure File

Before you execute `initsab`, you can create a file containing the product directory structure. Alternatively, this file can be constructed dynamically during the execution of `initsab`.

Suppose that you have a directory structure like the one shown in the figure below, and that the name of your first generic (i.e., release) is `v5.0`.

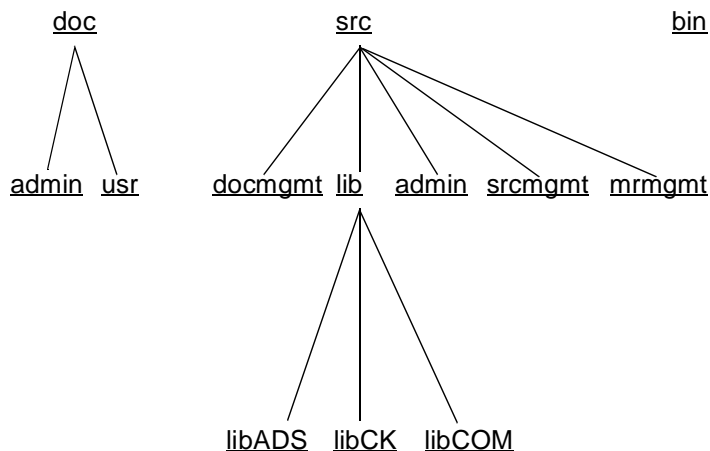


Figure 2-1. Sample Directory Structure

Before executing `initsab`, you would create a file named `v5.0` like the one shown below in Figure 2-2 to list all the directories, being careful to list each top-level directory before its subdirectories.



CAUTION:

Make sure that the directory names in the directory structure file do not begin with dot-slash (`./`); Sablime does not interpret that syntax correctly.

```
doc
src
bin
doc/admin
doc/usr
src/docmgmt
src/lib
src/admin
src/srcmgmt
src/mrmgmt
src/lib/libADS
src/lib/libCK
src/lib/libCOM
```

Figure 2-2. Sample Directory Structure File v5.0



NOTE:

Even if you do not plan to put any source files into Sablime, `initsab` still requires a directory file that contains at least one directory name. As mentioned above, you can create this file ahead of time, or you can let `initsab` put you into your editor to create it when it is needed.

Gathering the Information You Will Need

As the `initsab` program executes, you are prompted for information. To simplify the initialization procedure, look over the next section, *Running the Script*, and note the responses you want to make when you run the program. You may want to use *Worksheet 2: Installation Information (initsab)* in Appendix E to write your responses down and then keep the worksheet in front of you as you run `initsab`. In general, the information you will need to know is:

- n Product and generic (release) names
- n Project organization number
- n Administrator and test team logins
- n Directory paths for storage of product database directories
- n The major programming language used
- n The host machine name and the network type (if you will be installing Sablime on multiple machines)

- n The MR classes to be used (from among *document*, *hardware*, *software*, and *firmware*). MRs can only modify files that are of a file *type* in the same class as the MR. See the `addisrc` command.

⇒ **NOTE:**
You will also be asked to choose a type of MR dependency. For a discussion of this critical topic, see the section on *MR Dependencies* in the *Sablime User's Guide*.

Running the Script

⇒ **NOTE:**
Have your completed *Worksheet 2: Installation Information (initsab)* (see Appendix E) available if you have to call the Sablime helpdesk for help with installation.

⇒ **NOTE:**
Before you execute `initsab`, make sure that the `.usrld` license file is in the *sablime* Master Control Bin with permissions of 444. See the section *The License File*, above.

⇒ **NOTE:**
When you have finished running the installation script, you should read and follow the procedures in *Setting Up the Sablime Environment* in this chapter, and *Essential Product Customization* in Chapter 3, *Customizing the User Interface*, before using Sablime for your product.

To run the installation script:

1. Log in to the *sablime* login on your host machine; do not use the UNIX system `su` command.
2. Change directory to the *sablime* Master Control Bin, the directory containing the Sablime executables.
3. Execute the `initsab` program by entering the following command at the system prompt:

```
./initsab
```

The following two screens will appear:

```
*****
                Welcome to the Sablime (R)
                Configuration Management System

This script will initialize Sablime parameters and set up the
Global, Active, Inactive, and Source Databases for your
Product. In addition, the first Generic will be established
and you will be prompted for Administrative and Test Team
members. Finally the Sablime Level and Product Level Reserved
Groups will be created and the Field Tracking Data (FTD)
will be added.

The following items and/or information will be needed in
order to complete this initialization process:

- Korn Shell as default shell for all logins using Sablime,
- SCCS for source code control only.
*****

<<< Press the Return Key when ready to continue >>>
```

Figure 2-3. initsab Welcome Screen

```
*****
You will be prompted for the Following Questions:

Multi-Machine Env.      Source Admin Members
Global Database         Hardware Members
Active Database         Gen Admin Members
Inactive Database      MR Classes
Source Database        Document Test Teams & Members
Command Bin            Firmware Test Teams & Members
Product Name           Hardware Test Teams & Members
Product Title          Software Test Teams & Members
Generic Name           Trace Files
Major Prog. Lang.     MR History
Project Org. Num.     Developer Mail
Project Phone Num.    Collect In-Process Metrics
Project Site           Automatic Routing
Dir. Struct. File     Automatic Assignment
Version Ctrl Tool     MR Reassignment
DB Admin Members1     Automatic Dependency
MR Admin Members

*****

<<< Press the Return Key when ready to continue >>>
```

Figure 2-4. initsab List of Prompts

Each time you press RETURN, you will be prompted for one of the items listed on the above screen. Each prompt is explained in detail below. Press the interrupt key at any time to stop the process and abort the installation.

PROMPT: Will this be a 'Multi-Machine' Product [y or n]:

RESPONSE: Enter **y** if you plan to install Sablime so that your product can be used on more than one machine; if not, enter **n**. If you enter **y**, the following screen is displayed.

```
+++++
SET UP FOR MULTI-MACHINE Sablime

Upon completion of the 'initsab' process, please read the
following sections in the Sablime Administrator's Guide:

- Using Sablime on Multiple Machines
- Customizing the dot sablime Script

The Execution of Sablime commands can work across either of the
networks listed below:

- RFS/NFS
- TCP/IP

+++++
<<< Press the Return Key when ready to continue >>>
```



NOTE:

When you are asked for the location of a database or bin, you must enter the full path of an existing directory path that is owned by the *sablime* login. Sablime then creates the necessary subdirectories or files in that location. The *sablime* home directory is the default location for installing the databases.



CAUTION:

Do not enter /tmp or /usr/tmp as a location for creating the databases; these directories are temporary space that could be removed at any time.

PROMPT Would you like all four databases (global, active,
 inactive, source) created in the same location [y/n]?

 Location for creating the databases.
 Default- '[sablime home directory]':

RESPONSE Enter **y** and the full path name to the location to create all the databases
 in the same location. If you enter **n**, you will be prompted for the location
 of each database, as follows.

PROMPT: Location for creating [GLOBAL DATABASE]
 Default- '[sablime home directory]':

RESPONSE: Enter the full path to the location where the Global Database will be created and stored. Enter RETURN for the default.
Example: **/usr1/sablime**

PROMPT: Location for creating [ACTIVE DATABASE]
Default- '[sablime home directory]':

RESPONSE: Enter the full path to the location where the Active Database will be created and stored. Enter RETURN for the default.
Example: **/usr1/sablime**

PROMPT: Location for creating [INACTIVE DATABASE]
Default- '[sablime home directory]':

RESPONSE: Enter the full path to the location where the Inactive Database will be created and stored. Enter RETURN for the default.
Example: **/usr1/sablime**

PROMPT: Location for creating [SOURCE DATABASE]
Default- '[sablime home directory]':

RESPONSE: Enter the full path to the location where the Source Database will be created and stored. Enter RETURN for the default.
Example: **/usr1/sablime**

PROMPT: Location of [COMMAND BIN]
Default- '[sablime home directory]/bin':

RESPONSE: Enter the full path to the location of the Sablime commands. Enter RETURN for the default.
Example: **/usr1/sablime/bin**

PROMPT: PRODUCT NAME [Up to 5 characters]:

RESPONSE: Enter a name for your product. This name is also used as the MR prefix. You can enter up to five alphanumeric characters. The product name cannot contain a period (.) or a semicolon (;).
Example: **sab**



NOTE:

If you plan to use PCs for development work, make sure that the product and generic names can be valid DOS file names.

PROMPT: PRODUCT TITLE [String, up to 50 characters]:

RESPONSE: Enter the title for your product. You can enter up to 50 characters.
Example: **Lucent Technologies Configuration Management System**

PROMPT: GENERIC NAME [Up to 14 characters]:

RESPONSE: Enter a name for the first generic (release) for this product. You can enter up to 14 printable characters. Be careful with characters that the UNIX system shell considers "special" (such as *, ~, \$), especially at the beginning of the name. You can use a period inside the name (e.g., v4.2). The generic name must be unique in the Sablime instance. We recommend that the generic name reflect both the product name and the release.
Example: **sab4.2**

PROMPT: MAJOR PROGRAMMING LANGUAGE

bal	fortran	perl
c	html	pli
c++	java	sal
cobol	jcl	shell
comp	limbo	snoflake
dbd	lke	spitbol
dirjcl	mark4	vb
document	mll	other
ntp	parms	

Select 'ONE' from list of choices:

RESPONSE: Enter a programming language from the list. Enter only one even if more than one language is used. An entry is required. If your major programming language is not in this list, enter **other**. When you have finished running initsab, you can change the major programming language by running the PRX subcommand of the setrel command.
Example: **c++**

PROMPT: PROJECT ORGANIZATION NUMBER
[e.g. xyz1234500, up to 20 characters]:

RESPONSE: Enter the number of the department or organization responsible for the product.
Example: **xyz1234500**

PROMPT: PROJECT CONTACT PHONE NUMBER
[e.g. 1-908-582-7118, up to 15 characters]:

RESPONSE: Enter the phone number of the contact for the product.
Example: **1-908-582-7118**

PROMPT: PROJECT SITE
[e.g. Murray Hill, up to 20 characters]:

RESPONSE: Enter the site where the product development is taking place.
Example: **Murray Hill**

PROMPT: PATH to file containing directory structure for generic
[or hit carriage return to build file now]:

RESPONSE: Enter the full path to the file containing the directory structure for the first generic. This file is the one you created before you executed the initsab script. (See *Setting Up a Directory Structure File*, above.)
Example: **/usr1/sablime/v5.0**



NOTE:

This file is copied to the directory identified by \$sabGDB/DIR/*generic_name*. The variable sabDIRF is set to the full path of this file.

PROMPT: Version Control Tool for Non-Binary Files
[SBCS, SCCS or Either]:

RESPONSE: Enter **SBCS** or **SCCS** to specify which tool to use as a default version control tool for non-binary files. Enter **Either** to force the user to choose the tool when using addisrc. Binary files can only be stored under SBCS.



NOTE:

After you have responded to each of the above prompts, a screen like the following is displayed, showing the information you have supplied and giving you an opportunity to make changes.

```
*****
The following information will be used to set up your product:
1) PRODUCT NAME:      [sab]
2) PRODUCT TITLE:    [Lucent Technologies Configuration Mgmt System]
3) GENERIC NAME:     [sab5.0]
4) PROGRAM LANGUAGE: [c++]
5) ORGANIZATION NUMBER:[xyz12345]
6) CONTACT PHONE NUMBER:[1-908-582-7118]
7) PROJECT SITE:     [Murray Hill]
8) VERSION CONTROL TOOL:[Either]

The following locations will be used to set up:
9) GLOBAL DATABASE:  [/user1/sablime/gdb]
   ACTIVE DATABASE:  [/user1/sablime/adb/sab]
   INACTIVE DATABASE: [/user1/sablime/idb/sab]
   SOURCE DATABASE:  [/user1/sablime/sdb/sab]
10) HOST COMMAND BIN:[/user1/sablime/bin]
*****
```

Figure 2-5. initsab Summary of Responses

- PROMPT: Would you like to change the above information [y or n]?:
- RESPONSE: If you enter **y**, the following prompt will appear:
- PROMPT: Enter the number of the incorrect item:
- RESPONSE When you enter the number of the item you want to change, you will be prompted for the new value. The Summary of Responses screen will then reappear with the corrected information, and you will be asked if you want to make any further changes.

When you enter **n** to indicate that no more changes are required, the following message appears:

```
WARNING: Processing has BEGUN!!!
Do Not Exit
```


The program now builds the Global (if this is a new instance), Active, and Inactive Databases, the Source Database node, and the product directory structure. This process is complex and may take a very long time to run; be patient. If an error occurs at any time during this process, a message informs you of the problem and the program ends. Correct the problem if possible, and rerun `initsab`. If you cannot correct the problem, call the Sablime hotline for assistance.

If the database initialization completes with no problems, the following message is displayed:

```
ATTENTION:The Following have been installed:
```

```
Global Database
```

```
Active Database
```

```
Inactive Database
```

```
Source Database
```

Next, the following message appears:

```
The Mandatory Sablime Groups Are Now Being Set Up  
Please Stand By....
```

When the processing is complete, the following message is displayed:

```
Group Relations are now complete
```

Next, tuple files are created and records are written for the FTD relation in the Active Database. One record is created for each field in each command in Sablime. The following message is displayed:

```
[initftd] is being executed, please stand by...
```

When the processing is complete, the following message is displayed:

```
Relation [FTD] is now updated
```

Next, if this is an initial generic, groups are created to identify the personnel responsible for administrative and testing tasks. Then, group membership is defined using information you give in response to prompts. The following message is displayed:

```
<<< Give ANSWERS in the form of a COMMA SEPARATED list, no spaces >>>  
login id should not include machine (e.g., sablime)
```

Respond to the prompts that follow with one or more valid logins. Use commas to separate multiple logins (e.g., **li,scott**). The procedure checks that each login is valid; if it is not, an error message is produced. If the login is valid, the procedure looks in the PTS relation for the ID. If it does not exist there, a new PTS record is created for that ID. An existing PTS record in an existing Global Database is automatically authorized for the new product.



NOTE:

The *sablime* login does not have to be included as a group member for any of the administrative groups (DBA, MRA, SA, or GA); it is added automatically.



CAUTION:

When you run `initsab` and set up the administrative and test teams, be careful not to enter more than 256 characters of input for any team.

PROMPT: Database Administration:

RESPONSE: Enter the logins of personnel who will have Database Administrator privileges.
Example: **mjf,li**

PROMPT: MR Administration:

- RESPONSE: Enter the logins of personnel who will have Modification Request Administrator privileges.
Example: **marsar,jkh**
- PROMPT: Source Administration:
- RESPONSE: Enter the logins of personnel who will have Source Administrator privileges.
Example: **li,jkh**
- PROMPT: Generic Administration:
- RESPONSE: Enter the logins of personnel who will have Generic Administrator privileges.
Example: **marsar,pf**
- PROMPT: Hardware Administration (optional):
- RESPONSE: Enter the logins of personnel who will have Hardware Administrator privileges.
Example: **sablime,li**
- PROMPT: Which of the following MR Classes would you like?
Note: At least one MR Class must be selected!
- RESPONSE: Enter **y** or **n** in response to each of the following four prompts to select the MR classes to be used in this generic. You can assign test teams and move MRs through specified states for each class selected.
- Are you going to have Document Class MRs: [y or n]
Are you going to have Firmware Class MRs: [y or n]
Are you going to have Hardware Class MRs: [y or n]
Are you going to have Software Class MRs: [y or n]

After you have chosen the MR classes you want to use, you are prompted to supply names for test team members for each state allowed in the classes you have chosen.



NOTE:

The messages, prompts, and responses shown below are those that appear if you choose to have Software Class MRs. Messages, prompts, and responses are similar for the other three classes. For the Document Class, the team names displayed refer to Inspection and Publishing teams instead of Integration Test and System Test teams, respectively. Each of the four MR classes can have up to seven teams defined.

If you choose Software Class MRs, the following message is displayed:

```
Set Up Product Software Teams and States
```

```
<<< Give ANSWERS in the form of a COMMA SEPARATED list, no spaces >>>  
login id should not include machine (e.g., sablime)
```

PROMPT: Pre-Integration Team:

RESPONSE: Enter the logins of Pre-Integration testers. This is an optional field. If members are entered for this group, the `preitpass` command and the `preitpassed` state will be available for the Software Pre-Integration Test Team.

PROMPT: Integration Team:

RESPONSE: Enter the logins of Integration testers. This is an optional field. If members are entered for this group, the `itpass` command and the `itpassed` state will be available for the Software Integration Test Team.

PROMPT: Pre-System Test Team:

RESPONSE: Enter the logins of personnel who will have Pre-System Test privileges. This is an optional field. If members are entered for this group, the `prestpass` command and the `prestpassed` state will be available for the Software Pre-System Test Team.

PROMPT: System Test Team:

RESPONSE: Enter the logins of System testers. This is an optional field. If members are entered for this group, the `stpass` command and the `stpassed` state will be available for the Software System Test Team.

PROMPT: Pre-Approval Team:

RESPONSE: Enter the logins of Pre-Approval testers. This is an optional field. If members are entered for this group, the `preapprove` command and the `preapproved` state will be available for the Software Pre-Approval Team.


PROMPT: Approval Team:

RESPONSE: Enter the logins of personnel who will have Approval privileges. This is a mandatory field. The `approve` command and the *approved* state will be available for the Software Approval Team.

PROMPT: QA Team:

RESPONSE: Enter the logins of personnel who will have Quality Assurance privileges. This is an optional field and is needed only if the optional Quality Assurance feature is used.

The next set of prompts asks for yes (**y**) or no (**n**) responses.

- PROMPT Do you want C-Sab Temporary Branching enabled [y or n]?
- RESPONSE Enter **y** if you want to permit Temporary Branching (unreserved check-outs), otherwise enter **n**. Enter **n** if you want the same behavior as in Sublime releases prior to v6.0.
- PROMPT: Do you want a 'Trace File of the Commands' kept [y or n]?
- RESPONSE: Enter **y** if you want to create a file that will contain tracking data for all commands executed by each PTS ID; otherwise, enter **n**.
- PROMPT: Do you want a 'MR History File' kept [y or n]?
- RESPONSE: Enter **y** if you want to create a file that will contain tracking data for all MR activity; otherwise, enter **n**. We recommend **y**.
-  **NOTE:**
The history file tracks a great deal of important information and should be kept.
- PROMPT: Do you want mail turned on for all the developers [y or n]?
- RESPONSE: Enter **y** if you want mail to be sent to the appropriate administrator, developer, or test team. Otherwise, enter **n**. We recommend **y**. If a project is very small (i.e., one or two people), you may want to turn mail off. However, in most cases, mail should be left on to ensure good project communication.
- PROMPT: Do you want to collect In-Process Metrics Data [y or n]?
- RESPONSE: Enter **y** if you want in-process metrics data collected for your project; otherwise, enter **n**.
- PROMPT: Do you want Automatic MR Routing [y or n]?
- RESPONSE: Enter **y** if you want MRs to be routed automatically to the appropriate MRAs for your project; otherwise, enter **n**.
- PROMPT: Do you want Automatic MR Assignment [y or n]?
- RESPONSE: Enter **y** if you want MRs to be assigned automatically to the appropriate ADs for your project; otherwise, enter **n**.
- PROMPT: Do you want MR Reassignment [y, n, or all]?
- RESPONSE: Enter **y** if you want to allow ADs and GAs to reassign MRs for your project. Enter **n** to allow only the GA to reassign MRs. Enter **all** if you want to allow all Sublime users to reassign MRs.

PROMPT: Do you want Automatic Dependency [y or n]?

RESPONSE: Enter **y** if you want MRs in your project to be made dependent automatically; otherwise, enter **n**. The following two prompts are displayed only if you answer **y** to the above prompt.

PROMPT: Automatic Dependency Type [line-level or file-level]?

RESPONSE: Enter **file-level**, **line-level**, or **none** according to your product development needs. Automatic dependency affects the `edput` command and the MR used to return a modified file to the Source Database. When you use `edput` and a type of automatic dependency is selected, the current MR is made dependent upon any unapproved MRs that have modified this file according to the following conditions:
file-level—current MR and unapproved MRs touched any line in this file
line-level—current MR and unapproved MRs touched the same line in this file.
We recommend that you take advantage of one of these types of dependency for SCCS files. For details, see the section on *MR Dependencies* in the *Sablime User's Guide*. If you need to change dependency type later, you can do it with the ADM subcommand of the `setrel` command. SBCS files have implicit file-level dependency.

PROMPT: Do you want Developer Override of Automatic Dependency [y or n]?

RESPONSE: Enter **y** if you want ADs to be allowed to override Automatic Dependency when using the `edput` command. Otherwise, enter **n**.

Once all this information is entered, the ADM relation, the G relation, the GT relation, the GRP relation, and the GRPM relation are updated to store the information you have supplied. The following message is displayed:

```
*****  
  
The ADDGEN Command Is Now Being Executed  
Please Stand By....  
  
*****
```

The next step is to add the new generic. When this step is complete, the following types of messages are displayed:

- + An internal id [1.1] has been generated for new Generic [sab5.0].
- + Generic [sab5.0] G Relation Record added to the Active db.
 - A new GT Relation Record for document teams/states added.
 - A new GT Relation Record for hardware teams/states added.
 - A new GT Relation Record for software teams/states added.
- + The TR Relation Record has been added to the Global Database.
 - Active Database [FILES/solution] master directory validated.
 - Active Database [solution] directory for Generic has been created.
 - Inactive Database [FILES/solution] master directory validated.
 - Inactive Database [solution] directory for Generic has been created.
 - Active Database [FILES/resolution] master directory validated.
 - Active Database [resolution] directory for Generic has been created.
 - Inactive Database [FILES/resolution] master directory validated.
 - Inactive Database [resolution] directory for Generic has been created.
 - Active Database [FILES/rejection] master directory validated.
 - Active Database [rejection] directory for Generic has been created.
 - Inactive Database [FILES/rejection] master directory validated.
 - Inactive Database [rejection] directory for Generic has been created.
 - Active Database [FILES/mrhistory] master directory validated.
 - Active Database [mrhistory] directory for Generic has been created.
 - Inactive Database [FILES/mrhistory] master directory validated.
 - Inactive Database [mrhistory] directory for Generic has been created.
 - Active Database [FILES/spawnotes] master directory validated.
 - Active Database [spawnotes] directory for Generic has been created.
 - Inactive Database [FILES/spawnotes] master directory validated.
 - Inactive Database [spawnotes] directory for Generic has been created.
 - Active Database [FILES/sqservice] master directory validated.
 - Active Database [sqservice] directory for Generic has been created.
 - Inactive Database [FILES/sqservice] master directory validated.
 - Inactive Database [sqservice] directory for Generic has been created.
 - Active Database [FILES/mrcommit] master directory validated.
 - Active Database [mrcommit] directory for Generic has been created.
 - Inactive Database [FILES/mrcommit] master directory validated.
 - Inactive Database [mrcommit] directory for Generic has been created.
 - Active Database [FILES/hcode] master directory validated.
 - Active Database [hcode] directory for Generic has been created.
 - Inactive Database [FILES/hcode] master directory validated.
 - Inactive Database [hcode] directory for Generic has been created.
 - Active Database [FILES/initdoc] master directory validated.
 - Active Database [initdoc] directory for Generic has been created.

If any problems are encountered, an error message is displayed; otherwise, the following message is displayed when the installation is complete:

```
*****
Note that the file, called the sabVAR file, has been created. The file name is
    [full path to sabVAR file]
The sabVAR file contains some of the variables required by Sablime.
When you customize the xsablime.sh script, you can either
    * set and export the sabVAR variable,
        [ i.e., export sabVAR=[full path to sabVAR file] ]
    OR
    * set and export all the variables contained in the sabVAR file.
We recommend that you define the variables sabGDB, sabPROD, sabGEN, and sabDIRF
in the xsablime.sh script since they are product and generic specific; leave
the rest in the sabVAR file.

Please refer to Sablime Administrator's Guide for more details.
*****

                Sablime is now INSTALLED
```

Figure 2-6. initsab Installation Complete Message

⇒ NOTE:
Sablime has now been installed for your product. However, the *dot sablime* programs must be modified before users can set up for the generic established for the product. Modification of these programs is explained in *Setting Up the Sablime Environment*, below. In addition, you must customize the user interface for your product. (See *Essential Product Customization* in Chapter 3, *Customizing the User Interface*.)

⇒ NOTE:
To install Web Sablime, refer to the instructions available at the Sablime Web Site, or to the ".html" files included with the Sablime distribution.

Setting Up the Sablime Environment

Defining Environment Variables

Sablime commands rely on a number of variables. Because a large number of environment variables can affect overall performance of the commands, you have the option of moving all but the `sabVER` and `sabVAR` variables to a Sablime variable file. If you do this, the `sabVAR` environment variable is set to the full path of this file. But if you prefer, you may instead define all the variables as environment variables in the `sablime.sh` and `xsablime.sh` files. Or, you can divide the variables between both locations if you wish. If a variable is defined in both places, the environment variable takes precedence over the variable in the `$sabVAR` file.

If you want to use the `$sabVAR` file for variables, we suggest that you put it in the same directory as the Sablime commands; however, you can put it anywhere you like. When you set up your environment by customizing *dot sablime* (see the next section), you must enter the full path to this file.

The definition format for the variables in the `$sabVAR` file is:

```
variable=value
```

For example,

```
sabHOST=mozart
```

The definition format for the environment variables in the `sablime.sh` and `xsablime.sh` files is:

```
export variable=value
```

For example,

```
export sabHOST=mozart
```



CAUTION:

Do not enter any spaces around the equal sign in either of these formats, and do not include any comments or additional characters after the value, or the Sablime commands may not work correctly and your files could become corrupted.

You can use both the `whence` and `whereis` commands to find the path to an executable. If you are not certain of the path to the commands required by any of the following variables, ask your UNIX system administrator.

You may define the following variables either in your \$sabVAR file or as environment variables in the sablime.sh and xsablime.sh scripts. Variables preceded by an asterisk [*] are mandatory. Examples use the \$sabVAR file format.

Table 2-2. Sablime Variables


Variable	Explanation
mergeVAR	Can be used to define an alternative merge utility, in place of Sablime's "SabMerge". See the "smerge" command in the User's Reference Manual.
sab1MR	Allows external MR communicating Sablime products to use the same MR number. If this variable is set to y , then the MR creation operation initiated by the enter operation of the review command will use the MR number of the sending product for the MR number created locally. (See Chapter 7, <i>Using the External MR Communication Commands</i> .)
sabDIRF	Full path to the file containing the directory structure for the generic. Example: <code>sabDIRF=/usr1/sablime/v5.0</code>  NOTE: See <i>Setting Up a Directory Structure File</i> , above.
sabDKFB	Host machine name recognized by your satellite machine. This variable is needed only if you run in multi-machine mode, use TCP/IP, and the host machine name is different from that recognized by your satellite machine. For TCP/IP satellites: If your host machine name is an alias for another machine, set this variable to the official host name. See the example in step 6 in <i>Using Sablime over TCP/IP</i> , below, for details
sabGDB	Identifies the Sablime Global Database.
sabGEN	Identifies the Sablime Generic.

Table 2-2. Sablime Variables—Continued

Variable	Explanation
*sabHOST	<p>The name of the machine on which the databases reside. To obtain the host machine name, use <code>uname-n</code>.</p> <p>Example:</p> <p><code>sabHOST=mozart</code></p>
*sabLCB	<p>Full path to the Local Control Bin in which the Sablime commands are stored on this machine. (On a host machine, or if you are operating in single-machine mode or in a homogeneous NFS/RFS environment, the <code>sabLCB</code> and <code>sabMCB</code> variables are identical.)</p> <p>Example:</p> <p><code>sabLCB=/usr1/sablme/bin</code></p>
*sabMCB	<p>Full path to the bin in which the Sablime commands are stored on the host machine. (On a host machine, or if you are operating in single-machine mode or a homogeneous NFS/RFS environment, the <code>sabLCB</code> and <code>sabMCB</code> variables are identical.)</p> <p>Example:</p> <p><code>sabMCB=/usr2/sablme/bin</code></p>
sabMFR	<p>Full path to the location of the <code>grap</code> executable. This variable is used by the <code>report</code> command only to produce pie chart and bar chart reports.</p> <p>Example:</p> <p><code>sabMFR=/usr/local/bin/dwb/grap</code></p> <p>⇒ NOTE: <code>grap</code> is used by the Sablime <code>report</code> command only; it is not supported by the Sablime team. Within Lucent, <code>grap</code> is available (including source, if desired) through "exptools". See http://exptools.web.lucent.com. It is part of the "dwb" (Documenter's Workbench) package. There is also a public-domain version of <code>grap</code> being maintained (version 1.22 was released Jan. 7, 2002). See http://www.lunabase.org/~faber/Vault/software/grap/.</p>

Table 2-2. Sablime Variables—Continued




Variable	Explanation
sabNET	<p>Set to the code for the type of network being used between machines. The acceptable codes are:</p> <ul style="list-style-type: none"> n 0=host machine or no network n 5=NFS/RFS n 7=TCP/IP. <p>Example: sabNET=0</p> <p> NOTE: When you use the Multi-Machine feature, you must set sabNET to 0 (zero) on the host machine. See <i>Using Sablime on Multiple Machines</i>, below.</p>
sabOPATH	<p>This variable is used in PATH environment variable manipulation within the <code>xsablime</code> script. It attempts to limit the growth of your PATH environment variable after successive executions of <code>. sablime \$sabGEN</code>.</p>
sabPROD	<p>Defines the current Sablime product.</p>
sabPTS	<p>This variable should always be set to the following: sabPTS=1</p>
*sabSERV	<p>On the host machine, sabSERV=/dev/null</p> <p>On the satellite machine, if the host and satellite communicate across TCP/IP, sabSERV=net_recv</p> <p> NOTE: The <code>net_recv</code> executable may be renamed; check with your UNIX system administrator. If the satellite machine is connected to a host running more than one version of Sablime (e.g., 4.3 and 5.0), the <i>dot sablime</i> program must include a case statement to reference separate sabVAR files with a different sabSERV variable and different sabMCBs and sabLCBs for each version of Sablime.</p>

Table 2-2. Sablime Variables—Continued

Variable	Explanation
sabSPOOL	<p>Full path to the parent directory where the MailSpool directory resides. This variable is used to change the default path, which is set to the value of sabLCB. If multiple MailDemons are desired, sabSPOOL can be used to set up individual mail processing directories (for details, see <i>Customizing Sablime Mail</i> in Chapter 4, <i>Other Administrative Procedures</i>).</p> <p>Example:</p> <pre>sabSPOOL=/usr/public</pre> <p> NOTE: You must create the /usr/public/MailSpool directory before running Sablime for the first time.</p>
sabTMP	<p>Full path to the directory where Sablime commands can create temporary files. The default value for this optional variable is /usr/tmp.</p> <p>Example:</p> <pre>sabTMP=/usr/add-on/local/tmp</pre>
sabVAR	<p>Identifies the full path to a file that contains Sablime environment variables. The purpose of this file is to allow one to reduce the number of Sablime variables defined in one's environment. This file is created automatically by the Sablime initsab command. The use of the sabVAR variable is optional.</p>
sabVER	<p>Identifies the current Sablime version.</p>
*sabVHELP	<p>Full path to the directory where the Sablime verbose messages are stored. The verbose help files are delivered in a directory called VHELP in the <i>sablime</i> bin.</p> <p>Example:</p> <pre>sabVHELP=/users1/sablime/bin/sabVHELP</pre>

Customizing the dot sablime Script

dot sablime command sets up the Sablime environment for a generic; it must be issued each time the user logs in to use Sablime. (See the section on Sablime Initiation in the *Sablime User's Manual* for more information about the execution of the *dot sablime* command.)

When the user issues the *dot sablime* command, two commands are actually executed, *sablime* and *xsablime*. These two commands are written as shell scripts and are delivered to you as *sablime.sh* and *xsablime.sh*.

The *sablime.sh* script:

- n Determines whether the user is running Korn Shell (ksh)
- n Establishes the current version of Sablime (*\$sabVER*) and the location of the Sablime variable file (*\$sabVAR*)
- n Exports the values of *sabVER* and *sabVAR* to the environment
- n Makes sure *xsablime* is executable and calls the routine to execute it
- n Passes the argument (the generic name) given on the command line to *xsablime*.

The *xsablime.sh* script:

- n Sets up all product-specific environment variables
- n Sets up all generic-specific environment variables
- n Checks the correctness of the GDB and the directory structure file
- n Creates or checks the user's private work node
- n Creates or checks the private work node's directory structure
- n Performs local setup requirements.

If you modify these scripts, you must do so on all machines (host and satellite) on which the new product will be established. You must modify *xsablime* shell for each new product; you may modify *sablime* shell if you wish.



NOTE:

In both the *sablime.sh* and the *xsablime.sh* scripts, the locations of areas where modifications must be made when adding a product to Sablime are specified by a comment block:

```
#####  
# USER MODIFICATION #  
#####
```

In the following two sections on making changes to *sablime.sh* and *xsablime.sh*:

- n If you choose to reduce the number of environment variables in your operating environment, you must define the *sabVAR* variable. Variables can then be placed in the *\$sabVAR* file.
- n If you do not want to remove any variables from the environment, you do not have to define the *sabVAR* variable.

Both these sections reflect the scripts as they are delivered; you can customize them to suit your development environment.

Making Changes to `sablime.sh`

To modify the `sablime.sh` script for your new product.

1. Search for `USER MODIFICATION`.
2. Move down two lines:

```
sabVAR=?????
```

Replace the question marks with the full path to the file where the set of environment variables is stored.

Example:

```
sabVAR=/usr6/sablime/sab5.0/.VARFILE
```

If you choose not to define this file, you must define and export the variables listed in *Defining Environment Variables*, above.

You can make other changes to this program to customize it for your product.



CAUTION:

Do not change the `sabVER` variable. It contains version information needed by Sablime support staff.

3. Write and quit the file.

Making Changes to `xsablime.sh`

Three sections of the `xsablime.sh` script must be changed for the first product installed in a Sablime instance.

- n The variables-export section
- n The print section
- n The case section.

If other products are added, the variables-export section must be modified only if the data is different for the new product. The other two sections must be modified for each new product.

The following sections show you how to modify the `xsablime.sh` script for your new product.

The variables-export section

1. Search for USER MODIFICATION.
2. Move down two lines:

```
export BASE=?????
```

Replace the question marks with the full path to the home directory of the *sablme* login.

Example:

```
BASE=/home/users9/sablme
```

The print section

1. Search for USER MODIFICATION.
2. Move down two lines:

```
print -u2 "'test5.0' (Sablme Test Environment;\nGeneric test5.0)"
```

For the first generic (the one created with the *initsab* program), change the first appearance of *test5.0* to the new generic name. Change the information inside the parentheses to a description of the new generic.

For subsequent generics, add a new print statement in the same format below this one, for example:

```
print -u2 "'generic_name' (generic description)"
```

The case section

1. Search for USER MODIFICATION.
2. Move down two lines:

```
test5.0 )
```

This line marks the beginning of a case in a shell case statement. It sets up three variables and provides some checking for network settings. The statement ends with two semicolons (;).

For the first generic, replace *test5.0* with the name of your generic.

3. Move to the next line:

```
export sabGDB=?????
```

Replace the question marks with the full path to the Global Database for the product for which your generic was added.

Example:

```
export sabGDB=/users6/sablme/gdb
```

4. Move to the next line:

```
export sabPROD=????
```

Replace the question marks with the name of the product for which the generic named in the print statement has been added. The product name must match exactly one of the products defined by initsab. For further information, see the PR subcommand of the setrel command.

Example:

```
export sabPROD=abcd
```

5. Move to the next text line:

```
export sabNET=?????
```

Replace the question marks with the code for the type of network being used between machines. The acceptable codes are:

- n 0 host machine or no network
- n 5 NFS/RFS
- n 7 TCP/IP.

Example:

```
export sabNET=0
```



NOTE:

When you use the Multi-Machine feature, you must set sabNET to 0 (zero) on the host machine. See *Using Sablime on Multiple Machines*, below.



NOTE:

If you are running NFS/RFS and the host bin is being shared by the satellites (i.e., the file system is mounted on the satellite machines), search for Uncomment the following and uncomment the following section of code:

```
#if [ "$LOCSYS" = "$sabHOST" ]  
#then  
#     export sabNET=0  
#else  
#     export sabNET=5  
#fi
```

6. The sabSPOOL variable is an optional variable used to reset the MailSpool directory. If sabSPOOL is not defined, the MailSpool directory is in the sabLCB directory. See *Customizing Sablime Mail* in Chapter 4, *Other Administrative Procedures* for more information about setting this variable.

Example:

```
export sabSPOOL=/usr/add-on/local/tmp
```

Changing `xsablime.sh` and the `sabVAR` File When Communication Over TCP/IP is Needed

If a user issues the dot `sablime` command from a TCP/IP satellite, `xsablime.sh` is among the commands executed. In order for it to work properly in such circumstances, `xsablime.sh` must be modified as follows:

1. Search for `xsablime.sh [Version]`
2. Search for `unset`
3. Remove all lines above `unset`
4. Search for `sabHOST=`
5. Add any lines required to set more environment variables from the `sabVAR` file
6. Search for USER MODIFICATION
7. Move down two lines:

```
print -u2 "'test5.0' (Sablime Test Environment;\
Generic test5.0)"
```

For the first generic (the one created with the `initsab` program), change the first appearance of `test5.0` to the new generic name. Change the information inside the parentheses to a description of the new generic.

For subsequent generics, add a new print statement in the same format below this one, for example:

```
print -u2 "'generic_name' (generic description)"
```

8. Search for USER MODIFICATION again
9. Move down two lines:

```
test5.0 )
```

This line marks the beginning of a case in a shell case statement. It sets up three variables and provides some checking for network settings. The statement ends with two semicolons (`;;`).

For the first generic, replace `test5.0` with the name of your generic.

10. Delete `export sabGDB=?????`
11. Replace the question marks for `sabPROD` with a value
12. Delete `export sabNET=?????`

In addition, the `sabVAR` file should contain the following settings (refer to Table 2-2 for full explanations of the variables):

```
sabDIFF=place on TCP/IP satellite
sabDIRF=place on TCP/IP satellite
sabHOST=host name
```

```

sabNCSL=place on TCP/IP satellite
sabPTS=1
sabLCB=place on TCP/IP satellite
sabMCB=place on host
sabNET=7
sabVHELP=place on TCP/IP satellite
sabSERV=net_recv
sabGDB=place on host
sabTM=/usr/tmp
sabDOT=place on TCP/IP satellite
sabEXT=place on TCP/IP satellite

```

Changing xsablime.sh When Adding a New Generic

Every time a new generic is added, you must add a new case to the case statement. (For information about adding a new generic, see the section *Adding a Generic* in Chapter 4, *Other Administrative Procedures*.) To do this, copy the lines:

```

test5.0 )
export sabGDB=?????
export sabPROD=?????

export sabNET=?????
###
# Uncomment the following "if-then-else"
# statement if using NFS/RFS and sharing
# the same ". sablime" script.
###
#if [ "$LOCSYS" = "$sabHOST" ]
#then
#     export sabNET=0
#else
#     export sabNET=5
#fi

###
# This variable need only be set if you do not
# want to use the MailSpool directory found in
# the Sablime bin. (Uncomment if necessary).
###
# export sabSPOOL=?????

if [ "$LOCSYS" = "$sabHOST" -o "$sabNET" = "5" ]
then
        export sabDIRF=$sabGDB/DIR/$sabGEN
else
        export sabDIRF=$sabLCB/$sabGEN
fi

sabSERV=`grep "^sabSERV=" $sabVAR | sed \
        's/sabSERV=//'\`

;;

```

to the next line after the preceding case statement. Add appropriate new values for the variables for the new generic. Then write and quit the file.



NOTE:

By default, the `xsablime.sh` script resets the `VPATH` variable to `$HOME/$sabGEN:$OFNODE`. If this definition of `VPATH` causes a problem in your build environment, you can remove it. If you remove the definition, the system may no longer calculate the relative directory for source management commands automatically.

Creating Executable Versions of `sablime.sh` and `xsablime.sh`

After you have made all the changes to `sablime.sh` and `xsablime.sh`, the programs must be made executable so that a user can issue the `dot sablime` command. To do this:

1. Run the `sh2x` script on each of the files to remove all blank lines, tabs, and comments. Elimination of blanks and tabs makes the resulting shells run (i.e., interpret) faster.

```
sh2x sablime.sh sablime
sh2x xsablime.sh xsablime
```

2. Run the UNIX system `chmod` command on each of the files to make them executable.

```
chmod +x sablime xsablime
```

The programs are now executable, and users may issue the `dot sablime` command and begin working on the new product.

Using Sablime On Multiple Machines

Sablime can be used to manage products across multiple machines. One machine, known as the host, maintains the four Sablime databases while the other machines, known as the satellites, can be used to execute most commands.

The multi-machine feature is useful for:

- n Larger projects that have systems engineering, development, and testing activities on different machines
- n Networks in which workstations are linked to a fileserver
- n Projects with a distributed product-development environment
- n Projects that need to distribute users on different machines for better load balancing.

The figure below shows a schematic diagram of possible multi-machine connections.

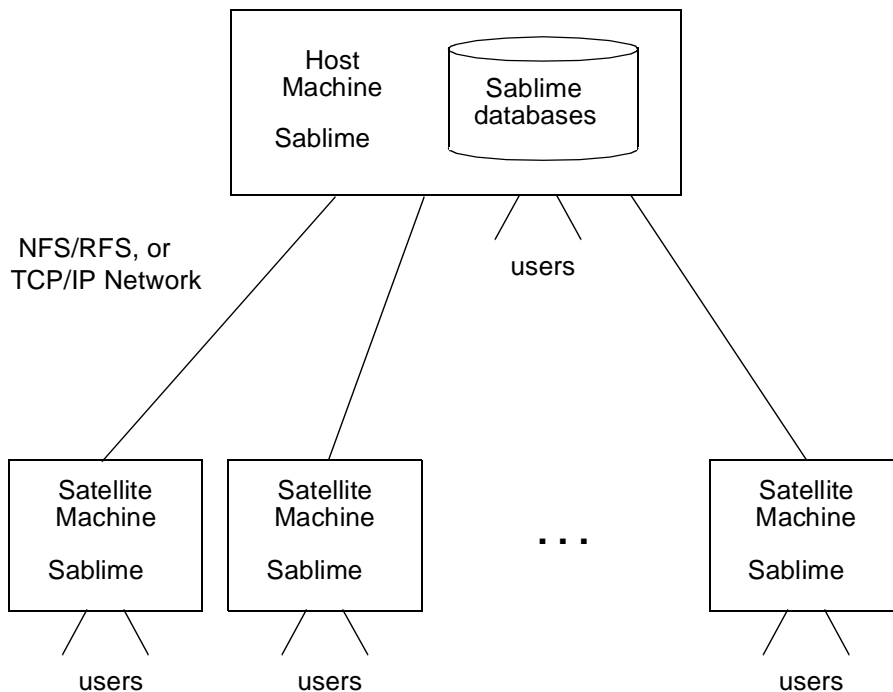


Figure 2-7. Multiple Machines for Sablime

Because only one set of Sablime databases exists, there is only one copy of official source files. Except for network overhead, the Sablime user sees no difference between running on the host and running on the satellite.

The networks currently supported by Sablime for multi-machine environments are:

- n NFS/RFS
- n TCP/IP.

A single network can support heterogeneous machines and network types. Users can work on the host or any of the satellite machines.

A copy of the Sablime executables must reside on the satellite machine for TCP/IP networks. For NFS/RFS networks, if host and satellite are the same type of

machine, only one bin is required; it must be on the host machine.

If you have a separate bin on the satellite, you must have a copy of the generic directory structure file in the satellite bin.

You can check whether you are on a satellite machine by entering:

```
echo $sabNET
```

If the result is not 0 (zero), you are on a satellite.

Installing Sablime On Multiple Machines

If a single set of Sablime databases will be accessed by more than one machine for your product, follow the steps specified below in the appropriate section for the network type (NFS/RFS or TCP/IP) to be used between machines. Then make sure you have done everything listed in the following section, *Notes for All Multi-Machine Configurations*.



NOTE:

For the examples in this section, it is assumed that all Sablime system variables are stored in the \$sabVAR file. See *Defining Environment Variables*, above, for details).

Notes for All Multi-Machine Configurations

- n All users with a login on a satellite machine must have the same login (including UID) on the host machine.
- n All users on a satellite machine must have PTS records in the Global Database (GDB) on the host machine.
- n Use the PR subcommand of the setrel command to make sure that the Multi-Machine Type field for the product is set to 1.
- n Copy the files for the appropriate generics in the \$sabGDB/DIR directory on the host machine to the satellite machine's Local Control Bin (\$sabLCB). Do not change the file names.
- n Permissions for the gdb/tmp and the adb/product_name/tmp directories must be set to 777.
- n Run the setperm script to set the correct permissions on all Sablime files.
- n Run the hotline.ck script after you have finished the setup for your network type. It identifies errors in your environment or configuration.

Using Sablime over NFS/RFS

To use the Sablime NFS/RFS multi-machine feature:

1. An NFS or RFS connection must be established between the host machine and the satellite machines.
2. The file system on the host machine where the Sablime databases reside must be mounted on the satellite machines. The file system on the satellite must have the same permissions as the file system on the host.



NOTE:

The database file system must be hard mounted, not soft mounted; if you have any questions about mounting the file system, see your UNIX system administrator.

3. The file system mounted on the satellite must have the same name as the file system on the host machine.
4. Set the `sabNET` variable to 0 (zero) on the host machine and to 5 (five) on the satellite machines. If you use the same `xsablime.sh` file for both your satellite and host machine, you must set the proper value for the `sabNET` variable for each generic in `xsablime.sh`.
5. Modify the `xsablime.sh` file and the Sablime variable file (`$sabVAR`) on each satellite machine to show the locations of temporary directories, the name of the host machine, and which network is being used.
6. If you are not running Network Information Services (formerly called Yellow Pages), you must have the same entries for the `sablime` login in both `/etc/passwd` and `/etc/group` on the satellite machine and the host machine.
7. Make sure that you have followed all the steps in *Notes for All Multi-Machine Configurations*, above.
8. Make sure the Master Control Bin is accessible to your NFS satellite, i.e., make sure that it is mounted.

Using Sablime over TCP/IP

To use the Sablime TCP/IP multi-machine feature:

1. Set the `sabNET` variable to 0 (zero) on the host machine and to 7 (seven) on the satellite machines.
2. Set the `sabSERV` variable in the `$sabVAR` file on the satellite machine to `net_recv`.
3. Have the UNIX system administrator add Sablime service to the table of available TCP/IP services by inserting the following line in the `/etc/services` file on both the host and the satellite machines.


```
sablme xxxx/tcp
```

where *xxxx* is a number between 1524 and 6000. You can pick any number that is not already used by your host/satellite machines.



NOTE:

The number must be the same on the host and the satellite.

- Someone with root privileges must start the Sublime daemon (*sablmed*) on the host machine. The daemon is in the Sublime *bin* directory. From the two procedures below, select the appropriate one for your machine configuration. If you are not certain which procedure applies to your configuration, see your UNIX system administrator.



NOTE:

If no argument is given to *sablmed*, the default version control tool path is set as follows:

```
PATH=/usr/ucb:/bin:/usr/bin:/usr/sbin:/usr/ccs:\
      /usr/ccs/bin
```

This is the path that the *sablmed* daemon uses to find SBCS or SCCS.

If SBCS or SCCS is not in the standard location on your system, the UNIX system administrator must set the appropriate path for the *sablmed* daemon as an argument to *sablmed*. For example, if the SBCS commands are located in */usr/sbcs/bin* and the SCCS commands are located in */usr/add-on/sccs*, the UNIX system administrator must set up the *sablmed* path to include both locations:

```
/etc/in.sablmed PATH=/usr/sbcs/bin:/usr/add-on/sccs
```

The relative order of the SBCS and SCCS paths is irrelevant.



CAUTION:

If you change the PATH variable for in.sablmed, you must specify the full path to both SBCS and SCCS commands.

Enabling the *sablmed* daemon when TCP/IP does not have the *tcplisten* daemon:



NOTE:

Different Unix and Linux operating systems may require somewhat different procedures for such things as automatically starting the Sublime daemon. Please refer to your local system administrator and/or documentation for further assistance. These instructions are directly applicable to Sun Solaris

systems. HPUX systems, for example, place a daemon startup script (having the text described below as being added to the rc.local script) into /sbin/init.d, and it is launched by having a symbolic link in /sbin/rc2.d.

Have the UNIX system administrator perform the following tasks:

- n Copy the `sablimed` executable to `/etc/in.sablimed`, where `/etc` is the directory that contains the other TCP/IP daemons (e.g., `rlogind`, `rexecd`, etc.).

- n Start the `sablimed` daemon by typing:

```
/etc/in.sablimed PATH
```

where *PATH* is the appropriate argument, if any.



CAUTION:

Make sure that the UNIX system TZ (Time Zone) environment variable is set appropriately.

- n Add Sablime service to the `/etc/rc.local` file by inserting a statement like the following:

```
if [-f /etc/in.sablimed]; then
    /etc/in.sablimed && (echo 'sablimed')\
    /dev/console
fi
```

Enabling the `sablimed` daemon when TCP/IP has the `tcplisten` daemon:

Have the UNIX system administrator perform the following tasks:

- n Copy the `sablimed` executable to `/etc`, where `/etc` is the directory that contains the other TCP/IP daemons (e.g., `rlogind`, `rexecd`, etc.).
- n Kill the currently running `tcplisten` process.
- n Add Sablime service to the `tcplisten` command in the appropriate `/etc/rc*/*` files.
- n Append *sablime* to the line containing the `/etc/tcplisten` command. For example:

```
/etc/tcplisten ftp telnet finger rlogin remsh rexec\
sablime
```

- n Start a new `tcplisten` by typing:

```
tcplisten services sablime
```

where *services* is the argument list of the services included in the former `tcplisten` process.



CAUTION:

Make sure that the UNIX system TZ (Time Zone) environment variable is set appropriately.

5. The host machine must have an IP address entry for the satellite machine in its `/etc/hosts` file. The satellite machine must have an IP address entry for the host machine in its `/etc/hosts` file. If you are running Network Information Services (formerly Yellow Pages), `/etc/hosts` may not be on your local machine.

Example

The satellite machine name is `alvis`; the IP address is `136.18.3.1`. The entry in the host `/etc/hosts` file is:

```
136.18.3.1 alvis
```

The host machine name is `mozart`; the IP address is `136.4.118.99`. The entry in the satellite `/etc/hosts` file is:

```
136.4.118.99 mozart
```

6. If your host machine name is an alias for another machine, set the `sabDKFB` variable. For example:

```
sabDKFB=official_host_name
```

where *official_host_name* is the official name of the host machine in the `/etc/hosts` file.

An example is shown in the figure below.

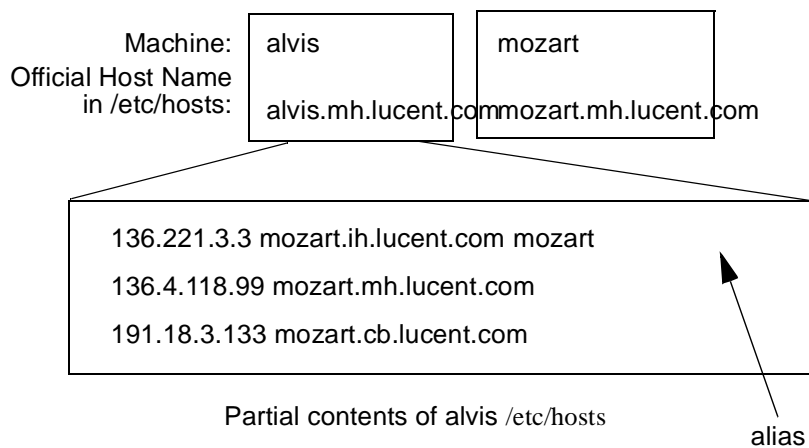


Figure 2-8. Sample TCP/IP Setup

In the `alvis /etc/hosts` file, `mozart` is an alias for `mozart.ih.lucent.com`. To make sure that the correct host machine is identified, set `sabDKFB` to the full name of the correct host, i.e.,

```
sabDKFB=mozart.mh.lucent.com
```

Otherwise, `alvis` will try to connect to `mozart.ih.lucent.com`.

If you are running Network Information Services (formerly Yellow Pages), the following command:

```
ypcat hosts
```

will display IP address, domain, and alias information for all hosts in your domain.

7. Make sure that you have followed all the steps in *Notes for All Multi-Machine Configurations* above.

Common TCP/IP Satellite Error Messages

Table 2-3 contains explanations of the common error messages that account for the majority of problems encountered in multi-machine mode.

Table 2-3. Multi-Machine Error Messages

Error Message	Diagnosis
\$sabHOST: Connection Refused	<code>in.sablimed</code> is not running on the host.
Login incorrect	The satellite user's login does not exist in the <code>/etc/passwd</code> file on the host, or the user's <code>/etc/passwd</code> entry does not contain a password.
Restricted login	The satellite user's login does not exist in the <code>/etc/passwd</code> file on the host, or the user's <code>uid</code> is less than 11 in the <code>/etc/passwd</code> entry.
No Remote Directory	The satellite user's home directory does not exist on the host but is defined in the <code>/etc/passwd</code> file.

Configuring Sablime for External MR Communications

This section describes how to configure Sablime so that two or more products can exchange MR information. (For details about the feature, see the section *Using the External MR Commands* in the *User's Guide*.)

Sablime can be configured to communicate MR information using any of the following communication protocols:

- n IPC (Inter-Process Communication).

- n TCP/IP
- n UUCP



NOTE:

The network/machine combinations currently supported by Sablime are available from the Sablime hotline.

The initial setup necessary to allow Sablime products to exchange MR information over each of the above communication protocols is described in the following sections.



WARNING:

When creating the .BIN and .EMR files mentioned in the following sections, be careful never to leave an extra space at the end of a line.

Setting Up an Inter-Process Communication Link

Inter-Process Communication (IPC) is used when communication is established between two Sablime products on the same machine. The products may or may not use the same instance of Sablime. The following three examples illustrate various possible configurations.

Example 1: Same machine, different instance Suppose you want to set up a communication link between product A and product B. Product A and product B are not in the same instance but are on the same machine. A sample Sablime environment for each product appears in the figure below.

sabHOST:	ma	
sabPROD:	A	B
sabMCB:	/u1/sablime/bin	/u2/sablime/bin
sabGDB:	/u1/sablime/gdb	/u2/sablime/gdb

Product A's Environment Product B's Environment

Figure 2-9. Environment for IPC Example 1

1. Set up a .EMR file in your MCB directory. The .EMR file should contain information about the location of the Sablime Global Database directory in the following format:

`sablime;product_name;path_to_GDB`

Example:

.EMR file for product A

```
sablime;A;/u1/sablime/gdb
```

.EMR file for product B

```
sablime;B;/u2/sablime/gdb
```

2. Set up a .BIN file in your MCB directory. The .BIN file should contain information about the product with which you intend to communicate in the following format:

```
sablime;other_product_name;full_path_to_other_product_MCB
```

Example:

.BIN file for product A

```
sablime;B;/u2/sablime/bin
```

.BIN file for product B

```
sablime;A;/u1/sablime/bin
```

3. Use the ES subcommand of the setrel command to set the configuration for product A. (For detailed information on the fields on this screen, see the manual page for setrel (ES) in Chapter 6, *The Administrative Commands*.)

```
logid:sablme   Sablime Configuration Management System v5.2   06/04/00
effid:sablme   Administrative System Command                   09:43:21

      External Communication (ES) Relation

            Function: add___
      External Project: sablime_____
      External Product: B_____
      Remote Machine: ma_____
            Network Type: 6
      Remote Program: rcv_msgs_____
            Parameters: _____

      Do you want to communicate MR Status Changes: n

      Accept: _           Nochange: _           Status1: _
      Approve: _         Spawned: _           Status2: _
      Assign: _          Submit: _           Status3: _
      Commit: _          Study: _           Status4: _
      Defer: _           Unaccept: _          Status5: _
      Do you want to communicate MR state at link time: n
      Do you want two-way mrnote communication: n
```

*For products on the same machine, the Remote Machine field entry is the name of the machine on which both products run.

Figure 2-10. IPC Example 1: setrel ES Screen for Product A

4. Use the ES subcommand of the setrel command to set the configuration for product B.

```
logid:sablme   Sablime Configuration Management System v5.2   06/04/00
effid:sablme   Administrative System Command                 09:44:21

      External Communication (ES) Relation

      Function: add___
      External Project: sablime_____
      External Product: A_____
      Remote Machine: ma_____
      Network Type: 6
      Remote Program: rcv_msgs_____
      Parameters: _____

      Do you want to communicate MR Status Changes: n

      Accept: _      Nochange: _      Status1: _
      Approve: _     Spawned: _      Status2: _
      Assign: _      Submit: _      Status3: _
      Commit: _      Study: _      Status4: _
      Defer: _       Unaccept: _     Status5: _
      Do you want to communicate MR state at link time: n
      Do you want two-way mrnote communication: n
```

*For products on the same machine, the Remote Machine field entry is the name of the machine on which both products run.

Figure 2-11. IPC Example 1: setrel ES Screen for Product B

This completes the set-up for this example.

Example 2: Same machine, same instance Suppose you want to set up a communication link between product C and product D. Both product C and product D are in the same Sablime instance (i.e., they share the same MCB). A sample Sablime environment for each product appears in the figure below.

sabHOST:	mc	
sabPROD:	C	D
sabMCB:	/u3/sablme/bin	
sabGDB:	/u3/sablme/gdb	

Machine mc

Figure 2-12. Environment for IPC Example 2

1. Set up the .EMR file in your MCB directory. Since both products share the same Global Database, the .EMR file in /u3/sablme/bin must contain two lines of information: one for product C, the other for product D:

```
sablme;C;/u3/sablme/gdb
sablme;D;/u3/sablme/gdb
```

2. Set up the .BIN file in your MCB directory. Again, since both products share the same MCB, the .BIN file in /u3/sablme/bin must contain two lines of information: one for product C, the other for product D:

```
sablme;C;/u3/sablme/bin
sablme;D;/u3/sablme/bin
```

3. Use the ES subcommand of the setrel command to set the configuration for product C. (For detailed information on the fields on this screen, see the manual page for setrel (ES) in Chapter 6, *The Administrative Commands*.)

```
logid:sablme   Sablime Configuration Management System v5.2   06/04/00
effid:sablme   Administrative System Command                   09:45:21

      External Communication (ES) Relation

            Function: add___
      External Project: sablime___
      External Product: D_____
      Remote Machine: mc_____
      Network Type: 6
      Remote Program: rcv_msgs___
      Parameters: _____

      Do you want to communicate MR Status Changes: n

      Accept: _           Nochange: _           Status1: _
      Approve: _         Spawned: _           Status2: _
      Assign: _          Submit: _           Status3: _
      Commit: _          Study: _           Status4: _
      Defer: _           Unaccept: _          Status5: _

      Do you want to communicate MR state at link time: n
      Do you want two-way mrnote communication: n
```

Figure 2-13. IPC Example 2: setrel ES Screen for Product C

4. Use the ES subcommand of the setrel command to set the configuration for product D.

```

logid:sablme   Sablime Configuration Management System v5.2   06/04/00
effid:sablme   Administrative System Command                 09:46:21

      External Communication (ES) Relation

      Function: add___
      External Project: sablime_____
      External Product: C_____
      Remote Machine: mc_____
      Network Type: 6
      Remote Program: rcv_msgs_____
      Parameters: _____

      Do you want to communicate MR Status Changes: n

      Accept: _      Nochange: _      Status1: _
      Approve: _     Spawned: _      Status2: _
      Assign: _      Submit: _      Status3: _
      Commit: _      Study: _      Status4: _
      Defer: _       Unaccept: _     Status5: _

      Do you want to communicate MR state at link time: n
      Do you want two-way mrnote communication: n
    
```

Figure 2-14. IPC Example 2: setrel ES Screen for Product D

This completes the set-up for this example.

In some applications, many products within the same instance need to communicate with each other. If you follow the above example, you may find duplicate information in your .EMR and .BIN files. You can simplify the .EMR and .BIN files as follows:

- n .EMR: If all the third fields contain the same information, i.e., all products share the same Global Database, you can replace all of those lines with one line in your .EMR file. For example:

```
__ALL;__ALL;/u3/sablme/gdb
```

This line implies that all the products use the same Global Database. If you have any product that does not use this Global Database, you cannot use the simplified format.

- n .BIN: If all the third fields contain the same information, i.e., all products share the same MCB, you can replace all the lines with one line in your .BIN file. For example:

```
__ALL;__ALL;/u3/sablime/bin
```

This line implies that all the products use the same MCB. If you have any product that does not use this MCB, you cannot use the simplified format.

Example 3: Same machine, different instances, same MCB Suppose you want to set up a communication link between products E and F. Products E and F are not in the same Sablime instance but are on the same machine and share the same MCB. A sample Sablime environment for each product appears in the figure below.

sabHOST:	mc	
sabPROD:	E	F
sabMCB:	/u3/sablime/bin	
sabGDB:	/u1/sablime/gdb	/u2/sablime/gdb

Machine mc

Figure 2-15. Environment for IPC Example 3

1. Set up the .EMR file in /u3/sablime/bin with two lines of information, one for product E, the other for product F:

```
sablime;E;/u1/sablime/gdb
sablime;F;/u2/sablime/gdb
```

2. Set up the .BIN file in your MCB directory. Since both products share the same MCB, the .BIN file in /u3/sablime/bin may contain two lines of information, one for product E, the other for product F, or a single line as indicated in Example 2, above.

Example:

```
sablime;E;/u3/sablime/bin
sablime;F;/u3/sablime/bin
```

or

```
__ALL;__ALL;/u3/sablime/bin
```

3. Use the ES subcommand of the setrel command to set the configuration for product E. (For detailed information on the fields on this screen, see the manual page for setrel (ES) in Chapter 6, *The Administrative Commands*.)

```
logid:sablme   Sablime Configuration Management System v5.2   06/04/00
effid:sablme   Administrative System Command                   09:47:21

      External Communication (ES) Relation

                Function: add__
      External Project: sablime_____
      External Product: F_____
      Remote Machine: mc_____
                Network Type: 6
      Remote Program: rcv_msgs_____
                Parameters: _____

      Do you want to communicate MR Status Changes: n

      Accept: _      Nochange: _      Status1: _
      Approve: _     Spawned: _      Status2: _
      Assign: _      Submit: _      Status3: _
      Commit: _      Study: _      Status4: _
      Defer: _       Unaccept: _     Status5: _

      Do you want to communicate MR state at link time: n
      Do you want two-way mrnote communication: n
```

For products on the same machine, the Remote Machine field entry is the name of the machine on which both products run.

Figure 2-16. IPC Example 3: setrel ES Screen for Product E

4. Use the ES subcommand of the setrel command to set the configuration for product F.

```
logid:sablme   Sablime Configuration Management System v5.2   06/04/00
effid:sablme   Administrative System Command                 09:48:21

      External Communication (ES) Relation

      Function: add___
External Project: sablime_____
External Product: E_____
      Remote Machine: mc_____
      Network Type: 6
      Remote Program: rcv_msgs_____
      Parameters: _____

      Do you want to communicate MR Status Changes: n

Accept: _      Nochange: _      Status1: _
Approve: _     Spawned: _      Status2: _
Assign: _      Submit: _      Status3: _
Commit: _      Study: _      Status4: _
Defer: _       Unaccept: _     Status5: _

Do you want to communicate MR state at link time: n
Do you want two-way mrnote communication: n
```

For products on the same machine, the Remote Machine field entry is the name of the machine on which both products run.

Figure 2-17. IPC Example 3: setrel ES Screen for Product F

This completes the set-up for this example.

Setting Up a TCP/IP Communication Link

This section describes how to set up a TCP/IP link for external MR communications (Sablme-to-Sablme communications).

As an example, suppose you want to set up a communication link between product A (on machine ma) and product B (on machine mb). A sample Sablime environment for each product appears in the figure below.

sabHOST:	ma	mb
sabPROD:	A	B
sabMCB:	/u1/sablime/bin	/u2/sablime/bin
sabGDB:	/u1/sablime/gdb	/u2/sablime/gdb

Product A's Environment Product B's Environment

Figure 2-18. Environment for TCP/IP Example

1. Set up a .EMR file in the MCB directory on host ma and host mb. The .EMR file should contain information about the location of the Sablime Global Database directory in the following format:

`sablime;product_name;path_to_GDB`

Example:

.EMR file for product A

`sablime;A;/u1/sablime/gdb`

.EMR file for product B

`sablime;B;/u2/sablime/gdb`

2. Set up a .BIN file in the MCB directory on host ma and host mb. The .BIN file should contain information about the product with which you intend to communicate in the following format:

`sablime;other_product_name;full_path_to_other_product_MCB`

Example:

.BIN file for product A

`sablime;B;/u2/sablime/bin`

.BIN file for product B

`sablime;A;/u1/sablime/bin`

3. Use the ES subcommand of the setrel command to set the configuration for product A.

```
logid:sablme   Sablime Configuration Management System v5.2   06/04/00
effid:sablme   Administrative System Command                 09:49:21

      External Communication (ES) Relation

      Function: add___
      External Project: sablime_____
      External Product: B_____
      Remote Machine: mb_____
      Network Type: 7
      Remote Program: rcv_msgs_____
      Parameters: _____

      Do you want to communicate MR Status Changes: n

      Accept: _      Nochange: _      Status1: _
      Approve: _     Spawned: _      Status2: _
      Assign: _      Submit: _      Status3: _
      Commit: _      Study: _      Status4: _
      Defer: _       Unaccept: _     Status5: _

      Do you want to communicate MR state at link time: n
      Do you want two-way mrnote communication: n
```

For products on the same machine, the Remote Machine field entry is the name of the machine on which both products run.

Figure 2-19. TCP/IP Example: setrel ES Screen for Product A

4. Use the ES subcommand of the setrel command to set the configuration for product A.

```
logid:sablme   Sablime Configuration Management System v5.2   06/04/00
effid:sablme   Administrative System Command                 09:50:21

      External Communication (ES) Relation

              Function: add__
      External Project: sablime_____
      External Product: A_____
              Remote Machine: ma_____
              Network Type: 7
      Remote Program: rcv_msgs_____
              Parameters: _____

      Do you want to communicate MR Status Changes: n

      Accept: _           Nochange: _           Status1: _
      Approve: _         Spawned: _           Status2: _
      Assign: _          Submit: _           Status3: _
      Commit: _          Study: _           Status4: _
      Defer: _           Unaccept: _          Status5: _

      Do you want to communicate MR state at link time: n
      Do you want two-way mrnote communication: n
```

For products on the same machine, the Remote Machine field entry is the name of the machine on which both products run.

Figure 2-20. TCP/IP Example: setrel ES Screen for Product B

5. Start the UNIX Sablime daemon as in step 4 in *Using Sablime over TCP/IP*, above.

This completes the set-up for this example.

Setting Up a UUCP Communication Link

This section describes how to set up a uucp link for external MR communications (Sablime-to-Sablime communications).

As an example, suppose you want product A and product B to communicate with each other. A sample Sablime environment for each product appears in the figure below.

sabHOST:	ma	mb
sabPROD:	A	B
sabMCB:	/u1/sablime/bin	/u2/sablime/bin
sabGDB:	/u1/sablime/gdb	/u2/sablime/gdb
database owner	a_sablime	b_sablime

Product A's Environment Product B's Environment

Figure 2-21. Environment for UUCP Example

1. Set up a .EMR file in the MCB directory on host ma and host mb. The .EMR file should contain information about the location of the Sablime Global Database directory in the following format:

sablime;product_name;path_to_GDB

Example:

.EMR file for product A

sablime;A;/u1/sablime/gdb

.EMR file for product B

sablime;B;/u2/sablime/gdb

2. Set up a .BIN file in the MCB directory on host ma and host mb. The .BIN file should contain information about the product with which you intend to communicate in the following format:

sablime;other_product_name;full_path_to_other_product_MCB

Example:

.BIN file for product A

```
sablime;B;/u2/sablime/bin
```

.BIN file for product B

```
sablime;A;/u1/sablime/bin
```

3. Use the ES subcommand of the `setrel` command to set the configuration for product A.

```
logid:sablime   Sablime Configuration Management System v5.2   06/04/00
effid:sablime   Administrative System Command               09:51:21

      External Communication (ES) Relation

                Function: add___
      External Project: sablime_____
      External Product: B_____
      Remote Machine:  mb_____
                Network Type: 8
      Remote Program:  rcv_msgs_____
      Parameters:    b_sablime_____

      Do you want to communicate MR Status Changes: n

      Accept:  _      Nochange:  _      Status1:  _
      Approve:  _     Spawned:  _      Status2:  _
      Assign:  _     Submit:  _       Status3:  _
      Commit:  _     Study:  _       Status4:  _
      Defer:  _     Unaccept:  _     Status5:  _

      Do you want to communicate MR state at link time: n
      Do you want two-way mrnote communication: n
```

For products on the same machine, the Remote Machine field entry is the name of the machine on which both products run.

Figure 2-22. UUCP Example : setrel ES Screen for Product A

In the *Parameters* field, you must enter the Sablime database owner's login.

4. Use the ES subcommand of the setrel command to set the configuration for product B.

```

logid:sablme   Sablime Configuration Management System v5.2   06/04/00
effid:sablme   Administrative System Command                 09:52:21

      External Communication (ES) Relation

            Function: add___
External Project: sablime_____
External Product: A_____
      Remote Machine: ma_____
            Network Type: 8
      Remote Program: rcv_msgs_____
            Parameters: a_sablme_____

      Do you want to communicate MR Status Changes: n

Accept: _      Nochange: _      Status1: _
Approve: _     Spawned: _      Status2: _
Assign: _      Submit: _       Status3: _
Commit: _      Study: _        Status4: _
Defer: _       Unaccept: _     Status5: _

Do you want to communicate MR state at link time: n
      Do you want two-way mrnote communication: n
  
```

For products on the same machine, the Remote Machine field entry is the name of the machine on which both products run.

Figure 2-23. UUCP Example: setrel ES Screen for Product B

In the *Parameters* field, you must enter the Sablime database owner's login.

5. Set up the Sablime uucp daemon. There are three files in the SABDM directory in the MCB, two of which must be modified. The three files, pickmsg.sh (a shell script), execd.m.sh (a shell script), and suucpdm (a binary) and the necessary modifications are described below.

- n pickmsg.sh. Copy pickmsg.sh to pickmsg. Edit pickmsg and set the ksh environment by entering the full path of ksh, and the value of the sabMCB variable. Modify the command line shown below (in pickmsg) as shown in the example that follows.

```
ksh -c $sabMCB/SABDM/execd.m >>.cust.log 2>&1
```

Example (pickmsg for ma):

```
/usr/tools/bin/ksh -c /u1/sablme/bin/SABDM/execd.m\  
>>.cust.log 2>&1
```

Save your modifications to pickmsg.



NOTE:

It is not necessary to redirect output to log files; instead, you can direct the output to /dev/null.

- n execd.m.sh. Copy execd.m.sh to execd.m. Modify the command line shown below (in execd.m) as shown in the example that follows.

```
. full_path_of_Sablme_setup_shell/sablme generic > \  
/dev/null 2>&1  
full_path_of_Sablme_bin_directory/SABDM/suucpdm >> \  
.cust.log 2>&1
```

Example (execd.m for ma):

```
. /u1/sablme/bin/sablme a1 > /dev/null 2>&1  
/u1/sablme/bin/SABDM/suucpdm >> .cust.log 2>&1
```

Save your modifications to execd.m.

- n suucpdm. This is a binary file and should not be modified.

6. Start up the daemon:

- n Set the permissions for pickmsg, execd.m, and suucpdm to 755 by entering the following command line:

```
chmod 755 pickmsg execd.m suucpdm
```

- n Log in as *sablme* and run pickmsg to start up the daemon.

Installing the End User Web Interface

⇒ **NOTE:**
The templates of the HTML and Perl scripts for this feature are in the \$sabMCB/SABDM directory. The two daemons, sabldm and sabrepdm, are in the \$sabMCB directory.

⇒ **NOTE:**
This feature allows a user to establish a Web Interface to their own Sablime instance, permitting *their* customers to create MRs for review. This is not the "Web Sablime" interface, nor does it permit the creation of MRs against Sablime itself.

To install the End User Web interface to Sablime, you must do the following:

1. Create a UNIX account `sabmail`.
2. Create an PTS ID `sabmail` for the mail process.
3. Create an PTS ID `cust` for the review command.
4. Add `sabmail` in your `.usrid` license file. (`cust` can be an unlicensed user.)
5. Put `setup`, `varfile`, `pickupmsg`, and `execustrep` in `~sabmail`, and make `sabmail` the owner of those processes.
6. Customize all the above shell scripts as needed.
7. Customize the HTML files for your own project. You will have to supply the correct ACTION field, and create the system and sub-system files (the `*.html` files) needed by your project. Follow the instructions in the HTML files.
8. Customize the PERL 4.0 script for your own project. For example, you may want to change the "Thank you note", the "sabmail email address", the copy-to mail message, and the confirmation message for your project.
9. Put the HTML files and PERL scripts at the location on your web server that will allow your customers to create MRs or obtain an MR report from your Sablime database using a Web browser.
10. Add `field_mod` & `field_enh` to the *Category* field for your create command. (This will be done by the v5.0 conversion script.)
11. Set the `sabMAIL` environment variable. For example:

```
export sabMAIL=/var/spool/mail/sabmail
```
12. Tell your customers the URL that will allow them to access your HTML files.
13. For testing purposes, use your URL to create an MR and request an MR report.

14. Log in as `sabmail` on your UNIX machine.
15. Check to see that your mail box contains two messages that you have just created.
16. Run your `pickupmsg` shell script under the `sabmail` login. Then check to see that a type 15 message has been created under your `$sabGDB/rq` directory and a type 16 message under your `$sabGDB/cron` directory.
17. Run your `execustrep` script; the requested report should be sent to the originator's e-mail address.
18. Run the `review` command to see the type 15 message and decide whether you would like to create an MR in your Sablime database or remove the message from the message queue.
19. Periodically remove the log files `$sabGDB/cron/*.log` and `~sabmail/*.log`.

Pre- and Post-Trigger Routines

The pre/post-hook feature allows user-defined programs to govern the behavior of Sablime commands. By using this feature, a Database Administrator can write programs that take special actions based on the keyword/value pairs supplied by a Sablime user. As long as they conform to a basic application interface, pre/post-hooks can be written in any programming language. Conceptually, a prehook is executed before any changes are made to the Sablime databases, and can be used to flag project-specific error conditions, while a posthook is executed after a command updates the Sablime databases, and can be used to perform any additional processing connected to the given command.

To enable pre/post-hooks, first create a directory called `hooks` in the Local Control Bin (`$sabLCB`) of your Sablime instance. All hooks follow the same naming convention: `pre<command>` for a prehook, and `post<command>` for a posthook. For example, a file named `preaccept`, with `execute` permission in the `$sabLCB/hooks` directory, is the proper name (and file permissions) for the prehook associated with the `accept` command. All pre/post-hooks in the `$sabLCB/hooks` directory must be executable by the `sablime` login. If a hook adheres to the proper naming convention, but does not have `execute` permission, it will be skipped.

In Sablime v6.0, the following commands are enabled for pre/post-hooks:

`accept`, `addgsrsc`, `addisrsc`, `approve`, `assign`, `closemr`, `common`, `create`, `depend`, `edget`, `edput`, `fcreate`, `getversion`, `mmnote`, `propose`, `qmr`, `reject`, `reserve`, `sget`, `smerge` (pre-hook only), `spawnmr`, `study`, `submit`, `testpass`, `unaccept`, `uncommon`, `unedget`, `unedput`, `unreserve`

The input to a pre/post-hook is a set of `name=value` pairs, identical to the `name=value` pairs accepted by any command in no-prompt mode. When either a prehook or posthook is executed, the full set of `name=value` pairs that the user

entered for the Sablime command is passed to the hook's standard input (see `stdio(3S)`). Blank or empty fields are not passed.



CAUTION:

Pre/post-hooks rely on internal keywords for standardization, since customized FTD data is product-specific, but the Local Control Bin is not.

Each hook can then read the *name=value* pairs from `stdin`, and take any action deemed appropriate. The interprocess communication scheme used for the implementation is achieved via pipes (see `pipe(2)`). The hook mechanism detects errors based on the exit code of each pre/post-hook; exit codes of 0 indicate success, while exit codes greater than 0 indicate failure. When a prehook fails, the associated Sablime command makes no modifications to the Sablime databases.

Upon failure, a pre/post-hook can relay an error message back to the calling command using the hook's `stdout`. Error messages can be in one of the two following formats, both of which should be output as one line (i.e., a message followed by a newline):

1. <up to 70 character error message>
2. <up to 70 character error message>|<internal key>



NOTE:

Error messages are only examined on hook failure. Any message sent by a hook that exits normally (with an exit code of 0) will be ignored.

For prehooks, the supplied error message will be displayed at the bottom of the command screen in curses mode (or in Web Sablime). In no-prompt mode, the message will be sent to `stderr` by the Sablime command.

For posthooks, the supplied error message will be displayed to the command's `stderr` in both the curses and no-prompt modes, and to the Command Output frame in Web Sablime.

When a posthook fails in either curses or no-prompt mode, the associated Sablime command will have an exit code of 2. Also, when a prehook fails for a command in no-prompt mode, it will have an exit code of 2. Normally, Sablime commands have an exit code of 1 upon failure, so this information can be used to flag hook-specific errors in command calls.



NOTE:

In the hook's error message, characters that are neither alphanumeric nor punctuation characters are converted to spaces.

When a prehook's error message is followed by both a '|' and a valid internal key value (as in format 2 above), special behavior occurs when the command is run in curses mode. In addition to displaying the error message at the bottom of the command screen, the user is automatically positioned on the input field associated with the supplied internal key. Using this mechanism, a user will not have to manually reposition the input cursor to change an erroneous input field, since it can be automatically determined by the prehook. If an invalid internal key is specified, it will be ignored.



NOTE:

When a hook's exit code is greater than 0, all messages that the hook sent to `stderr` are saved in the product's `dbawarn` file. Care should be taken to monitor hook behavior in this case, to avoid unnecessarily large `dbawarn` files.

The hook implementation uses both `fork(2)` and `exec(2)` to execute hook code. Therefore, hooks written in shell should follow the standard interpreter file syntax. For example, a hook that uses `ksh(1)` constructs should have `#!/bin/ksh`, or something similar, as its first line. For further information, see `exec(2)`.

Listed below are three basic examples that demonstrate the general usage of the pre/post-hook feature.

1. The following shell, `precreate`, will only allow users to create MRs detected in the current development stage, `system_test`:

```
#!/bin/ksh
while read I
do
    INTERNAL_KEY=`print $I|cut -f1 -d='`
    VALUE=`print $I|cut -f2 -d='`
    DEVELOPMENT_STAGE="system_test"

    if [ "$INTERNAL_KEY" = "pd" ]
    then
        if [ "$VALUE" != "$DEVELOPMENT_STAGE" ]
        then
            print "Phase detected must match\
development stage, $DEVELOPMENT_STAGE|pd"
            exit 1
        fi
    fi
done
```

2. The following C program, `presubmit.c`, will not let users submit MRs after 6 P.M:

```
#include <stdio.h>
#include <sys/types.h>
#include <time.h>
```

```
main(int argc, char **argv)
{
    struct tm      *tm_ptr;
    time_t        the_time;

    (void)time(&the_time);
    tm_ptr = localtime(&the_time);

    /* prevent users from submitting MRs after 6\ P.M. */
    if (tm_ptr->tm_hour >= 18) {
        printf("MRs must be submitted before 6\ P.M.\n");
        exit(2);
    }
    exit(0);
}
```

3. The following shell, `postgetversion`, writes both the date and the `name=value` pairs used for each execution of the `getversion` command:

```
#!/bin/ksh

LOGDIR=/usr/tmp/logs
LOGFILE=$LOGDIR/getversion.log

update_log()
{
    print "***** getversion run on `date` *****"

    while read I
    do
        print $I
    done

    print "*****"
}

update_log >> $LOGFILE
exit 0
```

Contents

3	Customizing the User Interface	1
n	Essential Product Customization	2
	Customizing the System, Release-Detected, Site, and Category Menu	3
	Cascading Menus	10
	Customizing Cascading Menus	12
n	Customizing Commands and Command Screens	16
	The ftd Command and the FTD Relation	17
	Changing the Fields of the FTD Relation	18
n	Customizing Pop-Up Selection Windows	29
	Reserved Groups	29
	Changing Reserved Groups	31
n	Customizing Templates	38
n	Defining New Fields	40
	The UDF Feature	42
	Customizing a UDF	44
	Customizing UDFs: An Example	46

Contents

After Sablime has been installed and initialized for your product, you can customize it in a variety of ways. Customization is done on the product level; it affects only the product for which relations or data are altered. This chapter discusses the ways in which Sablime can be customized, beginning with things that must be customized to make Sablime meaningful for your product and proceeding to the following:

- n customizing commands and command screens
- n customizing Pop-Up Selection Windows
- n customizing templates
- n defining new fields

**CAUTION:**

Sablime gives the Database Administrator great power to alter the interface with Sablime (e.g., remove a field from a given screen; change field names, sizes, or locations; change menu choices and defaults) by using the `ftd` and `setgroup` commands. Such changes can affect the internal functions of Sablime and seriously degrade the performance of the system. Do not change anything unless this manual expressly permits changes to that item. If you are in doubt about whether an item can be changed, contact the Sablime hotline before proceeding.

Essential Product Customization

Sablime expects you to customize the menus for several dozen fields. These fields collect and categorize information about how your product is constructed, where and when problems are detected, root causes of problems, the source and nature of modification requests, and why certain requests are not adopted. In addition, several commands accept "user defined fields," which can gather additional data as defined by your project.

Used correctly, these fields together will help give project managers a good view of the state of your product as it evolves. It is up to someone on your project (possibly you) to decide on appropriate values for these fields, to educate your Sablime users to use them effectively, and to select, or create, suitable reports for managers and developers. There are many books and articles available on configuration management, project management, software metrics, and software process improvement; these may be a good place to start. Lucent's Software Technology Center and other organizations also offer consultation and jump-starts in these areas.

As installed by `initsab`, your product will have a working, but nondescript, set of values for these fields. Four fields are particularly bland as installed, and we begin by showing you how to customize them for your product. That is, you will walk through the mechanics, the syntax, of customization. To come up with meaningful values, you may need to consult with your project manager or architect.

The table below lists these fields, the commands that use them, and the Product-Level Reserved Group that supplies the values for them.

Table 3-1. Fields, Commands, and Groups

Field	Command	Group
System	create fcreate mredit review spawnmr	_SYS
Release Detected	create fcreate mredit review setrel spawnmr	_RELS
Site	create fcreate mredit review	_SITES
Category	create fcreate mredit review	_ICAT

The following sections describe how to customize the Pop-Up Selection Windows for these four fields.

**NOTE:**

In all the following examples, you must either be logged in as *sablime*, or must be in the DBA group for the product you are working with. If you are not in the DBA group, you cannot simply run `su sablime`; you must literally log in as *sablime* or as someone in the product's DBA group.

Customizing the System, Release-Detected, Site, and Category Menus

The four fields in Table 3-1, and the other fields described so far, are each governed by a so-called Product-Level Reserved Group. When a user executes a command that uses one of these fields, the menu in the Pop-Up Selection Window shows an entry for each member of the group. You may attach comments to entries to help users decide which entry to select. You may also define a default selection.

These fields are customized using the `setgroup` command. To use `setgroup` to customize one of these fields, you need to know the name of the group that governs the field. For the System, Release, and Site fields, the groups are, respectively, `_SYS`, `_RELS`, and `_SITES`. (See Appendix B for a listing of all the Product-Level Reserved Groups, together with the values and defaults for these groups, as initially installed by `initsab`.) For Product-Level Reserved Groups, you can add your own entries, and delete Sablime's initial entries, as appropriate.

⇒ NOTE:
We will postpone for now going into the details of how groups are stored, and the way in which groups, menus, and the FTD relation interact. See the discussion and diagrams in *Customizing Pop-Up Selection Windows*, below, for information about these matters.

See the examples below for more specific information about customizing these menus. Note that in all the examples, you can use the Command Line interface to accomplish the same result. See the description of the `setgroup` command in the Sablime *User's Guide* for more information.

⇒ NOTE:
Group entries should not contain spaces except after the comment character (`~`). Comments are allowed only in group type `other`. Although groups accept spaces, the `query` and `report` commands do not parse spaces. The Sablime convention is to use an underscore character (`_`) to separate words in a group entry.

Example 1: Customizing the System Menu

As installed, Sablime displays `none`, `other`, and `all` in the Pop-Up Selection Window for the *System* field and populates the field with `none` as the default. No other entries are allowed. The instructions below show you how to customize the menu and specify an appropriate default.

⇒ NOTE:
The *System* field can be used to contain any hierarchical specifications that are relevant to your product. In general, field names need not control the kind of data the user enters.

Use the `setgroup` command to edit the `_SYS` group. On the command line, enter:

```
setgroup
```

If you are using the Curses Forms interface, the screen shown in Figure 3-1 will be displayed. Enter the requested data in the first field and press RETURN. Since this is an existing group, Sablime populates the *Group Owner* and *Group Type* fields with their current values. Press RETURN twice to navigate to the *Member*

File field, leaving the *Owner* and *Type* values as is. At the *Member File* field, press RETURN again to display the file containing the group members and default information in your editor.

```
logid:xxxxxx      Sablime Configuration Management System v5.0      03/17/96
effid:sablime     Administrative System Command                    08:18:46

                Add/Delete Members to/from Groups

                Group Name:  _SYS_____
                Group Owner:  sablime_____
                Group Type:  other_____

Member File: _____
Copy To:    _____
```

Figure 3-1. setgroup Command for _SYS

The file looks like this:

```
^none
other
all
```

The default is specified by a caret (^).

Modify the file to specify the systems appropriate for your product. For the Sablime product, the entries might look like this:

```
admin
docmgmt
head inforet
^lib
mrmgmt
qamgmt
srcmgmt
xmrmgmt
all
none
```

The default specifier (^) has been used to make lib the default for this group.

When you leave the editor and confirm the command, the changes you have made take effect.

You can create cascading menus from one or more of the systems you entered. The System, Subsystem, and Module menus should be planned in concert. See *Customizing Cascading Menus*, below, for how to attach subsystems to systems and modules to subsystems.

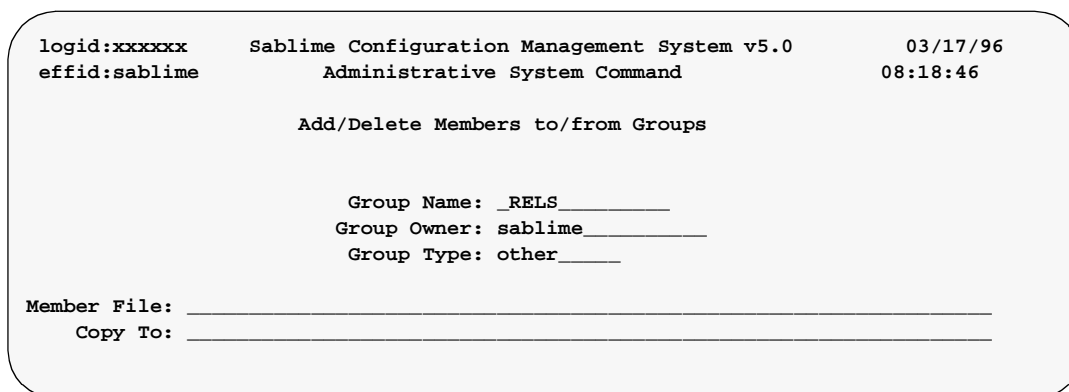
Example 2: Customizing the Release-Detected Menu

As installed, Sablime displays ^not_applicable in the Pop-Up Selection Window for the *Release Detected* field and populates the field with this entry as a default. No other entry is allowed. The instructions below show you how to customize the menu and specify an appropriate default.

Use the `setgroup` command to edit the `_RELS` group. On the command line, enter:

```
setgroup
```

If you are using the Curses Forms interface, the screen shown in Figure 3-2 is displayed. Enter the requested data in the first field and press RETURN. Since this is an existing group, Sablime populates the *Group Owner* and *Group Type* fields with their current values. Press RETURN twice to navigate to the *Member File* field, leaving the *Owner* and *Type* values as is. At the *Member File* field, press RETURN again to display the file containing the group members and default information in your editor.



```
logid:xxxxxx      Sablime Configuration Management System v5.0      03/17/96
effid:sablime     Administrative System Command                      08:18:46

                Add/Delete Members to/from Groups

                Group Name:  _RELS_____
                Group Owner:  sablime_____
                Group Type:  other_____

Member File: _____
Copy To:    _____
```

Figure 3-2. `setgroup` Command for `_RELS`

The file looks like this:

```
^not_applicable
```

The default is specified by a caret (^).

Modify the file to specify the releases appropriate for your product. For example, the entries might look like this:

```
3.0  
3.0.2  
3.1  
3.2  
4.0  
^5.0
```

The default specifier (^) has been used to make 5.0 the default for this group.

When you leave the editor and confirm the command, the changes you have made take effect.

Example 3: Customizing the Site Menu

As installed, Sablime displays a menu with only one entry (^not_applicable) in the Pop-Up Selection Window for the *Site* field. The instructions below show you how to customize the menu and specify an appropriate default.

Use the `setgroup` command to edit the `_SITES` group. On the command line, enter:

```
setgroup
```

If you are using the Curses Forms interface, the screen shown in Figure 3-3 is displayed. Enter the requested data in the first field and press RETURN. Since this is an existing group, Sablime populates the *Group Owner* and *Group Type* fields with their current values. Press RETURN twice to navigate to the *Member File* field, leaving the *Owner* and *Type* values as is. At the *Member File* field, press RETURN again to display the file containing the group members and default information in your editor.

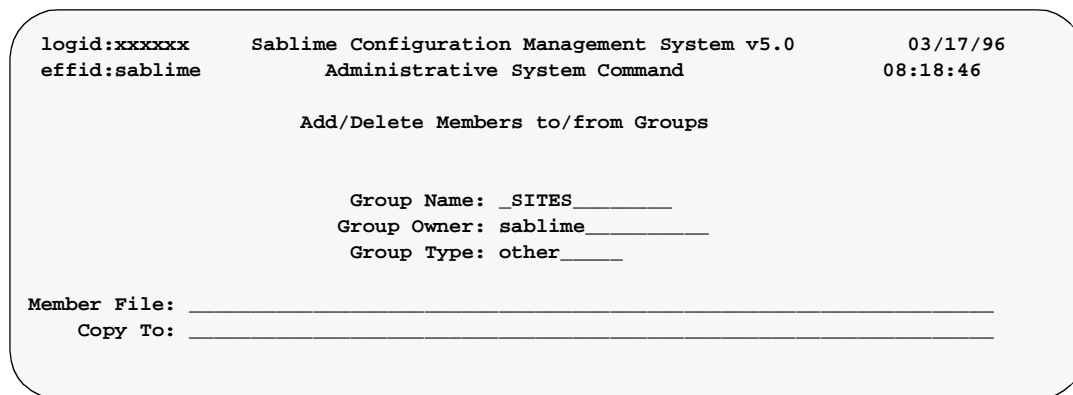


Figure 3-3. setgroup Command for `_SITES`

The file contains one entry:

```
^not_applicable
```

Modify the file to specify the sites appropriate for your product. The revised file might look like this:

```
London
^New_York
Tokyo
Frankfurt
Taipei
```

The default specifier (^) has been used to make `New_York` the default for this group.

When you leave the editor and confirm the command, the changes you have made take effect.

Example 4: Customizing the Category Menu

When Sablime is installed for your product, a Product-Level Reserved Group, `_ICAT`, is created for the *Category* field. The `_ICAT` group is accessed by the `create`, `fcreate`, and `review` commands.

When the user executes `create`, `fcreate`, or `review`, the information displayed in the Pop-Up Selection Window for the *Category* field depends on the entries in the

_ICAT group. To make Sablime more meaningful for your product, you should customize the menu items in this group. To do this, use the `setgroup` command to modify the members of the group and create a customized menu. See the example below for more specific information about customizing this menu.

As installed, Sablime displays a menu of available categories for the *Category* field and populates the field with a specified default. No other entry is allowed. See Appendix B for the members and defaults created for the _ICAT group at Sablime installation. The instructions below show you how to customize the menu and specify an appropriate default.

Use the `setgroup` command to edit the _ICAT group. On the command line, enter:

setgroup

If you are using the Curses Forms interface, the screen shown in Figure 3-4 is displayed. Enter _ICAT in the first field and press RETURN. Since this is an existing group, Sablime populates the *Group Owner* and *Group Type* fields with their current values. Press RETURN twice to navigate to the *Member File* field, leaving the *Owner* and *Type* values as is. At the *Member File* field, press RETURN again to display the file containing the group members and default information in your editor.

```
logid:xxxxxx      Sablime Configuration Management System v5.0      03/17/96
effid:sablime     Administrative System Command                    08:18:46

                Add/Delete Members to/from Groups

                Group Name: _ICAT _____
                Group Owner: sablime _____
                Group Type: other _____

Member File: _____
Copy To: _____
```

Figure 3-4. `setgroup` Command for _ICAT

The file looks like this:

```
inspection
review
code_reading
usage
^testing
document_reading
demo
discussion
maintenance
new_feature
```

The default is specified by a caret (^).

Modify the file to specify the MR categories appropriate for your configuration management process.

```
^customer_found
dev_found
st_found
other
```

The default specifier (^) has been used to make `customer_found` the default for this group.

When you leave the editor and confirm the command, the changes you have made take effect.

Cascading Menus

The other essential customization involves what are known as cascading menus. To understand what these are it is necessary to understand the concept of dynamically built menus, which is described in the following section. Then there is a discussion of how to customize cascading menus and there are examples of such customization.

Dynamically Built Menus

Sablime builds some menus dynamically based on information stored in the Sablime database or entries made in previous fields of the same command or in a different command.

For example, when you run the `addgen` command, the first screen asks you to select appropriate MR classes for the MRs in your generic. Your selections are stored internally. Later, when users on your project run the `fcreate` or `accept` command, the *MR Class* field supplies a pop-up window whose entries are supplied from your original *MR Class* choices.

In the `review` command, the menu built for the *Message Number* field depends on the messages present at that moment in the receive queue.

Other dynamically built menus, called Cascading Menus, are built primarily on the data entered in a previous field.

Cascading Menus and Fields

Certain fields in Sablime use Pop-Up Menus that vary their selections depending on the value selected for some previous field. For example, the menu for the Subsystem field depends on the entry in the System field. These menus are called cascading menus, and the fields that use them are called cascading fields.

The cascading fields in Sablime are:

- n Subsystem (depends on System)
- n Module (depends on Subsystem)
- n Subclass (depends on Class)
- n Subtype (depends on Type)
- n Root Cause Subcategory (depends on Root Cause)
- n Non-detection Cause Subcategory (depends on Non-detection Cause)

A cascading field is also called a lower level key, and the field it depends on is called an upper level key. You will need to remember these terms because the CAS screen of the `setrel` command uses them.



NOTE:

You can add a member to an upper level key that does not use the cascade effect. But if you do want to use it, you must create a group for that new group member first and then associate the two with the CAS subcommand of the `setrel` command. If an associated group does not exist, the commands that use these cascading menus will fail.

Cascading fields can be used to contain any hierarchical specifications that are relevant to your product. Field names do not have to direct your choice of the kind of data to be entered.

⇒ NOTE:
 If you choose not to display an upper level key, you should not display the cascading field that depends upon it.

Using this terminology, here are the cascading fields again, together with a listing of the commands which use them, in tabular form:

Table 3-2. Cascading Fields

Lower Level Key	Command	Upper Level Key
Subsystem	create fcreate mredit review spawnmr	System
Module	create fcreate mredit review spawnmr	Subsystem
Subclass	accept fcreate mrgedit spawnmr	Class
Subtype	accept fcreate mrgedit spawnmr	Type
Root Cause Subcategory	mrgedit submit	Root Cause
Non-Detection Cause Subcategory	mrgedit submit	Non-Detection Cause

Customizing Cascading Menus

When Sablime is installed for your product, no groups are created for cascading fields because Sablime has no way of knowing what your product options are. For each system, you must decide whether subsystems will be used. For each field choice with cascading menus, use the `setgroup` command to create a group of items to be displayed in the cascading field Pop-Up Selection Window when the field choice is specified. For example, if you have three systems defined, a series of subsystems can be defined for each of those systems using the cascade effect.

See the examples below for more specific information about customizing the cascading menus. These examples show how to supply cascading menus for the *System* to *Subsystem* and *Subsystem* to *Module* fields. Follow similar procedures for any of the fields shown in Table 3-2. In all the examples, you can use the Command Line interface to obtain the same result. See the description of the `setgroup` command in the Sablime *User's Guide* for more information.

Example 1 : Creating a Cascading Menu for the Subsystem Field

After you have modified the *System* menu, you may want to specify subsystems for some or all systems used by your project. As installed, Sablime does not include groups to be used as menus for the *Subsystem* field. If no groups are created, the subsystem field is skipped and no entry is allowed. The instructions below show you how to customize the menu to allow selection of a subsystem for each system and how to specify an appropriate default.

Use the `setgroup` command to create a group of subsystems called `_LIBSUB` for the system *lib*. On the command line, enter:

setgroup

If you are using the Curses Forms interface, the screen shown in Figure 3-5 is displayed. Enter the requested data in the first three fields. At the *Member File* field, press RETURN again.

```

logid:xxxxxx      Sablime Configuration Management System v5.0      03/17/96
effid:sablime    Administrative System Command                      09:15:29

                Add/Delete Members to/from Groups

                Group Name:  _LIBSUB_____
                Group Owner: xxxxxx_____
                Group Type:  other_____

Member File: _____
Copy To:    _____
    
```

Figure 3-5. setgroup Command for _LIBSUB

A temporary file is opened in your editor and a template displayed explaining how members should be listed in the file. Delete the template information and enter the

subsystems you want to allow for the lib system. For the Sablime product, your entries might look like this:

```
libADS
libCK
libDB
libCOM
^all
none
```

The default specifier (^) has been used to make all the default for this group.

When you leave the editor and confirm the command, the new group `_LIBSUB` is created in the Active Database.

Now use the `setrel` command to establish the `_LIBSUB` group as the subsystem group for the `lib` system. On the command line, enter:

```
setrel
```

Select the **CAS** relation from the first screen. The screen shown in Figure 3-6 is displayed. Select the items shown below from the menus, and confirm the screen.

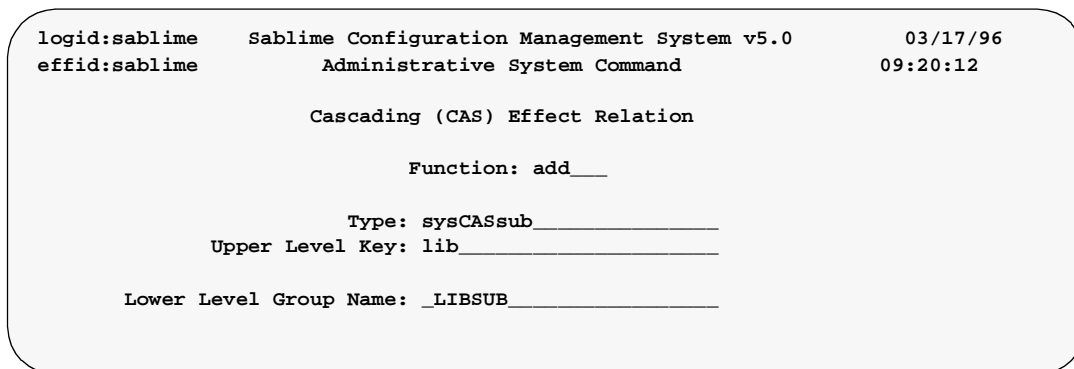


Figure 3-6. CAS Subcommand for lib/_LIBSUB

When the **lib System** is chosen on the `create`, `fcreate`, `review`, and `mredit` commands, the values entered in the `_LIBSUB` group will be displayed as the menu for the *Subsystem* field.

Example 2: Creating a Cascading Menu for the Module Field

Suppose your system tracks information about the universe. Then planets might be one choice for the *System* field, the individual planets could be the choices for the *Subsystem* field, and the moons could be the choices for the *Module* field.

Sablime allows you to specify the cascade to the *Module* field as a two-level specification, i.e., by specifying *system:subsystem* in the *Upper Level Key* field in the CAS subcommand of the *setrel* command.

To set up the cascade to the moons of Mars, first create a group called *marsmoons* with the *setgroup* command. This group has two members:



Phobos
Deimos

Then run the *setrel* command and specify the CAS relation. For the CAS relation, your entries would look like those in Figure 3-7.

```
logid:sablime    Sablime Configuration Management System v5.0    04/08/96
effid:sablime    Administrative System Command                15:09:43

                Cascading (CAS) Effect Relation

                Function: add_____

                Type: subCASmod_____
                Upper Level Key: planets:Mars_____
                Lower Level Group Name: marsmoons_____
```

Figure 3-7. CAS subcommand for planets:Mars/marsmoons

When the **planets** *System* and the **Mars** *Subsystem* are specified, the values entered in the *marsmoons* group are displayed as the menu for the *Module* field.

Customizing Commands and Command Screens

This section describes in detail the changes you can make to Sablime commands and Sablime command screens. Under appropriate conditions, you can control the aspects of the user interface listed below; see the sections indicated for further details. You may:

- n choose whether or not to display a field on the screen (see *Field 3 - Flags*)



NOTE:

If you choose not to display an upper level key, you should not display the cascading field that depends upon it.

- n make entry of data mandatory for a given field (see *Field 3 - Flags*)
- n change the verbose help message associated with a given field (see *Field 4 - Text*)
- n specify that a Pop-Up Selection Window be associated with a given field (see *Field 5 - Values*)
- n specify the maximum length for a field (see *Field 5 - Values*)
- n specify numerous details relating to the command screens (see *Field 7 - HMI*), among them
 - the row and column number at which field input may begin
 - the character used to display data entry length
 - the maximum number of characters the user can enter for a field
 - the relative positions of the field prompt and the field input
 - the screen prompt for each field
 - the number of menu items a user can select from a Pop-Up Selection Window
- n choose the keyword used for the field in the Command Line interface (see *Field 8 - External Keyword*)
- n remove or change certain default values (*Run-Time Expansion Keywords*)

All these changes can be made by using the `ftd` command. For additional information about this command, see the `ftd` manual page in Chapter 6, *The Administrative Commands*.

The ftd Command and the FTD Relation

The elements that Sablime uses to collect data are called fields. Field descriptions are stored in the FTD relation in the Active Database. They are described and modified with the ftd command, which updates the FTD relation. (See *The FTD Relation* in Chapter 5, *The Sablime Databases*.)



NOTE:

Changes can only be made to some of the fields in some of the commands in the FTD relation. General guidelines about acceptable changes are included here. FTD information is per-product. All generics for a product share the same FTD data, and every product has its own FTD data.

Like other Sablime relations, the FTD relation consists of a set of records, each divided into fields. The fields are ordered and named, and every record in a relation has the same number of fields. The FTD relation, as shipped, contains over 1500 records. Each FTD record uniquely represents a field for a Sablime command and defines its characteristics.

All the FTD records needed for all the Sablime commands are provided with the Sablime software. Each time you install a Sablime product with initsab, these records are loaded into the product's Active Database. Do not create any new FTD records; the Sablime commands will not recognize them. Do not delete any FTD records; this will render the corresponding commands inoperable.

Each Sablime relation is implemented as a UNIX directory, and the records reside in flat files called tuple files in that directory. Each tuple file has a two-letter name. There are about 100 of these tuple files in the FTD directory, for example. Each record is assigned a tuple file by hashing the record's first field value to a tuple file name. The dbhash command prints the hash value for an input string; you can use it to pinpoint the tuple file for a record if you know the record's first field value. For FTD records, the first field is the command name. This means that all FTD records for a command reside in the same tuple file.

Each record occupies one line, and its fields are separated by semicolons. Always be careful, when editing records, not to delete or insert any semicolons.

All FTD records have eight fields. For example, as created by initsab, the FTD record for the Generic field in the accept command is as follows:

```
accept;g;y,y,n;Generic:,16,,2002;1,67,__GENERIC;;\  
7,12,_,128,left,Generic: ,0,0;g
```



NOTE:

For more information, see *The FTD Relation* in Chapter 5, *The Sablime Databases*.

Changing the Fields of the FTD Relation

The eight fields in an FTD record are shown in the table below. This section describes the fields and their subfields as well as the changes permitted and their effects.

Table 3-3. The Fields of the FTD Record

Field	Contents	Type of Data
1	Command	Command name
2	Internal Keyword	Name of field that is used internally by Sablime
3	Flags	See Table 3-4
4	Text	See Table 3-5
5	Values	See Table 3-6
6	Keyload	Menu entries and defaults
7	HMI	See Table 3-7
8	External Keyword	External short field name used in the keyword=value Command Line interface



NOTE:

Every Sablime relation has a single primary key, i.e., a minimal set of fields whose values by definition uniquely identify each record in the relation. The fields in the primary key determine what object the record represents, and the remaining fields describe properties of that object. In Sablime relations, the fields of the primary key always come at the beginning of the record. For FTD, in particular, the primary key is the first two fields. In every Sablime relation, the first field is also the hash key, used to determine which tuple file the record resides in, as described on the previous page. Do not confuse the hash key with the primary key.

Field 1—Command

The *Command* field contains the name of the command in which the field is used. This is the name entered on the command line to invoke the command.



CAUTION:

Do not change this field. If the contents of this field are changed, the command will not run.

Field 2—Internal Keyword

The *Internal Keyword* field contains the name of the field used internally by Sablime. (See *Field 8 -External Keyword*, below.)



CAUTION:

Do not change this field. If the contents of this field are changed, the command will not run.

Field 3—Flags

The *Flags* field is divided into three comma-separated subfields, as shown in Table 3-4.

Table 3-4. Flags (Field 3)

Subfield	Contents	Type of Data
1	Display Flag	y or n
2	Mandatory Flag	y or n
3	Hideable Flag	y or n

1. The first subfield, *Display Flag*, indicates whether the field is to be displayed and used for data collection. If this flag is set to **n**, the field is not displayed in the Curses Forms interface or the Graphical User interface. Any entry made for the field in the Command Line interface is ignored.



NOTE:

If the *Hideable Flag* (subfield 3) is **n**, the *Display Flag* is populated with **y** and protected from change.

2. The second subfield, *Mandatory Flag*, indicates whether the field is to be considered mandatory for your product. This affects all interfaces.
 - n If the flag is set to **y** and no default is available, the user must enter data in this field and cannot advance to the next field or process the command without such an entry.
 - n If the flag is set to **n** and no default is available, the user can enter data or press RETURN without entering data to skip the field.
 - n If a default exists, it is used to populate the field regardless of the *Mandatory Flag*.



NOTE:

If the *Mandatory Flag* was set to **n** when Sablime was delivered, you can change it to **y**. But if it was set to **y**, do not change it.

3. The third subfield, *Hideable Flag*, indicates whether the field can be hidden in the Curses Forms interface and the Graphical User interface, i.e., whether the field is necessary for Sablime to operate correctly. This subfield is protected from change.



CAUTION:
Do not change this subfield.



NOTE:
By default, all the User-Definable Fields (UDFs) have the following attributes: Display Flag=n, Mandatory Flag=n, and Hideable Flag=y.

Field 4—Text

The *Text* field is divided into four comma-separated subfields, as shown in Table 3-5.

Table 3-5. Text (Field 4)

Subfield	Contents	Type of Data
1	Command Line Help	Text for the Command Line interface help display
2	Not Used	empty
3	Not Used	empty
4	Verbose Message Number	Number of the message displayed if user requests verbose help

1. The first subfield, *Command Line Help*, represents the literal string of characters that is displayed for the field in the Command Line interface help screen.
2. The second subfield is not used in the current release and may be left empty.
3. The third subfield is not used in the current release and may be left empty.
4. The fourth subfield, *Verbose Message Number*, is a four-character name for the verbose help message file that provides help information for the field in the Curses Forms interface and the Graphical User interface on the X Window System. (This subfield is not used by the Graphical User interface on the PC.) You can customize the verbose help message by changing the number to point to a different file containing a customized help message. Be sure that the file is in the directory named in the `sabVHELP` environment variable (see *Defining Environment Variables* in Chapter 2, *Installing Sablime*).



NOTE:
Although a customized help message could be created by editing the file named in this field, any changes made to the file would have to be made again when subsequent releases of Sablime were delivered.

Field 5—Values

The *Values* field is divided into three comma-separated subfields, as shown in Table 3-6.

Table 3-6. Values (Field 5)

Subfield	Contents	Type of Data
1	Field Type	1, 2, or 3
2	Field Length	Number of fill characters displayed to indicate the length of the field in the Curses Forms interface; if left/right buffer size=0, this number also indicates the maximum number of characters allowed for the field.
3	Default	Run-time Expansion Keyword or character string that populates the field if user does not enter data; this subfield is valid only for field types 1 and 3.

1. The first subfield, *Field Type*, designates the nature of the field in the Curses Forms interface and the Graphical User interface.

Type	Nature
1	A string field that allows the user to enter a string of characters.
2	A Pop-Up Selection Window field that uses a Sablime-Level or Product-Level Reserved Group or a product-defined group to populate a menu of acceptable user entries.
3	A Pop-Up Selection Window field that populates a menu dynamically when the command is executed.



CAUTION:

Field types 1 and 2 can be changed with caution. Fields of type 3 are protected and cannot be changed. Do not change another type field to type 3 because you will not be able to change it back.



NOTE:

UDFs allow only types 1, and 2. If you choose type 2, a Reserved Group must exist before changing to type 2.

2. The second subfield, *Field Length*, represents the maximum number of characters displayed by the fill character designated (see field 7) in the Curses Forms interface. If the *Left/Right Scrolling Buffer* (see field 7) field contains a zero (0), the *Field Length* also represents the maximum number of characters allowed for the field. This field can be changed.
3. The third subfield, *Default*, defines (for field type 1) the default entry that is made if the user does not enter data for the field; it is ignored for field types 2 and 3. The default entry must not contain more characters than the field length limit set unless the *Left/Right Scrolling Buffer* entry is greater than the field length limit. The default entries work as follows:
 - n If the field is empty, no default is entered.
 - n If the field contains a character string, the string populates the field.
 - n If the field contains a Run-time Expansion Keyword, the expansion takes place and the resulting data populates the field. (See the section *Run-time Expansion Keywords*, below.)

This field can be changed.

Field 6—Keyload

This field is applicable only to field type 2. The *Keyload* field contains either a Sablime-Level or Product-Level Reserved Group, or a product-defined group. If the field type is not specified as 2, the *Keyload* field is ignored. The group members defined in the group are used to populate the pop-up menu for the field when the command is executed from the Curses Forms interface or the Graphical User interface.



NOTE:

Menus are not displayed for the *Selection Criteria* field in the report command in the delivered version of Sablime. If you want menus to be displayed in these fields, use the *ftd* command to enter the appropriate group names in this field for each of the selection criteria fields in report. Keep in mind that if this change is made, users cannot enter group names in the affected fields.

Field 7—HMI

The HMI (Human-Machine Interface) field is divided into eight comma-separated subfields, as shown in Table 3-7..

Table 3-7. HMI (Field 7)

Subfield	Contents	Type of Data
1	Row Number	Vertical starting place for field input from the top of the screen
2	Column Number	Horizontal starting place for field input from the left of the screen
3	Fill Character	Character to be used to display data entry length on the screen
4	Left/Right Scrolling Buffer	Maximum number of characters that can be entered for this field; 0 (zero) if the buffer is not used; must be between 10 and 1024; if nonzero, must be greater than <i>Field Length</i> entry in field 5, subfield 2
5	Screen Label Position	Position of the field name in relation to data entry display: left , right , above , or below
6	Screen Label	Prompt displayed for user entry on the screen
7	Attribute	Used to control the screen display; this field must always contain a 0 (zero)
8	Number of Menu Choices Allowed	Number of menu entries the user can select at one time for data entry

1. The first subfield, *Row Number*, designates the vertical position from the top of the screen for the start of the field prompt in the Curses Forms interface. The entry can be a number from 0 to 23.
2. The second subfield, *Column Number*, designates the horizontal position from the left side of the screen for the start of the field prompt in the Curses Forms interface. The entry can be a number from 1 to 79.



NOTE:

Changing the row or column in the above subfields changes the location of the field on the screen, but it does not change the order in which the fields are visited. Consequently, changes in the location of a field could prove confusing to users.

3. The third subfield, *Fill Character*, designates the character to be used to display data entry length on the screen in the Curses Forms interface. Currently, all Sablime commands use the underscore character (`_`) for this purpose. You can change this character.
4. The fourth subfield, *Left/Right Scrolling Buffer*, designates the maximum number of characters that you can enter for this field in the Curses Forms interface and the Graphical User interface. Data entry length is limited only by the *Field Length* entry described in field 5 (*Values*), subfield 2, if the *Left/Right Scrolling Buffer* field is 0.

When you make an entry for the *Left/Right Scrolling Buffer* field, Sablime creates a buffer to hold data entered and, when in the Curses Forms interface, the entry area scrolls left to right to accommodate more characters than the *Field Length* would normally allow.

This field can be changed, but the number entered must be greater than the *Field Length* designation (in subfield 2 of the *Values* field). Remember that the initial data scrolls off the screen as more data is entered. Although the maximum number of characters allowed is 1024, do not use a number greater than 256 in this field because it may confuse the user and make data entry more difficult. It can also skew data positions on reports.

5. The fifth subfield, *Screen Label Position*, designates the position of the field name in relation to the data entry display in the Curses Forms interface.

In most cases, the entry in this field is **left**. The other choices are **right**, **above**, and **below**.

In other words, the screen label Group Name is to the left of the input line. At times, use of another position designator may be necessary.

6. The sixth subfield, *Screen Label*, represents the literal string of characters that is displayed to prompt the user in the Curses Forms interface and the Graphical User interface.



NOTE:

The screen label can be changed to coordinate with the external keyword change; however, the field name change is not reflected in the query and report commands unless the field is a UDF.

7. The seventh subfield, *Attribute*, controls the attributes of the screen display in the Curses Forms interface. This field must always contain a 0 (zero).



CAUTION:

Do not change this field.

8. The eighth subfield, *Number of Menu Choices Allowed*, specifies the number of menu entries the user can select as data entry in the Curses Forms interface or the Graphical User interface. This value allows the user to select more than one item from a Pop-Up Selection Window.

If the field has been specified as field type 1, this field must contain a 0 (zero). For field types 2 or 3, an integer from 1 to the number of items in the menu can be specified.

Field 8—External Keyword

The *External Keyword* field represents the short name of the field that is used in the Command Line interface. The user can enter the external keyword on the command line with a value that is populated when the command is executed.

When Sablime is initially installed, all the external keywords have the same value as the internal keywords. (See *Field 2 - Internal Keywords*, above.) Although you can change any external keyword with the `ftd` command, the only fields you are encouraged to rename are the UDF fields. (See *Defining New Fields*, below.)



NOTE:

For the UDF keywords, the changes you make can be extended to the query and report commands; for other external keywords, they cannot be. See the `ftd` manual page in Chapter 6, *The Administrative Commands*, for details.



CAUTION:

Do not change the external keyword prompt.



NOTE:

Since the documentation makes extensive use of external keywords, any change to an external keyword should be noted in the documentation.

Run-Time Expansion Keywords

A Run-time Expansion Keyword is a Sablime reserved word that defines a default value when expanded during the execution of a Sablime command. Each keyword represents a variable in the Sablime environment. Sablime v5.0 uses 12 Run-time Expansion Keywords.

Each keyword name begins with a double underscore (__) followed by text in capital letters (*TEXT*). When the user presses RETURN in response to a request for input to a field that uses a Run-time Expansion Keyword, the keyword is expanded to produce the appropriate value.

The Run-time Expansion Keyword is entered in the FTD relation record when Sablime is installed. The only changes that can be made without affecting the integrity of Sablime are to remove the keyword from the FTD relation record and make the user enter the necessary information, or to replace the keyword with some other value that will serve as the default.



CAUTION:

Any other change to fields normally populated through the use of a Run-time Expansion Keyword is unsupported by Sablime, and the results cannot be predicted.

The keywords and resulting default values are explained in Table 3-8. The first column shows the keyword; the second describes the data that will be entered and the field in which it will appear; the third explains how the value is determined by Sablime.

Table 3-8. Run-Time Expansion Keywords

Keyword	Data Entered	Determination
__ADB	Full path to product's Active Database (ADB)	Determined by accessing the PR relation record of the current product and selecting the ADB Directory Path from the record.
__EFFID	Effective User ID (Screen Header)	Normally the effective user ID is <i>sablime</i> whenever you run a Sablime command. However, when you execute commands on a satellite in a multi-machine environment, your effective user ID is your login ID on the satellite and <i>sablime</i> on the host.
__GENERIC	Current Generic (<i>Generic</i>)	Determined by storing the generic entered by the user when setting up for Sablime. Looks at the <i>sabGEN</i> variable.
__IDB	Full path to product's Inactive Database (IDB)	Determined by accessing the PR relation record of the current product and selecting the IDB directory path from the record.
__MCB	Full path to product's Master Control Bin (MCB)	Determined by accessing the PR relation record of the current product and selecting the MCB directory path from the record.
__PROD	Current Product (<i>Product</i> or <i>Authorized Product</i>)	Determined by storing the generic entered by the user when setting up for Sablime and selecting the product for that generic. Looks at the <i>sabPROD</i> variable.
__PTSID	User's PTS ID (PTS ID)	Taken from the user's PTS record.
__REALID	User's login ID	Taken from the user's environment.

Table 3-8. Run-Time Expansion Keywords—Continued

Keyword	Data Entered	Determination
__RELDIR	Relative path of current directory (<i>Directory</i>)	Taken from the VPATH environment variable and the current working directory. If VPATH is not set, no value is entered.
__SDB	Full path to product's Source Database (SDB)	Determined by accessing the PR relation record of the current product and selecting the SDB directory path from the record.
__TODAY	Current date (<i>Date</i> fields)	Date returned by user's system.
__VPDEV	Path name of user's node (<i>Node</i>)	Determined by accessing the VPATH variable containing the user's node.

Example 1 : Removing Default Values

As installed, the default entry for the *Generic* field in the `accept` command is the name of the setup generic.

To remove the default:

1. In the Curses Forms interface, enter:
ftd
2. The FTD screen will be displayed. After you fill in the information in the first three fields, Sablime populates the rest of the screen and allows you to move the cursor through the screen to the fields to be modified.

```

logid:logid      Sablime Configuration Management System4 v5.0      07/07/96
effid:sablime    Administrative System Command                      09:09:37

                          Field Tracking Data Maintenance

Function: modify      Internal Key: g_____
Command: accept_____ External Key: g_____

Field Type: 1          Attribute: 0_
Mandatory: y          Prompt Padding: 16
Hideable: n           Prompt Position: left_
Display: y            L/R Scroll Size: 128_
Row Number: 7_        Fill Character: _
Col Number: 12        Popup Selections: 0__
Length: 67           Verbose Help Key: 2002

Screen Label: Generic: _____
Command Line Help: Generic: _____
Default Value: _GENERIC_____
Group/File: _____
Copy To: _____
                          Edit Another Field: _

```

Figure 3-8. ftd Screen for Generic Field in accept Command

3. Press RETURN to move the cursor through the fields until you reach the *Default Value* field. Delete the contents of this field and leave it blank, i.e. do not type anything, even a blank, in the field. There will no longer be a default for this field.

Example 2: Changing Default Values

As installed, the default entry for the *Generic* field in the `accept` command is the name of the setup generic.

To populate the field with a specified default generic (regardless of the setup generic):

Follow steps 1 and 2 from *Example 1 : Removing Default Values* above.

Press RETURN to move the cursor through the fields until you reach the *Default Value* field. Delete the contents of this field and replace them with the text to be used as a default (e.g., v6.0).

Customizing Pop-Up Selection Windows

Reserved Groups

Sablime-Level Reserved Groups and Product-Level Reserved Groups contain lists of values that Sablime uses to populate menus during the execution of a Sablime command. To change the Pop-Up Selection Windows, it is necessary to change the Reserved Groups. (See the following section for examples of how to do this.)

Each Sablime-Level Reserved Group name begins with a double underscore (__) followed by text in capital letters (*TEXT*), while each Product-Level Reserved Group begins with a single underscore (_) followed by text in capital letters (*TEXT*).

When Sablime is installed:

- The Reserved Group names are entered in the appropriate FTD relation records.
- The Reserved Group names are entered in the GRP relation and the members of each group are entered in the GRPM relation.

The FTD relation, GRP and GRPM relations, and the command menu in the Pop-Up Selection Window work together as shown in Figures 3-10 and 3-11.

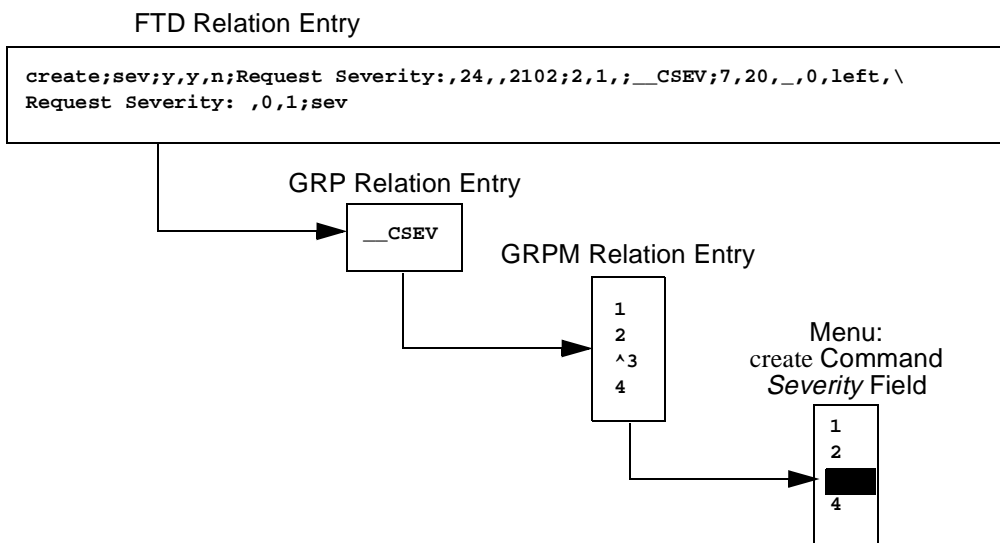


Figure 3-9. Sablime-Level Reserved Group Menus

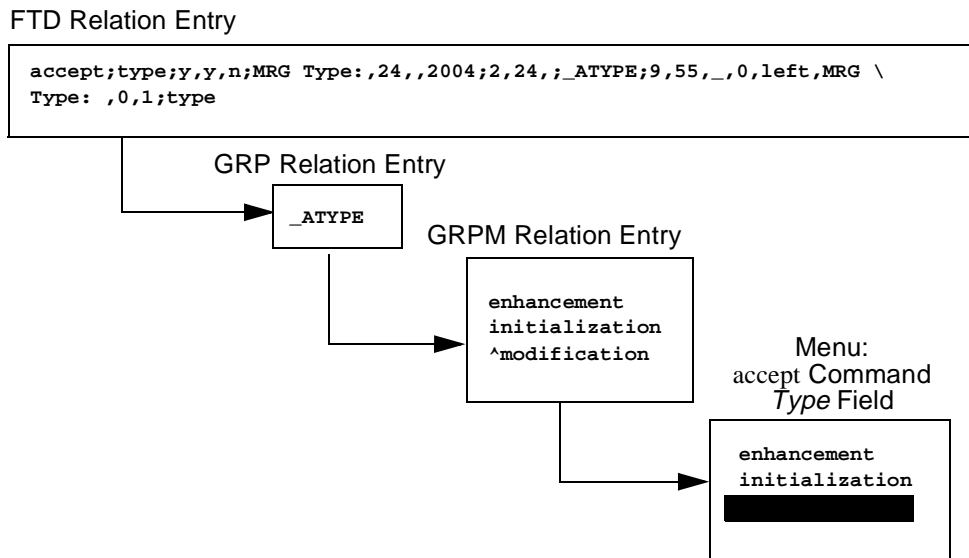


Figure 3-10. Product-Level Reserved Group Menus

Changing Reserved Groups

To customize Pop-Up Selection Windows, it is necessary to change the Reserved Groups (Sablime-Level or Product-Level) that determine their content. To do this, look at the FTD record for the field and command to determine which group defines the menu entries, then use the `setgroup` command to make the desired changes. You can:

- n Change the default. The default is specified by preceding the member by a caret (e.g., `^modification`). You can change it by moving the caret to the front of the appropriate selection. If more than one group entry is marked with a caret, the first one found is used as the default. A group without a caret has no default entry.
- n Change the order of the menu items in the group.
- n Add comments to the menu items by adding a tilde (~) after the item and entering an appropriate phrase. The maximum length of the Pop-Up Selection Window line (item and comment) is 70 characters. When the menu is displayed, the tilde is replaced by two spaces.
- n If the group is a Product-Level Reserved Group, add to or change the members of the group to make the selections more appropriate for your product. The `report` command allows users to sort and select by entries made from many of these menus.



CAUTION:

You may not add members to or change members of a Sablime-Level Reserved Group.

The groups, members, and standard default values for Sablime-Level Reserved Groups are shown in Appendix A, those for Product-Level Reserved Groups in Appendix B. The *Notes* column in each Appendix contains additional information concerning which commands use each group.

The following examples show how to make the changes to Pop-Up Selection Windows described above. The menus in the Pop-Up Selection Windows in the first three examples are determined by Sablime-Level Reserved Groups; the menu in the final example is determined by a Product-Level Reserved Group.

You can use the Command Line interface in any of the examples in this section to obtain the same result. See the description of the `setgroup` command in the *User's Guide* for more information.

Example 1 : Changing the Default in a Pop-Up Selection Window

As installed, Sablime populates the *Severity* field in the *create* command with the default severity 3. To change the default to 2:

1. Use either the *ftd* or the *ssql* command to locate the name of the reserved group for the *Severity* field in the *create* command. You can use *ssql* to print out the group name as follows:

```
ssql fvalue from FTD where prog.eq.create and\ intkey.eq.sev
```

Or can invoke *ftd* interactively and view the full information about the field, either entering the command and internal keyword on the *ftd* command line, as follows:

```
ftd fcn=view cmmnd=create intkey=sev
```

or choosing them from the menus that FTD will display.

In this case, the reserved group is `__CSEV`.

2. Use the *setgroup* command to edit the pop-up selections. On the command line, enter:

```
setgroup
```

3. If you are using the Curses Forms interface, the screen shown in Figure 3-11 will be displayed. Enter the requested data in the first field and press RETURN. Sablime populates the next two fields as you press RETURN for each. At the *Member File* field, press RETURN again to display the file containing the group members and default information in your editor.

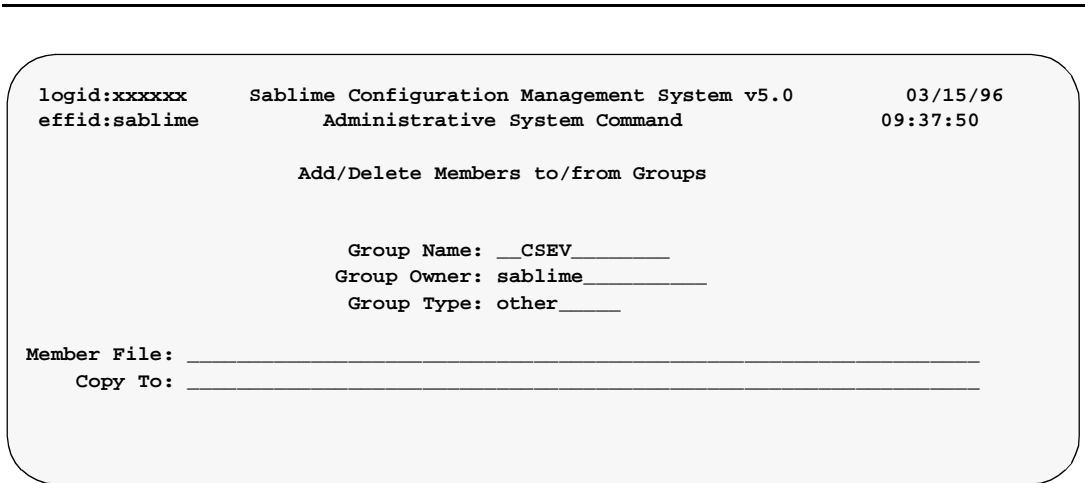


Figure 3-11. *setgroup* Command for `__CSEV`

The file looks like this:

```
1
2
^3
4
```

The default is specified by a caret (^). Use your editor to delete the caret in front of 3 and place it in front of the 2 so that the file looks like this:

```
1
^2
3
4
```

When you leave the editor and confirm the command, the changes you have made take effect.

Example 2 : Changing Order of Display in a Pop-Up Selection Window

As installed, Sablime displays a menu for the *Message Selection* field in the `sendmsgs` command. To change the order of the menu items:

1. Use either the `ftd` or the `ssql` command to locate the name of the reserved group for the *Message Selection* field in the `sendmsgs` command. As in the previous example, you can enter either

```
ssql fvalue from FTD where prog.eq.sendmsgs and\ intkey.eq.fcn
```

or

```
ftd fcn=view cmmnd=sendmsgs intkey=fcn
```

In this case, the reserved group is `__SEND`.

2. Use the `setgroup` command to edit the pop-up selections. On the command line, enter:

```
setgroup
```

3. If you are using the Curses Forms interface, the screen shown in Figure 3-13 will be displayed. Enter the requested data in the first field and press RETURN. Sablime populates the next two fields as you press RETURN for each. At the *Member File* field, press RETURN again to display the file containing the group members and default information in your editor.

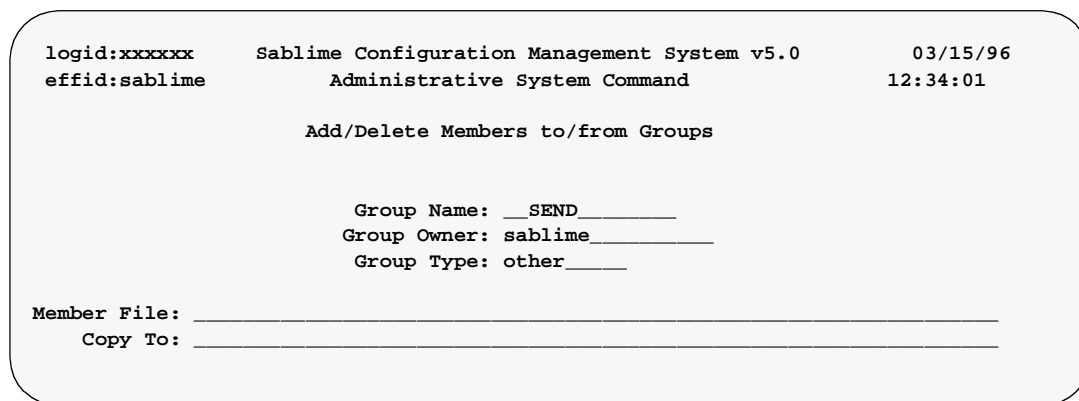


Figure 3-12. setgroup Command for __SEND

The file looks like this:

```
^all
group
individual
```

Use your editor to change the order of the items in the file so that the file looks like this:

```
group
individual
^all
```

When you leave the editor and confirm the command, the changes you have made take effect.

Example 3 : Adding Comments to a Pop-Up Selection Window

As installed, Sablime displays a menu for the *Severity* field used by the create command (and several other commands). To add comments to the menu items:

The tilde (~) is used for comments in Sablime groups of type other. In the menu, the tilde is replaced by two spaces. The maximum line length is 70 characters.

1. Use either the `ftd` or the `ssql` command to locate the name of the reserved group for the *Severity* field in the `create` command. As in the previous examples, you can enter either

`ssql fvalue from FTD where prog.eq.create and\ intkey.eq.sev`

or

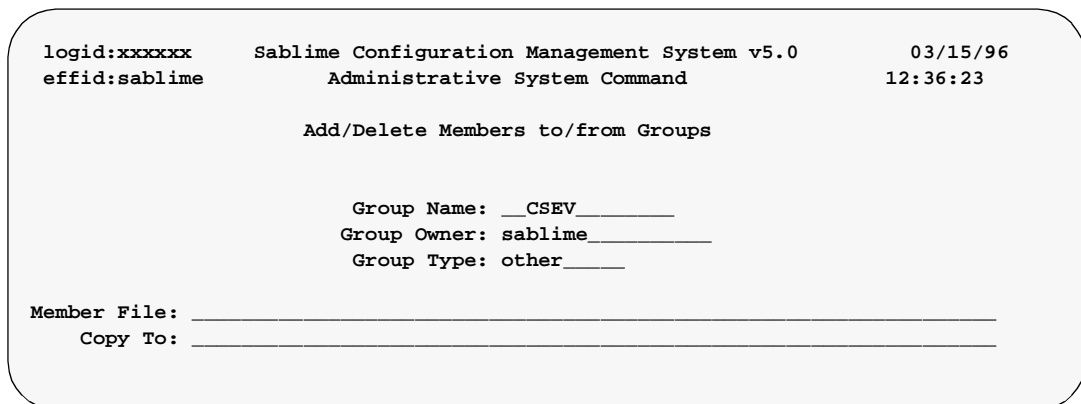
`ftd fcn=view cmmnd=create intkey=sev`

In this case, the reserved group is `__CSEV`.

2. Use the `setgroup` command to edit the pop-up selections. On the command line, enter

`setgroup`

3. If you are using the Curses Forms interface, the screen shown in Figure 3-13 is displayed. Enter the requested data in the first field and press RETURN. Sablime populates the next two fields as you press RETURN for each. At the *Member File* field, press RETURN again to display the file containing the group members and default information in your editor.



```
logid:xxxxxx      Sablime Configuration Management System v5.0      03/15/96
effid:sablime     Administrative System Command                    12:36:23

                Add/Delete Members to/from Groups

                Group Name: __CSEV_____
                Group Owner: sablime_____
                Group Type: other_____

Member File: _____
Copy To: _____
```

Figure 3-13. `setgroup` Command for `__CSEV` (comments)

The file looks like this:



```
1
2
^3
4
```

Use your editor to add comments to the items in the file so that the file looks like this:

```
A~Service interrupted
B~Service degraded but functioning
^C~Inconvenience to users
D~Minor problem
```

When you leave the editor and confirm the command, the changes made take effect. When the menu is displayed on the screen, it looks like this:

```
A Service interrupted
B Service degraded but functioning
>C Inconvenience to users
D Minor problem
```



CAUTION:

Severities are limited to a single alphanumeric character. The query command sorts only single-character severities.

Example 4: Changing Menu Items in a Pop-Up Selection Window

As installed, Sablime displays a menu for the *MR Type* field in the *accept* command. You can change the items in the menu to fit your project. To change the menu items:

1. Use either the *ftd* or the *ssql* command to locate the name of the reserved group containing the values and default for the *MR Type* field in the *accept* command. As in the previous examples, you can enter either

```
ssql fvalue from FTD where prog.eq.accept and\ intkey.eq.type
```

or

```
ftd fcn=view cmmnd=accept intkey=type
```

In this case, the reserved group is *_ATYPE*.

2. Use the *setgroup* command to edit the pop-up selections. On the command line, enter

```
setgroup
```

If you are using the Curses Forms interface, the screen shown in Figure 3-14 will be displayed. Enter the requested data in the first field and press RETURN. Sablime populates the next two fields as you press RETURN for each. At the

Member File field, press RETURN again to display the file containing the group members and default information in your editor.

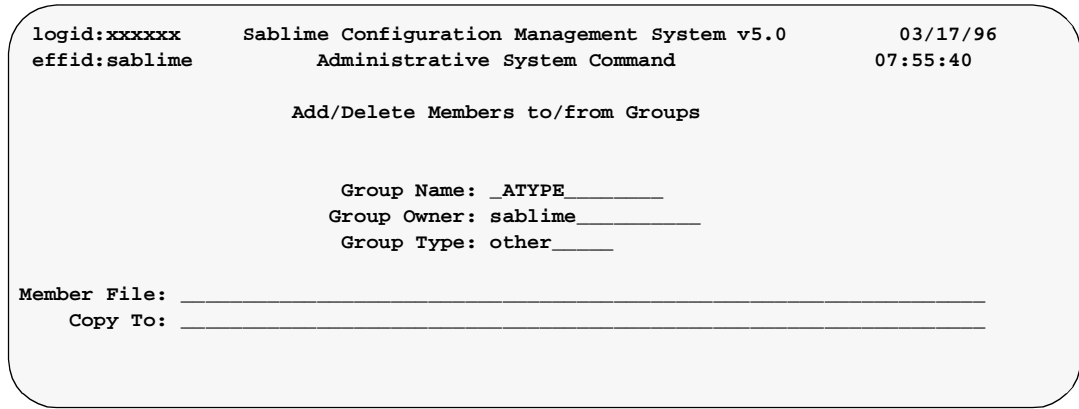
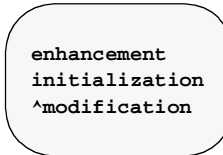


Figure 3-14. setgroup Command for _ATYPE

The file looks like this:



1. Use your editor to add to items in the file so that the file looks like this:



2. The default has also been changed from **modification** to **action_item**. When you leave the editor and confirm the command, the changes you have made will take effect.

Customizing Templates

Thirteen Sublime commands supply templates to instruct the user about the kind of information that should be entered or to explain actions being taken. The commands and the supplied text of the template for each are shown in the table below.

You can change the text of these templates by changing directories to the `adb/product/FILES/templates/sublime` directory and editing the appropriate file. Changes to templates are at the product level. The names of the template files are shown in Table 3-10.

Table 3-9. Command Templates

Command	Supplied Text
commit	<p>*** YOU ARE IN YOUR FAVORITE EDITOR *** PLEASE ENTER MR(S) TO BE COMMITTED — ONE MR NUMBER PER LINE</p>
create	<p>Please provide the following information: 1)Project Name: 2)Project Contact: 3)Project Phone Number: 4)Project EMAIL Address: 5)Description of MR:</p>
dbstop	<p>*** YOU ARE IN YOUR FAVORITE EDITOR *** THIS PRODUCT WILL BE STOPPED IF ANYTHING REMAINS IN THE STOP FILE AFTER EDITING. TO RESTART THE DATABASE, EXECUTE THE dbstart COMMAND (Remove any unnecessary lines before leaving editor) Sublime will be shutdown for approximately [XXXX] minutes.</p>
edput	<p>*** YOU ARE IN YOUR FAVORITE EDITOR PLEASE ENTER THE COMMENTS REGARDING THE CHANGES MADE TO THE SOURCE FILE(S). THESE COMMENTS WILL BE ACCUMULATED TO MAKE UP THE MR RESOLUTION.</p>
hcode	<p>*** YOU ARE IN YOUR FAVORITE EDITOR *** PLEASE ENTER THE COMPLETE HARDWAE CODE INFORMATION FOR THIS CODE ID - PRODUCT RELEASE COMBINATION</p>
propose	<p>*** YOU ARE IN YOUR FAVORITE EDITOR *** PLEASE PROPOSE A COMPLETE SOLUTION FOR THIS MR</p>
reject	<p>*** YOU ARE IN YOUR FAVORITE EDITOR *** PLEASE ENTER A COMPLETE DESCRIPTION FOR THE REJECTION OF THIS MR</p>

Table 3-9. Command Templates

Command	Supplied Text
setgroup	<p>*** YOU ARE IN YOUR FAVORITE EDITOR ***</p> <p>PLEASE ENTER ALL GROUP MEMBERS YOU WANT TO BE INCLUDED IN THIS GROUP</p> <p>(Members should be added, one member to a line, left justified. A default can be designated by including a ^ character as the first character of the line. Order is important if this is to be a Popup Selection Group in that the way the Group Members appear here is the way they will appear in the Popup Selection Window)</p>
spawnmr	<p>*** YOU ARE IN YOUR FAVORITE EDITOR ***</p> <p>PLEASE ENTER ANY ADDITIONAL NOTES ABOUT THE PROBLEM FOR THIS MR</p>
submit	<p>1)Resolution:</p> <p>2)Impact Areas (for system testers):</p>
tempset	<p>*** YOU ARE IN YOUR FAVORITE EDITOR ***</p> <p>PLEASE ENTER THE INFORMATION FOR YOUR TEMPLATE FILE</p>
testpass	<p>PLEASE ENTER A TESTER'S NOTES FOR THIS MR</p>

Table 3-10. Template File Names

Command	File Name
commit	com_mrs.tpl
create	cre_desc.tpl
dbstop	dbstop.tpl
edput	edp_com.tpl
hcode	hcode_dsc.tpl
mrnote	mrnt_desc.tpl mrnt_gsol.tpl mrnt_note.tpl mrnt_rej.tpl mrnt_res.tpl mrnt_sol.tpl mrnt_tps.tpl
propose	prop_solu.tpl
reject	rej_rsn.tpl
screate	smr_notes.tpl
setgroup	grp_memb.tpl
spawnmr	smr_notes.tpl

Table 3-10. Template File Names—Continued

Command	File Name
submit	sub_solu.tmpl
tempset	tempset.tmpl
testpass	tps_note.tmpl

Defining New Fields

Sablime uses predefined fields for data collection. The User-Definable Field (UDF) feature permits the collection of additional product-specific information by allowing the Database Administrator (DBA) to define product-specific fields for improved MR tracking. The UDF feature provides 19 UDFs.

The create, fcreate, review, and mredit commands can be used to collect and edit up to five product-level fields in the MR relation. The submit and mrgedit commands can be used to collect and edit up to five generic-level fields in the MG relation. The hcode command can be used to collect and edit up to two fields in the HC relation; the pdi command can be used to collect and edit up to two fields in the PDI relation.

The report and query commands display the information contained in each UDF of the MR, MG, HC, and PDI relations, in addition to the standard predefined fields provided by Sablime.

Table 3-11 shows the UDFs that are available, the relation in which they are stored, and the commands that allow collection, editing, or display of the data in these fields.

Table 3-11. User-Definable Fields

Default External Keyword	Number of UDFs	Relation	Command
mrudf1–mrudf5	5	MR	create fcreate mredit review query (Qmr) report
mrgudf1–mrgudf5	5	MG	mrgedit submit query (Qmg) report
hcudf1–hcudf2	2	HC	hcode query (Qhc) report
pdiudf1–pdiudf2	2	PDI	pdi query (Qpdi) report
emrudf1–emrudf5	5	EMR	review



CAUTION:

Do not customize any characteristics of the EMR UDFs other than the Display flag and the Screen Label. The characteristics of the EMR UDFs belong to the UDFs as they are defined by the external product. Customization of any of the EMR UDFs for your product could lead to a conflict with the external product and will be ignored by the review command.

If you change the Screen Label of an EMR UDF, it may cause confusion if you receive EMR UDF information from more than one external product.

Once the new UDFs have been properly defined, they are available for the **LONG**, **ALL**, and **CUSTOM** MR reports.

The next section describes the UDF feature. The following sections describe how to customize and use UDFs.

The UDF Feature

Customizing FTD Fields

The UDF feature allows projects to customize the following FTD fields for each UDF:

- n Label in the Curses Forms interface and the Graphical User interface
Each product can customize the screen label of the UDF by using the `ftd` command. For example, you can modify the screen label *MR UDF1*: to *Some Meaningful Label*: for the `create` command. All Curses Forms interface and Graphical User interface labels can be modified. The new label is displayed on the screen whenever a modified command is run. It is possible to modify the default Sablime labels that are not UDFs; however, only the UDF label changes are used by the `query` and `report` commands.



NOTE:

If you change a UDF label for one command, you should change it for all the commands that use that UDF. See Table 3-11 for a list that maps UDFs to commands.

- n Screen layout
You can customize the Curses Forms interface screen layout (column, row, and length) for the UDF(s) with the `ftd` command. You can hide the unused UDF(s) and display only the UDF(s) that your product uses, in which case traversal order is strictly maintained.
- n External Keywords
External keywords are used in the Command Line interface to identify a field when you enter data on the command line in the `keyword=value` format. You can use the `ftd` command to customize all the UDF keywords.
Consider the following example. Accepting all the defaults for all the MR attributes, the `create` command can be executed as follows:

```
create abst="This is a test" mrudef1=111\ desc=desc_file
```

We can then rename the `abst` and `mrudef1` keywords to `shortdesc` and `days`. Once this change has been made, we can then run the following command line:

```
create shortdesc="this is a test" days=111\ desc=desc_file
```

- n With or without a pop-up menu on the UDF
You can use the `ftd` command to customize a UDF in the Curses Forms interface and the Graphical User interface to have a pop-up menu for the field, to accept any string as input, or to accept only numeric input matched to entries in a file.

- n Left/Right (L/R) Scroll size

The maximum input length allowed for each UDF is 512 characters, but a limit of 256 characters is strongly recommended because of certain buffer size limits. You can use the `ftd` and `setgroup` commands to customize the input length of each UDF in the Curses Forms interface and the Graphical User interface. If the input length is longer than the input field displayed on the screen, you must set the left/right scroll size attribute to the appropriate value.

Maximum UDF Values

For each UDF, the following values are the maximum values that can be specified:

- n Maximum length of field on screen: (*Screen Label* length + *Length* field) up to 79 characters
- n Keyword length: up to 14 characters
- n Input data length: up to 512 characters.



NOTE:

Because of buffer-size issues in multi-machine mode using TCP/IP, we recommend that you limit UDF input data length to no more than 256 characters.

UDF Input Data Validation

The UDF provides input validation for the following:

- n Validate Input Length
Data can be entered up to the length specified in the *FTD Length* field if the UDF is not a Left/Right Scrollable field. If the UDF is a Left/Right Scrollable field, data can be entered up to the length specified in the *FTD Left/Right Scroll Size* field.
- n Validate Input Data
When the UDF has a pop-up menu for the field, input is accepted only if the input data matches one of the entries contained in the pop-up menu.

Default FTD Data for UDFs

The following is a list of the default FTD data:

- n All external keywords have the same names as their corresponding internal keywords for all fields.

- n All UDFs have the *Display* flag off, i.e., none of the UDFs are shown on the screen in the Curses Forms interface or the Graphical User interface.
- n All UDFs are optional, i.e., users do not have to enter data in these fields.
- n All UDFs have field type 1, i.e., none of the UDFs have a pop-up menu for the field in the Curses Forms interface or the Graphical User interface.
- n No UDFs are Left/Right Scrollable, i.e., the input length is limited to the field length displayed on the screen in the Curses Forms interface and the Graphical User interface.
- n All UDFs have *YYYUDFx* as a screen label in the Curses Forms interface and the Graphical User interface, where *YYY* can be MR, MRG, HC, PDI, or EMR and *x* indicates the number of the UDF. All UDFs have *yyyudfx* as an external keyword, where *yyy* can be mr, mrg, hc, pdi, or emr and *x* indicates the number of the UDF.

Customizing a UDF

This section describes how to modify the UDF for the commands shown in Table 3-11.

Displaying a UDF on the Command Screen

1. Run the `ftd` command to modify a UDF (specify the command and internal keyword).
2. Set the *Display* flag to **y**.

Making a UDF Mandatory

1. Run the `ftd` command to modify a UDF (specify the command and internal keyword).
2. Set the *Mandatory* flag to **y**.

Adding a Pop-Up Menu for a UDF

1. Run the `setgroup` command.
2. Create a group that contains all possible input entries for the UDF. See the `setgroup` command in the *Sablime User's Guide* for more details.
3. Run the `ftd` command.
4. Change the *Field Type* field value from 1 to 2.
5. Enter the maximum number of entries allowed on the UDF by modifying the value shown in the *Pop-Up Selections* field.
6. Enter the group name (from step 2) in the *Group/File* field.

Changing the Data Input Length for a UDF

1. Run the `ftd` command. If the maximum input length is greater than the length of the field displayed on the screen, enter the maximum length in the *L/R Scroll Size* field. If the maximum input length is less than the length of the field, enter the desired number in the *Length* field.

Adding Verbose Help to a UDF

1. Use your text editor to create a VHELP text file in your VHELP directory (under the MCB directory in its default location). The maximum length of each line is 42 characters. Name the file with one of the reserved UDF verbose VHELP numbers (i.e., 6001–6050).
2. Run the `ftd` command.
3. Add the file name to the Verbose Help Key field.

Changing the Screen Label and the NoPrompt Help Text

1. Run the `ftd` command.
2. Change the screen label to the desired label.
3. Change the noprompt help text to the desired text.

Changing the External Keyword

1. Run the `ftd` command to change the external keyword to the desired external keyword. The external keyword must be unique for each field of the command.



NOTE:

For consistency, when you define a UDF for a command, you should also define the same UDF(s) for the commands that use that UDF. See Table 3-11 for a list that maps UDFs to commands.

For example, if your product uses *mrudf1* in the `create` command to allow users to enter additional information to the MR relation, your product should also use *mrudf1* in the `fcreate` and `review` commands because these two commands perform a similar function (they both can create a new MR). In addition, the `mredit` command should also use the *mrudf1* field to allow modification of the field when necessary. Finally, the `report` and `query` commands should use *mrudf1*, so the information stored in the *mrudf1* field is reflected whenever the `report` or `query` command is issued.

Therefore, if your project uses:

- n a UDF in the MR relation

You should update the FTD data of the same UDF in the create, mredit, fcreate, review , report, and query commands.

- n a UDF in the MG relation

You should update the FTD data of the UDF in the submit, mrgedit, report, and query commands.

- n a UDF in the HC relation

You should update the FTD data of the UDF in the hcode and query commands.

- n a UDF in the PDI relation

You should update the FTD data of the UDF in the pdi and query commands.

Customizing UDFs: An Example

In this example we will define the *mrudf1* and *mrudf2* UDFs for the create command. *mrudf1* will be defined to accept up to 10 characters as input and *mrudf2* will be defined as a L/R Scrollable field with a pop-up menu. In addition, we will define *mrudf1* and *mrudf2* for the query command. We will do this in five steps by:

1. Customizing the *mrudf1* UDF for the create command
2. Creating a pop-up menu for the *mrudf2* UDF
3. Customizing the *mrudf2* UDF for the create command
4. Customizing the *mrudf1* UDF for the query command
5. Customizing the *mrudf2* UDF for the query command

This example assumes that the external keywords have not been changed by the Database Administrator. The following message is displayed if the external keywords have been changed and an attempt is made to use the command lines used in this example:

Err[999]: Invalid parameter [xxx] to [ftd] command.

Following the display of this message, the valid keywords will be listed.

Figure 3-16 shows the default screen that appears when the create command is run and no UDFs have been defined:

```
logid:xxxxxx      Sablime Configuration Management System v5.0      03/09/96
effid:sablime     MR Management System Command      11:12:06

                          Create a New Modification Request

Originator PTS ID: _____      Origination Date: _____
Request Severity:  _                Required Date:  _____

      Product:  _____      Site:  _____
      System:  _____
      Subsystem: _____
      Module:  _____
      Rel. Detected: _____
      Phase Detected: _____
      Category: _____

Abstract of Request: _____
      Request Desc File: _____
      Copy To:  _____
```

Figure 3-15. Default create Screen

Customizing the *mrudf1* UDF for the create Command

Issue the following command line to display the default *mrudf1* FTD data for the create command:

ftd fcn=modify cmmnd=create intkey=mrudf1

The screen shown in Figure 3-16 appears:

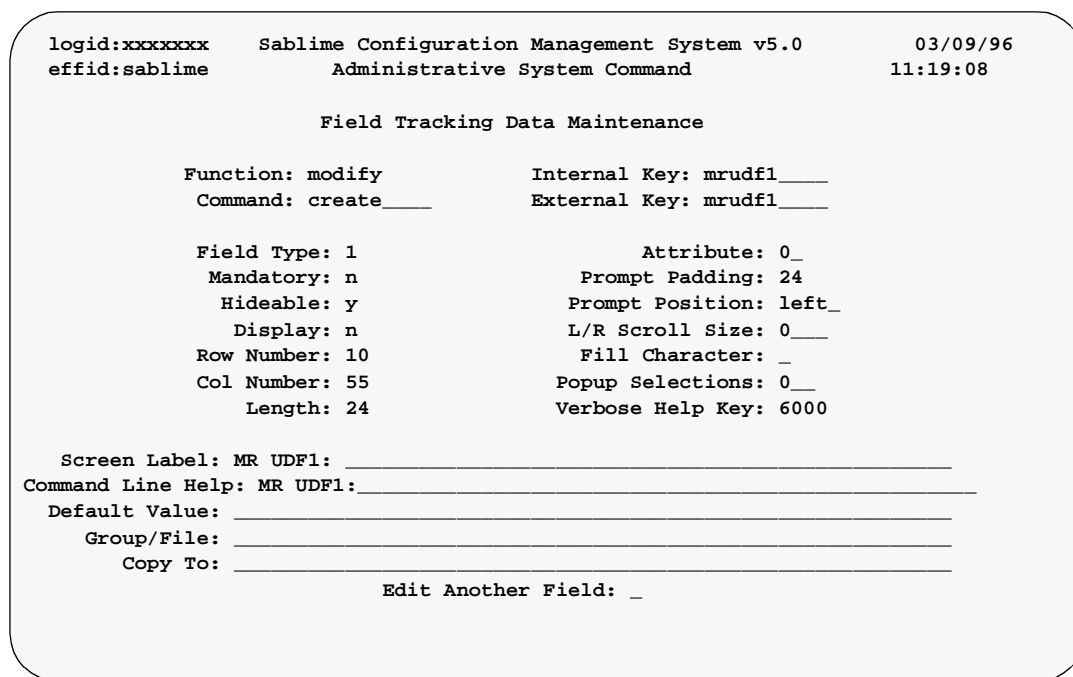


Figure 3-16. Default *mrudf1* FTD Data for the create Command

Customize the *mrudf1* UDF for the create command by doing the following:

1. Change the external keyword (*mrudf1*) by entering the desired name in the *External Keyword* field, e.g., **clock**.
2. Display the field on the screen by setting the *Display* field to **y**
3. Change the value displayed in the *Length* field to the expected input length, e.g., **10**.
4. Change the *Screen Label* field value and the *NoPrompt Help* field value. For example, change the *Screen Label* field value from **MR UDF1:** to **Clock:** and change the *NoPrompt Help* field value from **MR UDF1:** to **Clock:.**
5. Change the *Verbose Help Key* number to a new number, e.g., **6001**. Create a file of verbose help messages with the same name (6001) and put it into the VHELP directory listed in the \$sabVHELP file.

Once the *mrudf1* UDF field has been customized, the *mrudf1* UDF FTD data for the create command should appear as shown in Figure 3-17:

```

logid:xxxxxxx   Sablime Configuration Management System v5.0   03/09/96
effid:sablime   Administrative System Command                   11:19:08

                          Field Tracking Data Maintenance

Function: modify           Internal Key: mrudf1____
Command: create____       External Key: clock____

Field Type: 1             Attribute: 0_
Mandatory: n             Prompt Padding: 24
Hideable: y              Prompt Position: left_
Display: y               L/R Scroll Size: 0__
Row Number: 10           Fill Character: _
Col Number: 55           Popup Selections: 0__
Length: 10               Verbose Help Key: 6001

Screen Label: Clock: _____
Command Line Help: Clock: _____
Default Value: _____
Group/File: _____
Copy To: _____

                          Edit Another Field: _

```

Figure 3-17. Customized mrudf1 FTD Data for the create Command

6. Confirm the entries made by selecting **y** at the CONFIRM field.

As processing takes place, information like that shown below appears on the screen:

```

+ Field [mrudf1] in the FTD Record [create] has been modified in the active database.

```

Creating a Pop-Up Menu for the mrudf2 UDF

Create a pop-up menu for *mrudf2* for the *create* command by doing the following:

1. Create a file that contains four group members (e.g., *high_temp*, *low_temp*, *high_humidity*, and *dusty*) and save the file under the file name *popup_entries*.
2. Use the following *setgroup* command line to create a group called *XENVGRP* that contains the four members described in step 1:

```
setgroup grp=XENVGRP type=other mfile=popup_entries\ prompt=n
```

Customizing the *mrudf2* UDF for the create Command

Issue the following *ftd* command line to display the default *mrudf2* FTD data for the create command:

```
ftd fcn=modify cmmnd=create intkey=mrudf2
```

The screen shown in Figure 3-18 appears:

```

logid:sablime      SablIME Configuration Management System v5.0      03/14/96
effid:sablime     Administrative System Command                      09:30:14

                          Field Tracking Data Maintenance

Function: modify      Internal Key: mrudf2___
Command: create_____ External Key: mrudf2___

Field Type: 1          Attribute: 0_
Mandatory: n          Prompt Padding: 24
Hideable: y           Prompt Position: left_
Display: n            L/R Scroll Size: 0___
Row Number: 11        Fill Character: _
Col Number: 55        Popup Selections: 0___
Length: 24            Verbose Help Key: 6000

Screen Label: MR UDF2: _____
Command Line Help: MR UDF2: _____
Default Value: _____
Group/File: _____
Copy To: _____

                          Edit Another Field: _
    
```

Figure 3-18. Default *mrudf2* FTD Data for the create Command

Customize *mrudf2* for the create command by doing the following:

1. Change the external keyword (*mrudf2*) by entering the desired name in the *External Keyword* field, e.g., **extenv**.
2. Change the *Field Type* value from 1 to 2 to indicate that this field has a pop-up menu.
3. Display the field on the screen by setting the *Display* field to **y**.
4. Change the *L/R Scroll Size* to **50**.
5. Allow the user to select up to two items from the pop-up menu by changing the *Popup Selections* field value to **2**.

6. Change the *Screen Label* field entry from MR UDF2 to **External Env:**. Change the *NoPrompt Help* field entry from MR UDF2 to **External Env:**.
7. Enter the pop-up menu group name in the *Reserved Group* field, e.g., **XENVGRP**.
8. Change the *Verbose Help Key* number to a new number, e.g., **6002**. Create a file of verbose help messages with the same name (6002) and put it into the VHELP directory listed in the \$sabVHELP file.

Once the *mrudf2* UDF field has been customized, the ftd command screen for the create command should appear as in Figure 3-19:

```

logid:sablime      Sablime Configuration Management System v5.0      03/14/96
effid:sablime     Administrative System Command                        09:30:14

                          Field Tracking Data Maintenance

Function: modify      Internal Key: mrudf2___
Command: create_____ External Key: extenv___

Field Type: 2          Attribute: 0_
Mandatory: n          Prompt Padding: 24
Hideable: y           Prompt Position: left_
Display: y            L/R Scroll Size: 50__
Row Number: 11        Fill Character: _
Col Number: 55        Popup Selections: 2__
Length: 24            Verbose Help Key: 6002

Screen Label: External Env: _____
Command Line Help: External Env: _____
Default Value: _____
Group/File: XENVGRP_____
Copy To: _____

                          Edit Another Field: _
    
```

Figure 3-19. Customized mrudf2 FTD Data for the create Command

Confirm the entries made by selecting **y** at the CONFIRM field.

As processing takes place, information like that shown below appears on the screen:

```

+ Field [mrudf2] in the FTD Record [create] has been modified in the active database.
    
```

With the new FTD data, the modified screen that is displayed when the create command is run should appear as shown in Figure 3-20:

```
logid:sablme      Sablme Configuration Management System v5.0      03/31/96
effid:sablme      MR Management System Command                          12:33:18

                          Create a New Modification Request

Originator PTS ID: _____      Origination Date: _____
Request Severity:  _                Required Date:  _____

      Product:  _____      Site:  _____
      System:  _____      Clock:  _____
Subsystem:  _____      External Env:  _____
      Board:  _____

Rel. Detected:  _____
Phase Detected:  _____
Category:  _____

Abstract of Request:  _____
Request Desc File:  _____
Copy To:  _____
```

Figure 3-20. Customized create Command Screen

Customizing the mrudef1 UDF for the query Command

The screen shown in Figure 3-21 is the default screen that appears when the query command is run on the MR relation (i.e., no UDFs have been defined):

```
query relation=MR db=active
```

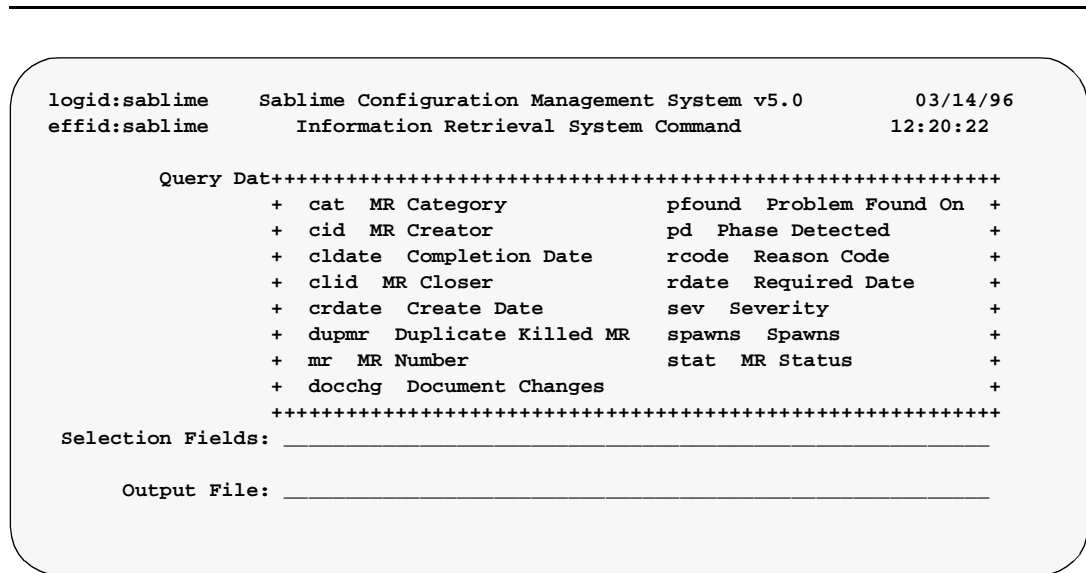



Figure 3-21. Default screen for query Command

Issue the following command line to display the default *mrudef1* UDF FTD data of the query command for the MR relation:

ftd fcn=modify cmmnd=Qmr intkey=mrudef1



NOTE:

Only fields without subfields are available for selection. (Fields are separated by semicolons in the databases; subfields are separated by commas.) For efficiency reasons, if you want to query FTD records of the query command itself, the query records are designated in the database by the fictitious command name of *Qrelation_name* (where *relation_name* represents the relation name in lower-case letters), i.e., **Qmr**. **Qmr** was specified because **Qmr** contains the FTD data for the appropriate screen of the query command. See the *User's Guide* for details about query.

The screen in Figure 3-22 appears:

```

logid:sablme      Sablime Configuration Management System v5.0      03/14/96
effid:sablme     Administrative System Command                      09:30:14

                          Field Tracking Data Maintenance

Function: modify      Internal Key: mrudf1____
Command: Qmr_____  External Key: mrudf1____

Field Type: 1        Attribute: 0_
Mandatory: n        Prompt Padding: 12
Hideable: y         Prompt Position: left_
Display: n          L/R Scroll Size: 0__
Row Number: 8_      Fill Character: _
Col Number: 22      Popup Selections: 0__
Length: 57          Verbose Help Key: 6000

Screen Label: MR UDF1: _____
Command Line Help: MR UDF1: _____
Default Value: _____
Group/File: _____
Copy To: _____

                          Edit Another Field: _
    
```

Figure 3-22. Default mrudf1 FTD Data for the query Command

Customize the *mrudf1* FTD data of the query command by doing the following:

1. Change the external keyword (*mrudf1*) by entering the desired name in the *External Name* field, e.g., **clock**.
2. Display the field on the screen by setting the *Display* field to **y**.
3. Change the *Screen Label* field from MR UDF1 to **Clock:**. Change the *NoPrompt Help* field from MR UDF1 to **Clock:**.
4. Change the *Verbose Help Key* number to a new number, e.g., **6003**. Create a file of verbose help messages with the same name (6003) and put it into the VHELP directory listed in the \$sabVAR file.

Once *mrudf1* has been customized, the *mrudf1* FTD data for the query command should appear as shown in Figure 3-23.

```

logid:sablime      Sablime Configuration Management System v5.0      03/14/96
effid:sablime     Administrative System Command                        11:50:10

                          Field Tracking Data Maintenance

Function: modify      Internal Key: mrudf1___
Command: Qmr_____   External Key: clock_____

Field Type: 1          Attribute: 0_
Mandatory: n          Prompt Padding: 12
Hideable: y           Prompt Position: left_
Display: y            L/R Scroll Size: 0__
Row Number: 8_       Fill Character: _
Col Number: 22       Popup Selections: 0__
Length: 57           Verbose Help Key: 6003

Screen Label: Clock: _____
Command Line Help: Clock: _____
Default Value: _____
Group/File: _____
Copy To: _____

                          Edit Another Field: _
    
```

Figure 3-23. Customized mrudf1 FTD data for the query Command

Confirm the entries made by selecting **y** at the CONFIRM field.

As processing takes place, information like that shown below appears on the screen:

```

+ Field [mrudf1] in the FTD Record [Qmr] has been modified in the active database.
    
```

Customizing the mrudf2 UDF for the query Command

Issue the following command line to display the default *mrudf2* FTD data of the query command for the MR relation:

```

ftd fcn=modify cmmnd=Qmr intkey=mrudf2
    
```

The screen in Figure 3-24 appears:

```

logid:sablme      Sablime Configuration Management System v5.0      03/14/96
effid:sablme     Administrative System Command                        11:50:10

                          Field Tracking Data Maintenance

Function: modify          Internal Key: mrudf2__
Command: Qmr_____      External Key: mrudf2__

Field Type: 1            Attribute: 0_
Mandatory: n             Prompt Padding: 12
Hideable: y              Prompt Position: left_
Display: n                L/R Scroll Size: 0__
Row Number: 9_           Fill Character: _
Col Number: 22           Popup Selections: 0__
Length: 57               Verbose Help Key: 6000

Screen Label: MR UDF2: _____
Command Line Help: MR UDF2: _____
Default Value: _____
Group/File: _____
Copy To: _____

                          Edit Another Field: _
    
```

Figure 3-24. Default *mru~~df~~2* FTD data for the query Command

Customize the *mru~~df~~2* FTD data for the query command by doing the following:

1. Change the external keyword by entering the desired name in the External Keyword field (e.g., **extenv**).
2. Display the field on the screen by setting the *Display* field to **y**.
3. Change the *Screen Label* field from UDF2 to **External Env:**. Change the *NoPrompt Help* field from UDF2 to **External Env:**.
4. Change the *Verbose Help Key* number to a new number, e.g., **6004**. Create a file of verbose help messages with the same name (6004) and put it into the VHELP directory listed in the \$sabVAR file.

Once *mru~~df~~2* has been customized, the *mru~~df~~2* FTD data for the query command should appear as shown in Figure 3-25:

```

logid:sablme      Sablime Configuration Management System v5.0      03/14/96
effid:sablme     Administrative System Command                       11:55:55

                          Field Tracking Data Maintenance

Function: modify      Internal Key: mrudf2__
Command: Qmr_____  External Key: extenv__

Field Type: 1          Attribute: 0_
Mandatory: n          Prompt Padding: 12
Hideable: y           Prompt Position: left_
Display: y            L/R Scroll Size: 0__
Row Number: 9_        Fill Character: _
Col Number: 22        Popup Selections: 0__
Length: 57            Verbose Help Key: 6004

Screen Label: External Env: _____
Command Line Help: External Env: _____
Default Value: _____
Group/File: _____
Copy To: _____

                          Edit Another Field: _
    
```

Figure 3-25. Customized mru~~df~~2 FTD data for the query Command

Confirm the entries made by selecting **y** at the CONFIRM field.

As processing takes place, information like that shown below appears on the screen:

```

+ Field [mrudf2] in the FTD Record [Qmr] has been modified in the active database
    
```

Once *mru~~df~~1* and *mru~~df~~2* have been defined for the query command, when the query command is run (to query the MR relation), the screen in Figure 3-26 is displayed:

query relation=MR db=active

```
logid:sablime      Sablime Configuration Management System v5.0      03/14/96
effid:sablime      Information Retrieval System Command      12:06:39

Query Datab+++++
+ cat MR Category          extenv External Env +
+ cid MR Creator           pd Phase Detected   +
+ cldate Completion Date   rcode Reason Code   +
+ clid MR Closer           rdate Required Date +
+ crdate Create Date       sev Severity         +
+ dupmr Duplicate Killed MR spawns Spawns          +
+ mr MR Number             stat MR Status       +
+ clock Clock              +
+++++

Selection Fields: _____

Output File: _____
```

Figure 3-26. Customized query Command

The pop-up menu includes the *clock* and *extenv* UDFs in the pop-up selection window.

This concludes the description of User-Defined Fields. For completeness, the following commands should also be customized to include the two UDFs: *fcreate*, *mredit*, *review*, and *report*.

Contents

4	Other Administrative Procedures	1
n	Managing a Product	1
	Adding a Product to a Sablime Instance	1
	Moving a Product on a Machine	1
	Removing a Product from a Sablime Instance	2
n	Managing a Generic	4
	Adding a Generic to a Product	4
	Renaming a Generic	9
	Closing a Generic	11
n	Managing Files and Directories	12
	Adding Directories to a Generic	12
	Adding Files to a Generic	14
	Moving, Renaming, and Deleting Files	22
	Restoring a Deleted File to a Generic	23
n	Managing the Databases	26
	Changing Data in the Databases	26
	Changing Data in the MR Relation	26
	Changing Data in the MRG Relation	27
	Changing Data in Other Relations	27
	Auditing the Databases	27
	Checking for Disk Space	32
	Checking the System Error Message File	33
n	Changing Machines	34
n	Customizing Sablime Mail	35
	Mail Consolidation	35
	How Mail Works	38
	Customizing Mail Recipients	38
n	Customizing Command Executors and Email Recipients	38

Contents

Other Administrative Procedures

4

Managing a Product

The following sections deal with adding a product to Sablime and with moving and removing products already under Sablime.

Adding a Product to a Sablime Instance

Your first product is established when you install Sablime. To add a product to an existing Sablime instance, execute the `initsab` script as described in *Running the Installation Script* in Chapter 2, *Installing Sablime*. You may choose to use the same GDB or a different GDB; using the same GDB is preferable because it makes maintenance easier. Depending on how much space is available, you may want to use different ADB/IDB/SDBs. Be sure to make the necessary changes to the *dot sablime* programs as well. (See *Customizing the dot sablime Script* in Chapter 2.)

When you add a product, you also add a first generic for that product. When the installation is complete, you are ready to add files to the SDB or write MRs against the generic of the product you have just established.

Moving a Product on a Machine

To move the product databases to another location on the same machine, follow these steps:

1. Enter `dbstop` to stop all processing on the database.
2. Move the databases to the new location(s).

3. Using the PR subcommand of the `setrel` command, change the paths to point to the new locations of the Active, Inactive, and Source Databases.
4. Reset the `sabGDB` variable in the `xsablime` shell script.
5. Enter `dbstart` to resume database processing.

Removing a Product from a Sablime Instance

When you have completed work on a product, you may want to remove it from the Sablime databases.



CAUTION:

We recommend a complete backup of the Sablime databases before you begin this procedure.

To remove a product, take the following steps:

1. Execute the `dot sablime` command for a generic that is in the product you want to remove. Make a list of all the generic names in the product by entering the following command:

```
query relation=G prompt=n
```

The generic names are in the first semicolon-separated field. For example, for a product `ancl`, we might have generics `sab1.0`, `sab1.1`, and `sab2.0`.

2. Go to the directory that contains Active Database. For example:

```
cd /home/users/Sablime/adb
```

3. Remove all the files and the directory for your product. For example.:

```
rm -rf sab/*
```

4. Repeat steps 2 and 3 for the Inactive Database and the Source Database.

5. Go to the directory that contains the generic files in the Global database. For example:

```
cd $sabGDB/DIR
```

6. Remove the files for all the generics in the product. For example:

```
rm sab1.0 sab1.1 sab2.0
```

7. Go to the PR relation. For example:

```
cd $sabGDB/PR
```

8. Remove all the records that have your product's name as the first semicolon-separated field. If, when you are finished, there are no more records in the tuple file, you can remove the tuple file as well.

9. Go to the PRX relation. For example:

cd \$sabGDB/PRX

10. Remove all the records for your product that have your product's name as the first semicolon-separated field. If no records are left, you can remove the tuple file as well.

11. Go to the TR relation. For example:

cd \$sabGDB/TR

12. Look for the product name in every tuple file. For example:

grep ';sab;' *

13. Edit the tuple files and remove the records containing the product name. If the tuple file is empty after you remove the product records, you can remove the tuple file as well.

14. If your product used the External MR Communications feature:

- n Go to the Sablime Master Control Bin, e.g.:

cd \$sabMCB



NOTE:

Assuming that the Sablime environment is properly established, the value for sabMCB will either be present in the shell environment (i.e. \$sabMCB), or can be extracted from the Sablime variables file (i.e. "grep sabMCB \$sabVAR"). This note applies to sabLCB also.

- n Remove any records in the .EMR file that have your product's name in the second semicolon-separated field.
 - n Remove any records in the .BIN file that have the name of the product with which your product communicated in the second semicolon-separated field.
 - n Use the ES subcommand of the setrel command to delete the product with which your product communicated.
 - n On satellite machines, go to the Local Control Bin, e.g.,

cd \$sabLCB

Repeat the actions performed on the host Master Control Bin to remove records in the .EMR and .BIN files; use the ES subcommand of the setrel command to delete the external product with which your product communicated.

15. Update the sablime.sh and xsablime.sh scripts to remove any reference to any generic of your product. If there are *dot sablime* commands on any satellites, you must update these files on the satellites as well.
16. Run the sh2x script to regenerate the sablime and xsablime scripts on the host and any satellites.

Managing a Generic

The following sections deal with adding, renaming, and closing generics belonging to a product under Sablime.

Adding a Generic to a Product



NOTE:

If you are using Sablime for file control as well as MR control and you want to propagate files from an old generic to the new one, see the section *Adding Files to a Generic*, in this chapter.

The first generic in a product is established when you execute `initsab` to establish a product. To add a new generic to the product, you can use the `newgen` script or perform a multi-step process using the `addgen` command. It is recommended that you use `newgen`, because `newgen` automates the procedure and forces the user to take all the necessary steps. (`addgen` is normally used to modify an existing generic.) Both procedures are described in the following sections.



NOTE:

If you are using the multi-machine feature, information must be changed for both the host and the satellite machines.

Adding a Generic Using `newgen`

The `newgen` script comprises the five steps required to add a new generic to an existing product in the Sablime product databases. Each step is presented as a separate screen. The script starts by stepping you through the `addgen` and `setgroup` commands to define the new generic and the administrative and test teams; then it creates the new directory structure file and updates the Source Database; finally it puts you into your favorite editor to update the `xsablime.sh` script.



NOTE:

Only the *sablime* login can execute this script; it must be executed on the host machine or on an NFS satellite machine.

Begin by entering::

```
newgen
```

The screen shown in Figure 4-1 appears.

```
*****
This script will add a new generic to an existing product in
your SABLIME databases. You will be prompted for the following
questions:

- New Generic Name,
- Is the Generic Released,
- The State(s) of MR's to accept into the new generic,
- Location of the Directory Structure File,
- What MR Classes to create (e.g. software, document),
- Logids of the Generic Administrator & Test Teams.

Once you have answered these questions, you will be put into
your favorite editor to modify the 'xsablime.sh' shell script

*****

<<< Hit return when ready to continue >>>
```

Figure 4-1. newgen Screen

The prompts that appear after you press RETURN and appropriate responses to them are described below.

- PROMPT: Enter the New Generic Name [Up to 14 characters]:
- RESPONSE: Enter the name of the generic you want to create. Remember that generic names must be unique in your Sablime instance. The name you give here is used in the remaining prompts; for this example, we use a2.0 as the new generic name and a1.0 as the previous generic name.
- PROMPT: Is Generic [a2.0] Released [y or n]:
- RESPONSE: If the generic has already been released to customers, enter **y**; otherwise, enter **n**.
- PROMPT: Enter the state(s) of MR's you want to accept into the new generic [a2.0] from the previous generic [a1.0]. The input should be comma separated, with no spaces.

accepted	inspected	prehstpassed	submitted
approved	itpassed	preinspected	understudy
deferred	nochange	preitpassed	all
fitpassed	preapproved	prepublished	none
fstpassed	prefitpassed	prestpassed	
hitpassed	prefstpassed	published	
hstpassed	prehitpassed	stpassed	

Enter State(s):

RESPONSE: Enter the states of MRs from the previous generic that you want to accept into the new generic. These MRs enter the new generic in the *accepted* state, regardless of their states in the previous generic. You can also choose to accept all MRs or no MRs.

PROMPT: Will the Directory Structure for generic [a2.0] be the same as used for the previous generic [a1.0] [y or n]:

RESPONSE: If the directory structure of the new generic is the same as the structure of the previous generic, enter **y**. If you want to use a new directory structure, enter **n**; you are then prompted for the file that contains the new structure.

PROMPT: Enter the full path and file name of the file that contains the New Directory Structure for generic [a2.0]:

RESPONSE: Enter the full path and file name.

At this point, you receive the following notification:

The Source Database has been updated with the new directories

PROMPT: < Give ANSWERS in the form of a COMMA SEPARATED list, no spaces >
If you would like the same Administration Team Members from the previous generic [a1.0] to be used for the new generic [a2.0], enter the word 'same', otherwise enter the logids of the team members to be included in the format of [sablime]. Enter the Generic Administration Team Members:

RESPONSE: If you want to have the same administration team members as the previous generic, enter **same**. If you do not, enter the members of the new team. Use the format shown (either machine!login or simple login).

PROMPT: Which of the following MR Classes would you like [y or n]:
NOTE:At least one MR Class must be selected!
Are you going to have Document Class MRs [y or n]:
Are you going to have Firmware Class MRs [y or n]:
Are you going to have Hardware Class MRs [y or n]:
Are you going to have Software Class MRs [y or n]:

RESPONSE: Enter **y** or **n** for each MR class you want for the new generic. You must answer **y** for at least one MR class.

You are next prompted for the team members who will participate in the test and approval process for the MR classes you have chosen. This example assumes document class MRs have been chosen.

PROMPT: Set Up Document Class Teams and States

< Give ANSWERS in the form of a COMMA SEPARATED list, no spaces >

If you would like the same Test Team Members from the previous generic [a1.0] to be used for the new generic [a2.0], enter the word 'same'. If you do not want this Test Team/State, leave it blank, otherwise enter the logids of the team members to be included in the format of [sablime].

Pre-Inspection Team:

Inspection Team:

Pre-Publishing Test Team:

Publishing Test Team:

Pre-Approval Team:

Approval Team:

QA Team:

RESPONSE: If you want to have the same generic administration team members as the previous generic, enter **same**. If you do not, enter the members of the new team. Use the format shown (either machine!login or simple login).

<<< Press RETURN when ready to continue >>>

When you press the RETURN key, a message similar to the following appears:

The Databases are now being prepared for generic a2.0 Please Stand By...

The Database Setup for generic a2.0 is now complete

PROMPT: The final step is to modify the '. sublime' script to include the new generic updates. You will now be put into your favorite editor to modify two sections in the script 'xsablime.sh'. The next set of instructions will also be at the top of the temporary file when you are inside your editor. Then a message like the following one appears. This message is also prepended to your xsablime.sh file.

```
#####
#Follow these instructions on updating the 'xsablime.sh' script:
#
#Enter a new print statement listing the new generic:
#####
### USER MODIFICATION ###
#####
#print -u2 " 'a2.0' (Ancl Environment; Generic a2.0)"
#
#Add the new generic name to the case statement:
#####
### USER MODIFICATION ###
#####
#a1.0|a2.0) NOTE: a2.0 HAS BEEN ADDED TO THE 'case'
  export sabPROD=ancl
#####
```

You must place the print statement and the shell case properly in the xsablime.sh file and remove the initial pound sign. The following message appears:

You will now be placed in your favorite editor

When you have modified your xsablime.sh file, the final messages appear:

A copy of these instructions on modifying the '. sublime' script will be left in the present working directory in the file 'README.newgen'

Generic [a2.0] is now Installed

The new generic has now been installed. It may have MRs accepted from the previous generic; if so, they are in the *accepted* state. You may now add files to the new generic.

If you choose not to use newgen, use the following manual procedure.

Adding a Generic Using addgen

1. Use the `setgroup` command to create the groups of Generic Administrators and Test Team members that will be used for the new generic. (See the `addgen` manual page in Chapter 6, *The Administrative Commands*, for information about the groups you will need. Instructions for using `setgroup` are in the *User's Reference Manual*.)
2. Execute the `addgen` command to add the generic using the administrative groups created in step 1.
3. Update the `xablime.sh` program to include information about the new generic.
4. Add a file to the `DIR` directory in the Global Database containing the directory structure of the new generic. (See the instructions for creating this directory structure in *Setting Up a Directory Structure File* in Chapter 2, *Installing Sablime*.)
5. Use the `setnode` command to update the directory structure of the Source Database for the product with any new directories required by the generic.
 1. Set up in the generic for which the directory is being added. For example:

```
. sablime gen1
```
 2. Change directory to the Source Database. For example:

```
cd /usr1/sablime/sdb
```
 3. Run `setnode` with the product name as the argument.

```
setnode $sabPROD
```



NOTE:

When you add a directory structure for a new generic to the Source Database, the node name used in the `setnode` command must be the same as the product name. This restriction only applies to creating or updating the directory structure of the Source Database (SDB). In other circumstances, the argument to `setnode` can be any directory name. Instructions for using `setnode` are in the *User's Reference Manual*.

Renaming a Generic

To rename an existing generic in the Sablime databases, follow the steps below.

1. Log in as `sablime`.
2. Execute the `dot sablime` command for the generic that you want to rename.
3. Enter `dbstop` to prevent any other user from modifying the Sablime databases while the conversion process is running.



CAUTION:

Do not invoke the mvgen script more than once for the same product at the same time. A second invocation may corrupt the Sablime databases. After mvgen has completed successfully , you can invoke it again.



NOTE:

You can abort the mvgen script at any time by pressing the Interrupt key. However, it is not recommended that you do this.

4. Execute the mvgen script.

The screen shown in Figure 4-2 will be displayed.

```
+++++
                                     WELCOME TO THE Sablime (R)
                                     Configuration Management System

This script will rename a product generic name from one to another.
This script should be executed as per the steps mentioned in the "Sablime
v4.x Generic Renaming Utility". Make sure to run the audits and
"hotline.ck" programs before rename.

Please note some of the key points:
- The rename will be done in the same place where you currently have
  Sablime v4.x Active and Inactive Databases. Make sure that you have
  enough disk space in the database file systems.
- Run "dbstop" command for all the product for which you plan to rename
  one of its generics of your Sablime instance.
- You must login with "Sablime" logid that owns the Sablime databases.

You can abort the rename at any time by hitting the break key. When
you start the rename again, it will only complete the unfinished rename.

+++++ Press <return> to continue +++++
```

Figure 4-2. mvgen Screen

The prompts that appear after you press RETURN and appropriate responses to them are described below; *version* indicates the current version of the Sablime software..

PROMPT: Enter the present version of Sablime [Default : *version*] :

RESPONSE: Enter the version of Sablime that you are using; if the default shown is correct, press RETURN.

PROMPT: Enter Full Pathname to the Sablime *version* Global Database
[Default : *full_path_of_sabGDB*] :

RESPONSE: Enter the full path of the GDB for the current product; if the default shown is correct, press RETURN.

PROMPT: ***** Ready to Rename a Generic of “*generic*” *****

Enter 'y' to continue, 'q' to quit :

RESPONSE: Enter **y** if you want to rename the generic; enter **q** to stop the renaming process.

PROMPT: You have the following generics in the product *product* instance:
[list of generics]

Enter the generic that you would like to rename
(i.e., *gen1*) :

RESPONSE: Enter the name of the generic that you want to rename.

PROMPT: Enter the new generic name (i.e., *newgen1*) :

RESPONSE: Enter the new name of the generic.

mvgen then proceeds to rename the generic, issuing processing messages as it proceeds.

5. Edit the *xsablime.sh* script to change the case statement from the old generic name to the new one. Run *sh2x* to update the *dot sablime* command.
6. Run the *dot sablime* command for a valid generic in the product to which the renamed generic belongs.
7. Use *dbstart* to start the databases again.



NOTE:

If an error is encountered or the interrupt key is pressed while *mvgen* is processing, trace information is left in the *\$sabGDB/tmp* directory and *mvgen* exits. *mvgen* can be restarted by following steps 2 through 4, above. When it is restarted, *mvgen* remembers the old and new generic names that were entered before the restart, and continues the conversion process from the point where it left off.



NOTE:

To undo the changes made by an execution of `mvgen`, execute `mvgen` again to change the new generic name back to the old one.

Closing a Generic

When all the work on a generic has been completed, no further work is contemplated, and all the MRs associated with the generic are in the *closed* state, the generic should be closed. When a generic is closed, all records associated with it are moved from the Active to the Inactive Database.

To close a generic, perform the following steps:

1. Log in as *sablime*. If you are in multi-machine mode, you must be on the host.



NOTE:

Because closing a generic may take considerable time if the generic contains many files and deltas, it is recommended that you run the process as a background process, using the Command Line interface.



CAUTION:

Once a generic is closed, the only operations that can be performed on it are query, report, ssql, and retrieving a snapshot. A closed generic cannot be reopened.

2. Enter:

```
closegen g=generic prompt=n
```



CAUTION:

Be careful; the default generic is the setup generic.

When processing completes, the generic has been closed.

Managing Files and Directories

The following sections deal with adding files and directories to a generic and with moving, renaming, deleting, and restoring files in a generic.

Adding Directories to a Generic

The directory structure for a generic is established when you add a generic for a product. To add a directory to the directory structure for a generic, follow the instructions below.

1. Log in as *sablime* and go to the directory that contains the directory structure file.

In single-machine mode or in a multi-machine environment, go to the location of the DIR directory in the GDB for your product on the host machine. For example:

```
cd /usr1/sablime/gdb/DIR
```

If in a multi-machine environment, go to the Local Control Bin where the Sablime commands reside on the satellite machine. For example:

```
cd /usr1/sablime/sabLCB
```

2. Modify the file named *generic*, where *generic* is the name of the directory structure file for the generic to which a directory is being added. Insert the new directory (or directories) in the correct position in the file.

For example, you might edit a file named *sab5.0* and add a new directory named *feature* at the *doc/* level as shown below:

```
bin
doc
doc/admin
doc/
feature
doc/usr
src
src/admin
.
.
.
```



CAUTION:

Be careful that there are no spaces at the end of any of the lines of the file and that none of the lines begins with a . / .

See the instructions in *Setting Up a Directory Structure File* in Chapter 2, *Installing Sablime*, for information about the order of the directories and subdirectories in the file.

3. Use the `setnode` command to update the SDB directory structure with the directories you added in step 2. (Instructions for using `setnode` are in the *User's Reference Manual*.) The examples below illustrate this step.



NOTE:

The name of the node in the `setnode` command must be the name of the product that contains the generic. This restriction only applies when updating the Source Database directory structure. In other circumstances, the argument to `setnode` can be any directory name.

1. Change directory to the directory for the product in the SDB directory structure on the host machine. For example:

```
cd /usr1/sablime/sdb
```

2. Set up in the generic for which the directory is being added. For example:

```
. sablime gen1
```

3. Run the `setnode` command to add the new directory to the directory structure for the generic. For example:

```
setnode $sabPROD
```



NOTE:

This procedure only updates the master directory structure file and the Source Database directory structure. It does not update any other instances of this structure. Users must still run `setnode` individually on their working nodes if they wish to pick up the new directories. (Users can re-run the `sablime` setup script to update their home node).

Adding Files to a Generic

Adding a single file to a generic can be accomplished by using the `addgsr` or `addisr` commands, depending upon whether the file to be added is already in another generic (in which case `addgsr` is used) or not (in which case `addisr` is used). However, if you want to add a large number of files, it is simpler to use the `primsdb` command, which is a shell interface to a batch process that runs `addgsr` or `addisr` for you. In multi-machine projects, the `primsdb` command must be run on the host where the databases are located.



NOTE:

`primsdb` allows you to add files from one generic to another by calling `addgsr` where appropriate; see the `addgsr` manual page (in the *User's Reference Manual*) for details before you run `primsdb`, especially if you need

a group of MRs to specify the version of the files you want to add to the new generic.

To run `primsdb`, you may want or need to categorize the files to be added by language and/or owner, and, if they are being added from a previous generic, by branch (MR or official) and by the generics in which they are common as well. Then, for each category, you must create a separate file containing all the files in that category. And finally, you must run `primsdb` as many times as you have categories, each time using a different category file as input.



NOTE:

You cannot use `primsdb` to add files to Sablime at the top of your product (generic) directory structure (i.e., the directory cannot be “.”). For those files, you must use the `addsrc` or `addgsrc` command.

If you are adding files to your product for the first time, use the `setnode` command (see `setnode` in the *User's Reference Manual*) to create a copy of the directory structure for the generic. These directories should then be populated with the files to be added. If you are adding files from an existing generic, the name of that generic is needed. In either case, you must create a file containing a list of the files with relative directory names.

Preparing to Run `primsdb`.

Before you execute `primsdb`:

1. Make sure that the generic to which you want to add files already exists.
2. Run the `dot sablime` program to set up for the generic to which the files will be added. For example:

```
. sablime new1.0
```

3. Create a file containing a list of the files to be added to the database. One way to create the file is to move to the top directory of the node in which all the files are located. For example:

```
cd /usr1/newpr/orig_new1.0
```

Issue the UNIX system `find` command to create a file containing the relative directories and file names for all files under the node where you are located. For example:

```
find * -type f -print > /usr1/newpr/sourcefiles
```

The resulting file might look something like this:

```
src/admin/admin.c
src/admin/makefile
src/headers/ancl.h
src/headers/makefile
```

```
src/lib/libA/a.c
src/lib/libA/b.c
src/lib/libA/makefile
src/lib/libB/c.c
src/lib/libB/d.c
src/lib/libB/makefile
src/netcmd/netcmd.c
src/netcmd/makefile
```



NOTE:

Do not use . as the argument to find; this argument prepends ./ to the file names. Sablime does not permit this prefix.



CAUTION:

The node where you issue the find command may contain files that you do not want to place under Sablime control. Edit the list of files so that it contains only the names of the files you want to place under Sablime control.

4. Create, accept, and assign (to yourself) an initialization MR to allow you to add the new files. This process is most easily accomplished with the fcreate command (see the *User's Reference Manual*).
5. If you establish file ownership, make sure that the user who is going to own this file has a valid PTS ID.

The information in steps 1 to 4 above is requested during the execution of primsdb. *Worksheet 4: primsdb Checklist* and *Worksheet 5: Installation Information* in Appendix E provide some help; please have the completed worksheets available if you call the Sablime hotline for help with primsdb.

Running primsdb.

Move to the directory containing the Sablime commands and execute primsdb by entering the following command at the system prompt:

```
primsdb
```

primsdb performs the following steps.

1. Verifying System Variables
2. Specifying Necessary Information
3. Adding the Source Files

They are described below.

Verifying System Variables.

The first step in the primsdB procedure is to be sure that the system variables set during installation are valid and that the Global Database and the necessary tools exist. If any of these conditions are not met, an error message is displayed and the program ends.

Specifying Necessary Information .

An introductory screen gives a general list of the information you need to have on hand when the program runs.

If you answer **n** to the question, the program exits.

If you answer **y**, each prompt is displayed individually and you are expected to respond with the requested data. If you give an invalid response, you are prompted to enter the correct data until valid data is accepted. Each prompt is explained in detail below.

- PROMPT: Are the Files Being Added for the First Time [y or n] ?
- RESPONSE: If the files to be added to the database for the new generic are already in Sablime under an existing generic, enter **n**; otherwise, enter **y**.
- PROMPT: Enter the MR Number Assigned to YOU to Add Files:
- RESPONSE: Enter the number of the MR with which you are adding the files. The MR class determines the options available for the File Type prompt.
- PROMPT: Enter the OWNER of the files being added ?
- RESPONSE: No entry is necessary; this is an optional field.
- If an owner is specified, only the owner is allowed to modify the files; no other user can modify them.
- Enter the PTS ID or the name of a group containing PTS IDs of the owner of the files being added. All the files must be owned by the same PTS ID or group.
- If the files are copied from another generic and you want to retain the previous owner, enter the word **same**. If no owner is desired, press RETURN.

The list of languages available is a function of the MR class. Assuming that the specified MR is of the software class, the following list appears.

PROMPT: File Type:

bal	fortran	perl
c	html	pli
c++	java	sal
cobol	jcl	shell
comp	limbo	snoflake
dbd	lke	spitbol
dirjcl	mark4	vb
document	mll	other
dtp	parms	

Pick 'ONE' from list of choices:

RESPONSE: Enter one of the languages from the list. Lists are dynamically selected based on the MR class. The Sablime Administrator can customize the lists for each product. An entry is mandatory.

PROMPT: Are the Files Binary [yes, no, or auto-detect]?

RESPONSE: Enter **yes**, **no**, or **auto-detect** to indicate whether the file is binary. The criteria for determining whether a file is binary for Sablime purposes are:

- n line length greater than 1021 characters on any line *or*
- n octal 1 (^A) in the first column of any line, or
- n does not end in a newline



NOTE:

Unlike the UNIX "vi" editor, most text editing programs do not force a file to end in a newline. SCCS (and thus Sablime), though, still requires the ending newline in order to treat the file as non-binary.

If you enter **yes**, SBCS is automatically entered as the version control tool. If you enter **no**, primsdB will check each file to determine if it is a binary file; if primsdB determines that the file is binary, an error message is produced and the file is not added into the Sablime database. If primsdB determines that the file is not binary, and if SBCS or SCCS is set as the default entry for the *Control Tool* field in the ADM relation (from the *Default Version Control Tool for Non-Binary Files* field in setrel), the default is automatically entered as the version control tool.

If you enter **auto-detect**, Sablime checks the file to determine if it is a binary file. If Sablime determines that the file is binary, SBCS is entered as the version control tool. If Sablime determines that the files are not binary, and if SBCS or SCCS is set as the default entry for the *Control Tool* field in the ADM relation, the default is automatically entered as the version control tool.

If your files are not properly sorted by binary status, primsdB fails. Check the SDBbad.*number* file in your home directory, where *number* is the background process number of primsdB.

PROMPT: Enter the Version Control Tool [SBCS or SCCS]:

RESPONSE: This prompt appears only if the files are non-binary and the default entry for the *Control Tool* field in the ADM relation is set to **Either**.

Enter the version control tool of choice.

PROMPT: Are the Files Being Counted for Quality Assurance Reports [y or n] ?

RESPONSE: This prompt appears only for non-binary files. Enter **y** to include the files for QA counting purposes; enter **n** to exclude them.

PROMPT: Enter the Full Path to the File
Containing the List of Files to be Added:

RESPONSE: Enter the full path to the file containing the list of files to be added to the Source Database. This is the list you created before executing `primsdb`. (See *Preparing to Run primsdb*, above.)

For example, enter:

```
/usrsl/newpr/sourcefiles
```

PROMPT: Enter the Full Path to the root of the Node where the Files Reside:

RESPONSE: Enter the root of the node where the files to be added are stored. For example, if you are working in the v5.0 generic, and the files you want to add are in `/home/mozart/ral/v5.0/manual/admin`, you would enter `/home/mozart/ral/v5.0`.

If you are adding files from a previous generic, the following prompts are displayed:

PROMPT: Enter the Previous Generic To Extract Sourcefiles From:

RESPONSE: Enter the name of the Sablime generic from which the source files will be added to the new generic.

PROMPT: Are the Sourcefiles Common Between New & Previous Generic [y or n]?

REPOSENSE: Enter **y** if all added files should be made common between the generics. (See the `common` command in the *User's Reference Manual* for more information.) Otherwise, enter **n**.

If you enter **n**, the following prompt is displayed; otherwise, the files are added from the MR branch and the procedure for adding the source files begins.

PROMPT: Propagate the Sourcefiles From Which Branch [ofc or mr]?

RESPONSE: Enter the branch from which the files should be added. (See the information about branches in the section on *Source File Control* in Chapter 5, *Using the Source Commands*, in the *User's Guide*.) Enter **ofc** for the official branch; enter **mr** for the unofficial branch.

PROMPT: Group of MRs for Specifying Version:

RESPONSE: This is an optional field. If it is used, enter the group of MRs (previously created with the `setgroup` command) to be used to determine which changes are to be carried over to this new generic.

The version of the files to be retrieved is governed by the factors in Table 4-1.

Table 4-1. MR Branches and Version Retrieved

Branch	MRs Specified	Version Retrieved
ofc	—	Latest official branch version
	Unapproved MRs	Latest official branch version plus changes associated with specified MRs
mr	—	Latest MR branch version
	Approved and/or unapproved MRs	Earliest MR branch version plus changes associated with specified MRs

See the `addsrc` manual page in the *User's Reference Manual*, for a full description of the way the branch and list of MRs affect the version of files added to the new generic.

Adding the Source Files.

After you have successfully supplied the requested information, the source files you have designated are added to the new generic. The following message is displayed while this procedure is taking place:

```
+++++
THE PROCESS OF ADDSRC'ing FILES INTO Sablime
IS NOW RUNNING IN BACKGROUND

The background job number is [ job number ]

The output files will be placed in your
home directory with the names:

    [full path]/SDBgood.[number]
    [full path]/SDBtrace.[number]
    [full path]/SDBbad.[number]
+++++
```

You can use the UNIX `ps` command with the background job number to find out whether the process has finished. Once `primfdb` is running in the background, you can log off if you like; `primfdb` continues to execute.

Output Files.

As `primfdb` executes, the following three output files are produced in your home directory to show the results of the processing; *number* is the foreground process ID number of the `primfdb` script.

n `SDBgood.number`

This file contains verbose information about the files that were successfully added. Messages in this file are similar to the following:

```
+++++
+++++
SUCCESSFUL: addsrc g=generic mr=mr_number srf=file\
dir=src/include\
initsrc=/usr/wolf/amr/generic/src/include/file\
fltype=document bfile=no

+ Processing the inputted data; please stand by!

+ File [file] is successfully added under the\ Sablime
Version Control.
```

n `SDBtrace.number`

This file contains terse information about the files that were successfully added. Messages in this file are similar to the following:

```
SUCCESSFUL: srf=file      dir=src/include
```

n SDBbad.*number*

This file contains verbose information about the files that were not successfully added. Messages in this file are similar to the following:

```
+++++
BAD EXEC: addisrc g=generic mr=mr_number srf=file
          dir=. initsrc=/tmp/./file
          fltype=document bfile=no vct=SBCS

Err[ 999]: A document file cannot be addisrc-ed in the
          . directory.

Err[9999]: Argument errors detected; [addisrc]\
          execution terminated.

+ A Master Trace Record has been generated for the\
  Database Administrator.
```

If you need to add more files having a different language or a different owner or a different QA flag, repeat the primsdb steps again for each set of files.



CAUTION:

The MRG used to add the files should always be submitted immediately after adding the files so that the official branch will contain the official version of the files as a base. Until the initial MR is approved, zero-length files will exist in the official branch. Also, making further modifications with the initial MRG makes it difficult to get back to the version of the files containing those modifications.

Moving, Renaming, and Deleting Files

The source command can be used to move and/or rename a file in the Sablime databases, and to delete a file from the databases.

The source command can be used in single-machine mode or only from the host machine in multi-machine mode.



NOTE:

Before you run the source command to move, rename, or delete a file, the Source Administrator should log in as *sablime* and execute the dbstop command to shut down the Sablime database.

Only one file may be specified at a time, but if a list of generics is provided, the command will move, rename, or delete the file in all of them.

If the specified file is out for edit, no operations are allowed by the `source` command.

To run the `source` command, the Source Administrator must:

1. Log in as *sablime*.
2. Run the `dbstop` command to freeze the databases (this is optional, but recommended).
3. Run the `source` command, specifying the type of operation desired, i.e. delete, move, or rename.
4. Run the `dbstart` command to unfreeze the databases.



NOTE:

If a file is deleted accidentally, it can be restored in the Sablime databases by a series of manual operations. See the following section for instructions.

Restoring a Deleted File to a Generic

If a file is accidentally deleted from the Source Database (SDB) using the `source` command, the following procedure allows you to restore it. This procedure will only work if one of the following is true:

- n The file was deleted from all the generics in which it was common and you want to restore it to all the generics.
- n The file was deleted from one or more generics and there are one or more other generics from which it was not deleted, but no work has been done on the file in the other generics since the deletion.



NOTE:

If there are generics from which the file was not removed, but work has been done on the file in those generics, you cannot restore the deleted file using the `source` command. Instead, you must `addgsrc` the file from an existing version into the generic from which it was deleted. But note that this procedure does not restore the MR history for the deleted file.



CAUTION:

Do not attempt this procedure unless you are comfortable with the UNIX system commands and are confident that you understand the procedures outlined. If in doubt, please contact the Sablime hotline for assistance.

To restore a deleted file, perform the following steps:

1. Locate the file you want to restore. All files stored for possible recovery by the `source` command are in the Active Database (ADB) in a directory named for your product followed by a subdirectory called `recover` (e.g., `adb/$sabPROD/recover`). There are two files associated with each deleted file. Both file names are similar, the lone difference being that one of the files ends with **S**. The first thirteen characters of both file names are composed of the date and time of the files creation and the process ID of the source command that produced them. They are of the form:

yydddhmmpppp

where:

- n *yy* is the last two digits of the year the file was created
- n *ddd* is the Julian date on which the file was created (e.g., Jan. 31 = 031)
- n *hh* is the hour of the day on which the file was created in military time (e.g., 2 p.m. = 14)
- n *mm* is the minute of the hour in which the file was created (e.g., 22)
- n *pppp* is the hexadecimal representation of the process ID (e.g., 4a52) of the process that created the file

The file whose name does not end with **S**, contains a 4-line header to help you identify the correct file for restoration. Line 1 of the header contains the relative path of the file that was removed. Line 2 identifies who issued the `source` command, what operation was performed, and the date of the operation. Line 3 specifies the generics affected by the operation. Line 4 contains the location of the SCCS or SBCS source archive file.

The following example displays the first 4 lines of the recovery file `980552217378e`.

Example:

```
Sablime Source command Recovery File for src/admin/pts.c
[Sablime] deleted src/admin/pts.c; date: Fri Feb 24 22:17:11
1998
Affected generics: v4.2
The s.pts.c is stored in /home/sablime/adb/prod/recover/
980552217378eS
```

2. Once you have found the files you want to restore, make a copy of them.



CAUTION:

Never edit a recovery file directly.

Example:

```
cp 980552217378e recover.file
cp 980552217378eS s.file
```

where *file* is the name of the file being restored.

3. Move the s.file to the SDB.



CAUTION:

If the s-file already exists in the SDB, the file was not removed from all generics. Overwrite the existing file only if you are sure that no work has been done on this file since it was deleted.

Example:

```
mv s.file $sabBASE/sdb/$sabPROD/dir
```

where *dir* is the relative directory name contained in the first line of the recovery file header.

4. Extract the deleted database records and write them to a file.

Example:

```
sed -e '1,/ Start of deleted records /d' \
-e '/ End of deleted records /,$d' \
< recover.file > recs.del
```

5. Each deleted record must be returned to the ADB or IDB. Locate the appropriate database directory for replacement. The directory is in the *adb/product/rename* directory where *product* is the name of your product and *rename* is the name of the relation to which the record is to be returned (e.g., *adb/sab++/GS*).

Example:

The deleted record starts with a string like the following:

```
oldGS:A:am:create.c;mrmgmt ...
```

- n oldGS indicates that the record was in the GS relation.
 - n The letter A indicates that the record came from the Active Database (the letter I indicates the Inactive Database).
 - n am is the name of the tuple file from which the record was deleted.
 - n create.c;mrmgmt ... begins the actual record.
6. Extract the record from the deleted record information.

Example:

```
cut -d ":" -f4 recs.del > oldrec
```

7. Append it to the am tuple file in the GS relation of the Active Database.

Example:

```
cat oldrec >> ~adb/Sablime/GS/am
```

8. If only some of the generics in which the file exists were affected by the source command, you must also extract the section of split records so that only the records for the appropriate generics are included in the restoration. Split records are created by the source command when a file common to several generics is not deleted from all generics.

Example:

```
sed -e '1,/ Start of GS split records /d' \  
-e '/ End of GS split records /,$d' \  
< restore.file > recs.split
```

9. The extracted records are in pairs, tagged oldGS and newGS. Extract the old GS records from the list of split records and append them to the appropriate GS relation tuple file as described above.

Example:

```
grep "^oldGS" recs.split > gs.restore
```

Managing the Databases

This section covers the following topics: changing data in the Active, Inactive, and Global Databases, auditing the databases, checking for disk space, and checking the system error message file.

Changing Data in the Databases

Changing Data in the MR Relation

The mredit command allows the MRA to edit the data stored in the MR and ORG relations. (The command may also be used to edit a Description file that has been entered using the create, fcreate, review, or spawnmr commands.) Data can be edited for any active MR. (See *The MR Relation* in Chapter 5, *The Sablime Databases*, for detailed information about the MR relation.)

Changing Data in the MRG Relation

The mrgedit command allows the GA to edit the data stored in the MRG relation and in the Rejection, Resolution, Solution, and Spawn Notes Files populated using the MR management commands. Data can be edited for any MR accepted into a generic.

Changing Data in Other Relations

The `dbedit` command allows the DBA to edit directly any of the relations in the Active, Inactive, or Global Databases. This command locks each named database tuple file so that it cannot be accessed by other commands while it is being edited. It then puts the file into the administrator's favorite editor for changes. When the administrator leaves the screen, the file is unlocked.

The DBA should be familiar with the databases and the relation structures before attempting to edit the databases. See Chapter 5, *The Sablime Databases*, for information about the databases.



NOTE:

In multi-machine projects, the `dbedit` command must be run on the host where the databases are located.



CAUTION:

Only highly experienced Sablime DBAs should use this command to change database information. If a change requires editing for an extended period of time, shut down the Sablime databases with the `dbstop` command use `dbedit` to make the necessary change, then start the databases again with the `dbstart` command.

Auditing the Databases

The Sablime database Audit Programs are used to detect corruption and inconsistencies in a set of Sablime databases. They compare the Active, Inactive, and Source Databases and inform the Database Administrator (DBA) of any discrepancies found.

Database corruption can occur as a result of a system going down, improper editing of the database, lack of space in a file system when updates were attempted, or a variety of other reasons.

If an error message indicates that something is wrong, but no data are provided, for example:

```
<*> The following MR names from MG are not in MR:
```

it means that the relation named in the message header contains a tuple file that includes a blank line. To find the tuple file that needs to be corrected, go to the directory containing the named relation and issue the following UNIX system command:

```
grep ^$ *
```

This command will return the file name for the tuple file containing the blank line. You can then delete that line and restore database integrity.

Three of the four audit programs can be run by any user; `hotline.ck` is available only to the `sablime` login. It is best to execute them on a nightly basis through the UNIX system `cron` utility. If a machine crashes and is brought back up, you should run the audits immediately to check for corruption.

At the beginning of `hotline.ck` is a variable called `TMPDIR`, which is where the temporary audit files created during execution are stored. The default for this directory is `/usr/tmp`. If this directory does not have enough free blocks or inodes for the audit files, you can change the variable to a different location.

If the audit programs are run while Sablime is running, you will not get an accurate account of the databases. It is best to run the audits at a time when the activity is minimal (e.g. at night) or while the databases have been stopped with the `dbstop` command.

The results should be checked each morning to verify the integrity of the databases. The audit programs produce output reporting any inconsistencies or corruption found in the Active, Inactive, and Source Databases. Before you change data to correct an inconsistency or corruption, look through the related database relations in all the databases to be sure where the problem occurred.

The four audit programs are described in the table below:

Table 4-2. The Audit Programs and their Functions

Program	Function
<code>dbcross</code>	This program does a cross-check of basic Active and Inactive Database relations by comparing the contents of certain records to the contents of other tuple file records.

Table 4-2. The Audit Programs and their Functions

Program	Function
dbxcross	This program does a cross-check of all the database relations for the External Communications feature by comparing the contents of certain records to the contents of other tuple file records. If you are not using this feature, you do not have to run this audit program.
dbdelta	This program compares relations containing information about files in the Active Database with the information in the Source Database where the files actually reside. If you do not use Sablime for file control, you do not have to run this audit program.
hotline.ck	This program checks for permissions of commands, database directories, and files, and makes some other checks as well. Run this program when the Sablime commands are not running or are generating system error messages.

**NOTE:**

Before executing `hotline.ck`, set the variables in the `spacecheck` script. The `hotline.ck` program calls `spacecheck` to check for available space and inodes. (See *Checking for Disk Space*, below.)

**CAUTION:**

`dbcross`, `dbxcross`, and `dbdelta` have options (in particular `g=generic`) that permit the user to limit the audits to certain subsets of the database. Since this could allow database discrepancies to be overlooked, these options should only be used when performance or database size considerations demand. Note also that successive auditing of each generic using "g=" is not equivalent to a complete audit.

**NOTE:**

The `dbcross`, `dbxcross`, and `dbdelta` programs can be run in single-machine mode or from the host machine in multi-machine mode.

**NOTE:**

If your project has converted from CMS to Sablime, errors may occur when you run the `dbdelta` program because of a discrepancy in the MD relation and in the SDB between Sablime and CMS. To prevent these errors, enter a CMS flag and the date on which the conversion was performed when starting the `dbdelta` program as follows:

```
dbdelta -cms yymmdd
```

- n -cms is a flag that informs dbdelta that the databases were converted from CMS
- n *yymmdd* is the year, month, and day when the conversion took place.

To execute the audits nightly using the cron utility, take the following steps:

1. Create a shell program to run the three audit programs. A sample program, named run.audits, is given below.

```
###
# Set all necessary variables for audits to run.
###
unset sabGDB sabGEN sabHOST sabMCB sabNET sabPROD

sabGDB=/usrsl/Sablime/gdb# Location of Global Database
sabGEN=sab5.0           # Generic Name
sabHOST=mozart
sabMCB=/users2/Sablime/sabbin
sabNET=0                # Network
sabPROD=Sablime        # Product
sabVAR=/usr6/Sablime/bin/.varfile # Variable file

export sabGDB sabGEN sabHOST sabMCB sabNET sabPROD

###
# Set the location of where the audit output should be
# written
###
AUDITS=/usr6/Sablime/audits

###
# Remove yesterday's dbdelta output
###
if[ -f $AUDITS/delta.out2]
then
    rm $AUDITS/delta.out2
fi
if[ -f $AUDITS/delta.err2]
then
    rm $AUDITS/delta.err2
fi

###
# Move today's dbdelta output
###
if[-f AUDITS/delta.out1]
then
    mv $AUDITS/delta.out1 $AUDITS/delta.out2
fi
if[-f AUDITS/delta.err1]
then
    mv $AUDITS/delta.err1 $AUDITS/delta.err2
fi
```

```
###
# Run dbdelta and write the output to delta.out1
###
nohup $sabMCB/dbdelta > $AUDITS/delta.out1 2>$AUDITS/delta.err1 &
###
# Wait until finished
###
wait

###
# Remove yesterday's dbcross output
###
if[-f $AUDITS/cross.out2]
then
    rm $AUDITS/cross.out2
fi
if[-f $AUDITS/cross.err2]
then
    rm $AUDITS/cross.err2
fi

###
# Move today's dbcross output
###
if[-f $AUDITS/cross.out1]
then
    mv $AUDITS/cross.out1 $AUDITS/cross.out2
fi
if[-f $AUDITS/cross.err1]
then
    mv $AUDITS/cross.err1 $AUDITS/cross.err2
fi

###
# Run dbcross and write the output to cross.out1
###
nohup $sabMCB/dbcross > $AUDITS/cross.out1 2>$AUDITS/cross.err1 &

###
# Wait until finished
###
wait

###
# Remove yesterday's dbxcross output
###
if [-f $AUDITS/xcross.out2]
then
    rm $AUDITS/xcross.out2
fi
if [-f $AUDITS/xcross.err2]
then
    rm $AUDITS/xcross.err2
fi

###
```



```
# Move today's dbxcross output
###
if [-f $AUDITS/xcross.out1]
then
    mv $AUDITS/xcross.out1 $AUDITS/xcross.out2
fi
if [-f $AUDITS/xcross.err1]
then
    mv $AUDITS/xcross.err1 $AUDITS/xcross.err2
fi

###
# Run dbxcross and write the output to xcross.out1
###
nohup $sabMCE/dbxcross > $AUDITS/xcross.out1 2>$AUDITS/xcross.err1
&

###

###
# Inform the Sablime login the audits have finished
###
echo The audits have completed, please check them out.|mail Sablime
```

**NOTE:**

If you redirect the stderr and stdout streams to the same output file for commands dbdelta, dbcross or dbxcross, system error messages and the like may get interspersed with the audit output. The above example script has each command sending its "stderr" output to a separate file.

2. Use your editor to create a cron file to tell the UNIX system when to execute the program. A sample file, called `audit.cron`, is given below.

```
* 4 * * 1-6 /usr6/Sablime/audits/run.audits
```

This example causes the shell `run.audits` to execute at 4:00 a.m., Monday through Saturday.

3. Save the file.
4. Write the file to your system's cron table using the UNIX system `crontab` command. See the *UNIX System User's Reference Manual* for more information about the parameters and permissions needed to use the `crontab` command.

Example:

```
crontab audit.cron
```

Checking for Disk Space

After your product has been installed and customized, it is a good idea to monitor space conditions on the file systems where the databases are stored. A script located in the Master Control Bin called `spacecheck` can be used to accomplish this task.

**NOTE:**

The `spacecheck` script is run by the audit programs, typically every night, but it is important to run it frequently throughout the day to avoid space problems.

The `spacecheck` script should be set up on a cron and executed every 5 to 15 minutes to check for the number of free blocks and free inodes on the specified file systems. When the free blocks and free inodes reported drop below a predefined amount, the databases are automatically stopped for that product, mail is sent to the Sablime Administrator, and a log file is created in the `gdb/tmp/product_log` directory, where *product* is the name of your product. The `spacecheck` script continues to monitor the space situation until the Administrator makes space available, at which time the databases are automatically restarted.

Before you can use it, you must modify `spacecheck` by setting the following variables, which can be found near the top of the script:

Table 4-3. The Variables in `spacecheck`

Variable	Function
FSDB	The file system(s) that should be monitored, e.g., <code>FSDB="/usr1" "/usr2"</code> . When more than one file system is to be checked, separate the path to each file system by a space.
STOPSPACE	The minimum number of free blocks before the databases are shut down, e.g., <code>STOPSPACE=2000</code> . Preset to 10,000.
SADMIN	The Sablime Administrator PTS ID that should receive mail after the database has been shut down, e.g., <code>SADMIN=sablme</code> .
STOPINODE	The minimum number of inodes before the databases are shut down, e.g., <code>STOPINODE=100</code> . Preset to 1,000.

After these variables have been set, write and quit out of your editor. You can now proceed to add the `spacecheck` script to the cron table. The following steps describe how to do this:

1. Use your editor to create a file (e.g., `space.cron`) to inform the UNIX system when to execute the `spacecheck` script. The contents of a sample file called `space.cron` are shown below:

```
5,20,35,50 06-21 * * 1-5 /usr6/Sablme/bin/spacecheck
```

This example causes the shell script `spacecheck` to start execution at 5 after the hour and then every 15 minutes after that, from 6 am to 9 pm, Monday through Friday.

2. Write the file to your system's cron table by using the UNIX system crontab (1) command (e.g., crontab space.cron).

Checking the System Error Message File

A file that logs the error messages generated by Sablime for each product is maintained in the tmp directory in the Global database. The file expands in size as Sablime commands are executed. The name of this file is based on the product name and is of the form:

product-name_dbawarn

where *product-name* is the name of your product.

This file should be cleaned periodically to reduce disk space use. The failure messages listed in the file do not necessarily mean there are problems with the Sablime commands. Consider the following error messages:

```
04/30/96 18:39:52 [wina] create @(#) send_mail.c \  
[Version 2.1.1.3] - dba_msg: No mail will be sent \  
to [scott] - Can't find PTS record..  
04/30/96 18:40:36 [wina] create @(#) send_mail.c \  
[Version 2.1.1.3] - dba_msg: No mail will be sent \  
to [scott] - Can't find PTS record..
```

These messages indicate that there were problems encountered when the create command finished execution and an attempt was made to send mail to scott. The create command is not at fault; the problem is that scott does not have a PTS ID for the product.

Changing Machines

If the machine on which Sablime is installed undergoes a name change or if an instance of Sablime is moved to a new machine, you must make some changes to the Sablime programs and databases to allow Sablime to run correctly on the new or changed machine. To do this, follow the steps below. (See Chapter 2, *Installing Sablime*, for more information.)



NOTE:

You must also call the Sablime hotline for a new key.

1. Edit the `sablime.sh` and `xsablime.sh` files to reflect the new location of the Sablime binary files (`$sabMCB/$sabLCB`) and the new name of the host machine (`$sabHOST`).
2. Set the following variables:

sabLCB	sabGDB
sabBASE	sabDIRF
sabHOST	sabMFR
sabMCB	sabVHELP

3. Run the `sh2x` program for both `sablime.sh` and `xsablime.sh` as described in *Customizing the dot sablime Script* in Chapter 2.
4. Edit the PR relation in the GDB to reflect the new host machine name (field 4), the new MCB (field 5), and the new ADB (field 6), IDB (field 7), and SDB (field 8). (See Chapter 5, *The Sablime Databases*, for a description of the PR relation.)

Customizing Sablime Mail

Many Sablime commands automatically send mail messages to affected users and administrators about an activity they have performed. While individual users can turn mail on and off for themselves, it is also possible to turn this automatic mail on or off at the product level. (See the `pts` command in the *User's Reference Manual*.)



NOTE:

Even if the mail flag is off, messages still go to users in a *Copy To* list.

To do this, it is necessary to set the *Mail Flag* field to **y** (for on) or **n** (for off). When you are installing a product, the `initsab` script asks you if you want mail turned on for all developers and sets the mail flag accordingly. But once a product is set up, you must use the `ADM` subcommand of the `setrel` command to turn the mail feature on or off.

To turn the mail feature on or off after a product has been set up:

1. Log in as `sablime` on the host machine
2. Set up the environment for the product for which you want to change the mail flag by executing:

```
. sablime generic
```
3. Execute the `setrel` command.
4. Select the **ADM** relation and confirm to enter the second screen.
5. Select the **modify** function.
6. Enter `dba_product-name` as the dba flag, where *product-name* is the name of your product.

7. Change the *Mail Flag* field to **y** to turn on or **n** to turn off the mail feature.

Alternatively, you can use `mmail` to turn the mail feature on, and `nomail` to turn it off. (See the manual page for `mmail/nomail` in Chapter 6, *The Administrative Commands*.)

Mail Consolidation

The users who receive mail messages from the Sablime commands are listed on the manual pages in the *User's Reference Manual* and *the Administrator's Guide*. For example, when an MR is created, the MR administrator receives mail. When an MR is fcreated, the Generic Administrator, MR Administrator, and Assigned Developer receive mail messages.

Sablime consolidates mail messages. Whenever a mail-sending command is used, it makes a list of all the email recipients, consolidates and eliminates duplicate email addresses, and then sends out mail messages. The result is that, for every invocation of a command, a user receives only one mail message. For example, if the Generic Administrator and the Test Team Member are the same user, only one message is sent to that user.

The contents of a typical email message from the execution of a Sablime command is shown in the following figure.

```
Subject: Fcreated MR

Initiator: connielu

MR [mclu990219] has been created by [connielu] and
has been assigned to [connielu].

<----- INFORMATION FOR MR mclu990219 ----->

Product: mclu                Release Detected: not_applicable
System: none                 MR Severity: 3
Subsystem: not_applicable    Required Date:
                              Originator: connielu

Abstract: s

                ----- GENERIC: mclul.0 -----

Class: software              MRG Severity: 3
Type: modification          Due Date:

[ Email abbreviated: the "Verbose Email" flag in your PTS record is off. ]

<----->

This mail is being sent to you in your capacity as:

[connielu@lucent.com]:  CC List, Database Admin
[priffle@lucent.com]:  Assigned Dev, CC List
[sablime@lucent.com]:  Database Admin, MR Admin

*****
On top of this change, MailServer will also become a library function instead of
a separate process forked by send_mail.c.
```

Figure 4-3. Mail Consolidation Message

How Mail Works

When Sablime mail-sending commands successfully complete their principal activity, the last action they perform is to send mail about the function just performed. The command instructs a process called MailServer.c to process the mail. This process will be forked so that commands do not wait for all mail to be sent.

This process first creates two temporary files. One file will contain the mail message for users with the verbose flag set to y, and the other will contain the mail message for users with the verbose flag set to n. The process appends the list of the users who will receive this mail to the end of each file. The list of users will include the role for each user. In this way, all users who want to receive verbose mail message will see the list of other users who want to see verbose mail messages, and all users who want to see concise mail messages will see the list of others who want to see concise messages.

After the temporary files have been created, the process calls popen on /bin/mail for each set of users.

Customizing Mail Recipients

For many commands, the Database Administrator can control to whom the command will send email. For detailed information on how to do this, see the following section, *Customizing Command Executors and Email Recipients*.

Customizing Command Executors and Email Recipients

For many commands, the Database Administrator can control, for each command-function within a generic, who the executors and email recipients will be. Both are controlled by the Command Permissions (CP) relation in the Active Database, which has the following structure:

Table 4-4. Definition of Command Permissions Database Relation

Field	Field Name	Internal Key	Values
1	command	cmd (key)	command name (only one)
2	generic	g (key)	generic name (only one, if applicable)
3	function	fcntype (key)	type of function (only one, if applicable)
4	executor	exec	executor-list (comma-separated)
5	email	email	email recipient (comma-separated)

The CP relation is created by initsab when a new product is installed and by conversion scripts when Sablime is upgraded to v5.0. The records in the CP relation can only be created, modified, or deleted by the Database Administrator using the CP subcommand of `setrel`.

**NOTE:**

In the case of commands which implement related but distinct functionality (such as `setrel`), the command's functionality is included in the command field, separated by a "-". For example, `setrel-ADM` would control the use of the `setrel` command to edit the ADM relation.

The acceptable values for the executor field appear with their definitions in the following table:

Table 4-5. Valid Executor Values

Value	Definition
__ALL	all product-authorized PTS IDs
__NONE	no one may execute
PTS ID	PTS ID
__AD	Assigned Developer
PTS ID Group	all group members
__DEFAULT	as-delivered permissions

The order in which the values are listed is also the checking order. The check is an OR check. `__ALL` and `__NONE` cannot be entered as part of a comma-separated list of values; they must appear alone. The `__DEFAULT` keyword, if it is included, is always used as a last resort, after all other matches have been attempted.

Sablime is delivered with default values for the execution of all command-function combinations. The Database Administrator can use the Command Permissions feature either to increase the number of executors or to reduce their number. If the intent is to increase the number of executors, then `__DEFAULT` should be included in the comma-separated list of executors, e.g. `__DEFAULT,__AD,mkp`. If the intent is to reduce the number of executors, then `__DEFAULT` should not be included. For example, if `mkp` is to be the only one to run `dbstart`, then enter `mkp`. Note that this means that not even the Database Administrator will be able to run `dbstart`, only `mkp`.

The acceptable values for the email field and the corresponding recipients of email are listed in the following table:

Table 4-6. Valid Email Recipients

Value	Recipients
__NONE	no email recipients
__DEFAULT	as-delivered email recipients
__AD	Assigned Developer (MG/MRX assignee)
__ORIG	internal MR originator
PTS ID	PTS ID
PTS ID Group	all group members (group type ptsid)
Email Group	all group members (group type other)
Email Address	email addressee

The email field of the CP relation will take a comma-separated list of values. Keywords like __AD and __ORIG will be expanded into lists of PTS IDs, and groups will be expanded into their group members. The keyword __NONE must appear alone. If used in a list, __DEFAULT will cause the default list of recipients to be included, as well as the other designated recipients.

The table below provides information about permissible entries in the fields of the CP relation for each of the commands affected by the Command Permissions feature. A "yes" in the Email? column means that an email list must be included. A "yes" in the Generic? column means that a generic must be included. A "yes" in the MRs? column means that __AD may be included in the list of executors.

In the Functions column, A stands for add, C for copy, D for delete, L for license management, M for modify, and V for view.

Table 4-7. CP Relation Fields

Command	Functions	Email?	Generic?	MRs?
accept		yes	yes	no
activate-MR		yes	no	no
activate-MRG		yes	yes	no
approve		yes	yes	yes
assign		yes	yes	yes
closemr		yes	no	yes
commit	ADMV	yes	yes	no

Table 4-7. CP Relation Fields

Command	Functions	Email?	Generic?	MRs?
common		no	no	no
create		yes	no	no
dbedit		no	no	no
dbstart		no	no	no
dbstop		no	no	no
defer-MR		yes	no	no
defer-MRG		yes	yes	no
depend	AD	yes	yes	yes
fcreate		yes	yes	yes
ftd	ADMV	yes	no	no
hcode	ACDMV	yes	yes	no
killmr		yes	no	yes
mredit		yes	no	no
mrgedit		yes	yes	yes
mrnote-MR		yes	no	yes
mrnote-MRG		yes	yes	yes
nochange		yes	yes	yes
pdi	ADMV	yes	yes	no
pts	ADLMV	yes	no	no
reject		yes	yes	yes
reserve		yes	yes	yes
review		yes	no	no
setgroup		yes	no	no
setrel-ADM	MV	yes	no	no
setrel-CAS	ADMV	yes	no	no
setrel-CP	ADMV	yes	no	no
setrel-CRIT	ADMV	yes	no	no
setrel-ES	ADMV	yes	no	no
setrel-PR	MV	yes	no	no
setrel-PRX	MV	yes	no	no
smerge		no	no	no
source		yes	yes	no
spawnmr		yes	yes	yes

Table 4-7. CP Relation Fields

Command	Functions	Email?	Generic?	MRs?
study-MR		yes	no	no
study-MRG		yes	yes	no
submit		yes	yes	yes*
testassign	DM	yes	yes	yes
testpass		yes	yes	no
unaccept		yes	yes	no
uncommon		no	no	no
unreserve		yes	yes	yes

* Note for the submit command: After these "Command Permissions" are evaluated, submit still verifies and requires that the submitter is the assignee. Submit is present in the Command Permissions system in order to allow local administrators to turn the command off for a generic and/or to permit customization of the email recipient list.

Contents

5	The Sablime Databases	1
n	Relations, Tuple Files, and Records	1
	Creating a Record in a Tuple File	2
n	Commands and Relations	2
n	Global Database	5
	Global Database Directories	5
	DBLOCK Directory	5
	DIR Directory	6
	rd/sd Directories	6
	rq/sq Directories	6
	tmp Directory	6
	cron Directory	6
	Global Database Relations	7
	The ES Relation	8
	The PR Relation	9
	The PRX Relation	10
	The PTS Relation	11
	The TR Relation	13
n	Active/Inactive Databases	14
	DActive/Inactive Database Directories	15
	DBLOCK in the ADB Directory	15
	DBLOCK in the IDB Directory	16
	FILES Directory	17
	Active/Inactive Database Relations	23
	The ADM Relation	29
	The CAS Relation	30
	The COM Relation	31
	The CP Relation	32
	The CRIT Relation	33
	The DEP Relation	34
	The EMG Relation	35

Contents

The EMR Relation	36
The FTD Relation	38
The FZ Relation	40
The G Relation	41
The GRP Relation	42
The GRPM Relation	43
The GS Relation	44
The GT Relation	45
The HC Relation	46
The MD Relation	47
The MG Relation	48
The MR Relation	52
The MRS Relation	53
The MRX Relation	54
The MS Relation	55
The ORG Relation	56
The PDEP Relation	57
The PDI Relation	58
The SNAP Relation	60
The UMS Relation	61
n Source Database	62
Delta Numbers	63

This chapter contains detailed information about all the Sablime databases. Each Sablime instance contains one Global Database (GDB), and one or more Active Databases (ADBs), Inactive Databases (IDBs), and Source Databases (SDBs). The Global Database contains product, external communication, and personnel information; the Active Database and Inactive Database are parallel in structure and contain MR, generic, and source file information for one or more products; and the Source Database for each product contains all the files for that product. All the directories and files in all the databases are owned by the *sablime* login.

Relations, Tuple Files, and Records

Most of the subdirectories in the GDB and in the ADB/IDB are special directories called relations. Each relation contains tuple files that contain records (lines of information).

Each tuple file is named by a hashing algorithm that uses the first field in a record to produce a file name consisting of two characters, each a letter from *a* to *p*. In each relation, all records with a first field that hashes to the same tuple file name are stored in the same tuple file in arbitrary order. Tuple files are created as needed; empty tuple files are not created in anticipation of records to be placed in them.

For example, suppose you want to find a record in the MR relation for MR number `sab960034`. To do this, you would use the `dbhash` command to find the name of the tuple file in which the information is stored, as follows:

```
dbhash sab960034
```

Sablime will return the tuple file name:

```
mn <--sab960034
```



NOTE:

For easy access to database information, use the report, query, or ssq command; they do not require knowledge of the tuple file hash name.

Information fields in each record are separated by semicolons. Some information fields have several subfields separated by commas. A sample record looks like this:

```
pts;dept;y,y,n;Dept Code:,19,,1404;1,25,;;\  
12,14,_,0,left,Dept Code: ,0,0;dept
```



NOTE:

When records are displayed with a backslash "\" at the end of a line, it indicates that the following line is actually a continuance of the existing line. Neither the backslash or the "return" are part of the database record.

Creating a Record in a Tuple File

The internal key of a record is made up of the first field of the record. For example, in the MG relation, the key fields are the first two fields: MR number and generic. If a command (e.g., accept) must create an MG relation record for an MR number and generic combination, it hashes the first key field (MR number) and finds the tuple file name. Then it looks for that tuple file in the MG relation. If it does not find a tuple file with that name, it creates one with one record for that MR number and generic. If it does find the tuple file, it searches the tuple file serially for the MR and generic combination. If it finds a record with those values, it overwrites that record; otherwise, it adds the record at the end of the tuple file.

Commands and Relations

Table 5-1 provides an alphabetical list of the Sablime commands and the relations they affect.

Table 5-1. Commands and the Relations They Affect

Command	Relation
accept	MG, MR, MRS
activate	MG, MR, MRX
addgen	G, GT, MG, TR

Table 5-1. Commands and the Relations They Affect

Command	Relation
addsrc	GS, MD, MS, UMS
addisrc	GS, MD, MS, UMS
approve	GS, MD, MG, MS, UMS
assign	MG
closegen	COM, CRIT, FZ, G, GS, GT, HC, PDEP, PDI, SNAP
closemr	COM, DEP, EMG, EMR, MD, MG, MR, MRS, MRX, MS, ORG, PDEP, SNAP
commit	COM, MG
common	GS
create	ADM, MR, ORG
defer	MG, MR, MRX
delay	PTS
depend	DEP, PDEP
edget	GS, MD, MS, UMS
edput	DEP, GS, MD, PDEP
fcreate	ADM, MG, MR, ORG
ftd	FTD
getversion	FZ, SNAP
hcode	HC
initsab	all relations
killmr	EMG, EMR, MR, MRX, ORG
listmsgs	none
mk_approve	none
mredit	MR, ORG
mrgedit	MG
nochange	MG
pdi	PDI
propose	MG, MR, MRX
pts	PTS
qmr	EMR, MG
reject	MG
reserve	MD, GS
review	EMG, EMR, MG, MR, ORG

Table 5-1. Commands and the Relations They Affect

Command	Relation
setgroup	GRP, GRPM
setrel	ADM, CAS, CP, CRIT, ES, PR, PRX
snapedit	FZ, SNAP
source	FZ, GS, MD, MS, PDEP, UMS
spawnmr	MG, MR, MRS, ORG
study	MG, MR, MRX
submit	HC, MG, PDI
smerge	none
testassign	MG
testpass	MG
unaccept	MR, MG, DEP
unassign	MG
uncommon	GS
unedget	GS, MD, MS, UMS
unedput	DEP, MD, MS, PDEP, UMS
unreserve	MD, GS

Global Database

The Sablime GDB contains information about the products that constitute a Sablime instance, product directory structures, external projects, and about personnel using the Sablime instance.

The GDB is located in a directory called `gdb` under the `sablime` login. Assuming that the `sablime` login's home directory is `/user/sablime`, the GDB directory structure contains subdirectories as shown in Figure 5-1:

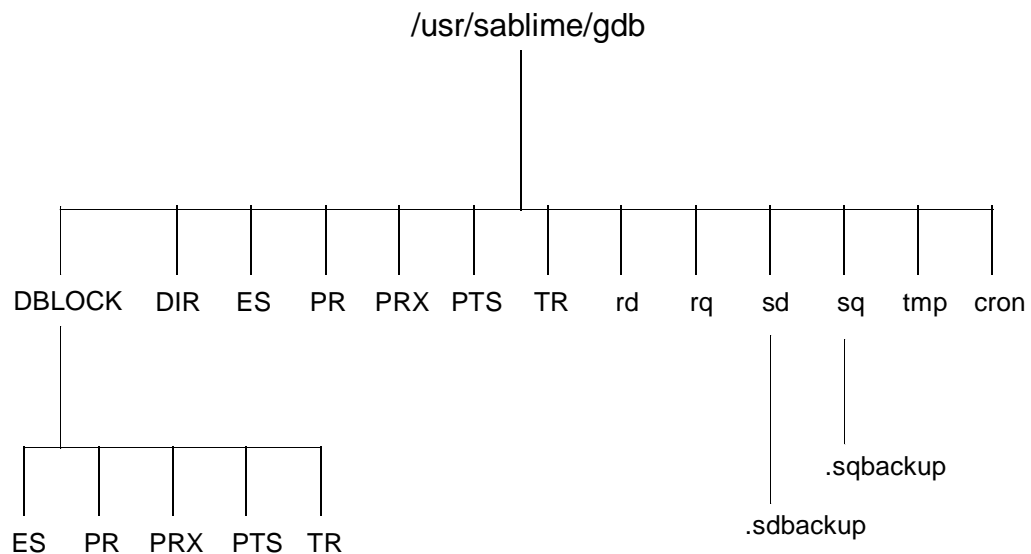


Figure 5-1. Global Database Directories and Relations

Global Database Directories

DBLOCK Directory

DBLOCK contains a set of directories matching the relations in the Global Database. This directory allows a tuple file to be locked when it is accessed or updated by a Sablime command.

DIR Directory

DIR contains a set of files that define the templates for the directory structure that holds the files for each generic of the products in the Sablime instance. In this directory, for each generic there is one file with the same name as that generic. Each file contains the list of directories and subdirectories that make up the directory structure for that generic.

rd/sd Directories

The rd/sd directories contain text description files received from or sent to external projects using the Sablime External MR Communications feature.

rq/sq Directories

The rq/sq directories contain the queues of messages received from or sent to external projects using the Sablime External MR Communications feature.

tmp Directory

tmp is used by Sablime as temporary work space to manipulate tuple files and records. It also contains the database warning file for every product in the instance. The format of the warning file name is *product_dbawarn*, where *product* is the name of a product in the Sablime instance.

cron Directory

The cron directory contains type 16 messages placed there by *sabmldm* and waiting to be further processed by *sabrepdm*, which sends reports to users who have requested them.

Global Database Relations

The GDB relations are presented in alphabetical order as shown in Table 5-2. The first column gives the name of the relation, the second shows the type of information stored in the relation, the third contains the name of the hash field for the relation, and the last lists the command or commands that create or update the information in the relation.

Table 5-2. Global Database Relations

Name	Information	Hash Field	Command
ES	External Source	External Project	setrel
PR	Product	Product Name	initsab setrel
PRX	Product Extra Information	Product Name	initsab setrel
PTS	Personnel	PTS ID	initsab pts
TR	Generics by Product	Generic Name	addgen initsab newgen

A detailed description including field names, keys, and examples for each of these five relations is given on the following pages.

The ES Relation

The ES (External Source) relation contains information about each external project with which a Sablime product in this Sablime instance can communicate. The keys for this relation are External Project Name and External Product Name.

There is one record for each external project/product combination. Each record contains the information shown in Table 5-3 gathered through the `setrel (ES)` command.

Table 5-3. ES Relation Fields

Field	Information
1	External Project Name
2	External Product Name
3	Host Machine Name of external product
4	Network Type
5	Name of Communications Program (service)
6	Program Parameters (if any)
7	Flag indicating whether MR state changes should be communicated to the external product
8	Status Flags for each MRG state that will be communicated to the external product (<i>accepted, nochange, deferred, understudy, assigned, submitted, test state 1, test state 2, test state 3, test state 4, test state 5, approved, committed</i>)
9	Flag indicating whether MR state changes should be communicated at link time
10	Flag indicating whether <code>mrnote</code> description files should be communicated to the originating project

Example:

```
sablime;ans;mozart;1;rcv_msgs;;y;y,y,y,y,y,y,y,y,y,y,y,y,y;n;n
sablime;mm;alvis;6;rcv_msgs;;y;y,n,n,n,n,y,n,y,n,y,n,y,y;n;n
sablime;syst;granjon;6;rcv_msgs;;y;y,y,y,y,y,y,y,y,y,y,y,y;n;n
sablime;test2.0;damler;6;rcv_msgs;;y;y,y,y,y,y,y,y,y,y,y,y,y;n;n
```

The PR Relation

The PR (PRoduct) relation contains information about each product supported by the Sablime instance. The key for this relation is Product Name.

There is one record for each product. Each record contains the information shown in Table 5-4 gathered through the `setrel` command or at installation:

Table 5-4. PR Relation Fields

Field	Information
1	Product Name (up to 5 characters)
2	Product Type (internal)
3	Multi-Machine Flag (1 if a multi-machine environment is used; 0 if not used)
4	Host Machine (name of the machine used or name of the host machine in a multi-machine environment)
5	Full Path of the Master Control Bin directory
6	Full Path of the Active Database directory
7	Full Path of the Inactive Database directory
8	Full Path of the Source Control Database directory
9	Not used
10	Not used
11	Not used
12	Not used

Example:

```
sab;internal;0;granjon;/u6/sablime/sab3.0.2/bin;/u6/sablime/\
adb/sablime;/u6/sablime/idb/sablime;/u6/sablime/sdb/sablime;;;
nmake;internal;1;granjon;/u6/sablime/sab3.0.2/bin;/u6/sablime/\
adb/nmake;/u6/sablime/idb/nmake;/u6/sablime/sdb/nmake;;;
```

The PRX Relation

The PRX (PROduct eXtra) relation contains extra product information about each product used in the Sablime instance. The key for this relation is Product Name.

There is one record for each product. Each record contains the information shown in Table 5-5 gathered through the `setrel` command or at installation:

Table 5-5. PRX Relation Fields

Field	Information
1	Product Name
2	Complete Product Name
3	Major Programming Language used for product
4	First Billable Customer of the product
5	Organization responsible for product
6	Sablime Product ID
7	Not used

Example:

```
prod1;Sablime Multi-Machine Test System;c++;sablime;5911;mozart_mm;  
sab;Sablime Software Admin System;c++;sablime;59112;mozart_sab;  
test3.1;The Test 3.1 Project;c++;sablime;59112;mozart_test3.1;
```


The PTS Relation

The PTS (Personnel Tracking System) relation contains personnel information about each authorized user of the Sablime instance. This relation also contains the choices made on how Sablime options will be customized for that individual. The key for this relation is PTS ID.

There is one record for each authorized user of the Sablime instance. Each record contains the information shown in Table 5-6 gathered through the `pts` command or at installation:

Table 5-6. PTS Relation Fields

Field	Information
1	PTS ID (login)
2	Full Name
3	Organization Code
4	Location Code
5	Room Number
6	Phone Number
7	Contains five subfields:
i.	Command Display Mode (choice of interface: fs=Curses Forms interface; np=Command Line interface)
ii.	Verbose Prompts (choice of default prompting: y=display of default on screen or after prompt; n=no default displayed)
iii.	Verbose Info (choice of amount of processing data displayed: y=all data displayed; n=only general information displayed)
iv.	Verbose Help (choice of help level: y=complete help given when requested; n=no help given)
v.	Number of Seconds before Display of Pop-up Menu in Curses Forms interface
8	Favorite Editor
9	Email Address
10	Licensed
11	Receive Mail Flag—Send mail when certain commands execute
12	Authorized Products (in a comma-separated list)
13	Last Usage
14	Automatic Originator Flag—Send mail to PTS ID if PTS ID is MR originator

Table 5-6. PTS Relation Fields—Continued

Field	Information
15	Automatic Assignee Flag—Send mail to PTS ID if PTS ID is MR assignee
16	Verbose Email Flag—Send mail with detailed description
17	Licensed for SabMerge merge tool usage?
18	Date of last SabMerge usage
19	Manager of user
20	Not used

Example:

```
scott;Scott C. Grawford;59553;LC;4NC07;(908)999-5324;y,n,n,n,4;vi;\
scott@company.com;internal;y;sab++,test;;y;y;n;n;01/01/00;lfg;
```

The TR Relation

The TR (TRAnslation) relation contains information about the generics for each product. The key for this relation is the Generic Name.

There is one record for each generic of a product. Each record reports the information shown in Table 5-7.

Table 5-7. TR Relation Fields

Field	Information
1	Generic Name
2	Product Name
3	Not used

Example:

```
test1.1;sablime;  
test1.2;sablime;
```

Active/Inactive Databases

Sablime keeps track of MRs, generics, and information about files in two databases with similar file structures; the Active Database (ADB) and the Inactive Database (IDB). The ADB contains information about the active or current MRs, generics, source files, and groups. The IDB contains information about the closed, killed, and archived MRs and generics.

The ADB is located in a directory called adb and the IDB in a directory called idb. The location for these directories was specified during the initsab script. (See *Running the Installation Script* in Chapter 2, *Installing Sablime*.) Under each of these directories, a subdirectory named for each product contains the product's ADB and IDB. The ADB directory structure is shown in Figure 5-2; the IDB directory structure is shown in Figure 5-3. Each database contains a number of subdirectories as explained later in this section.

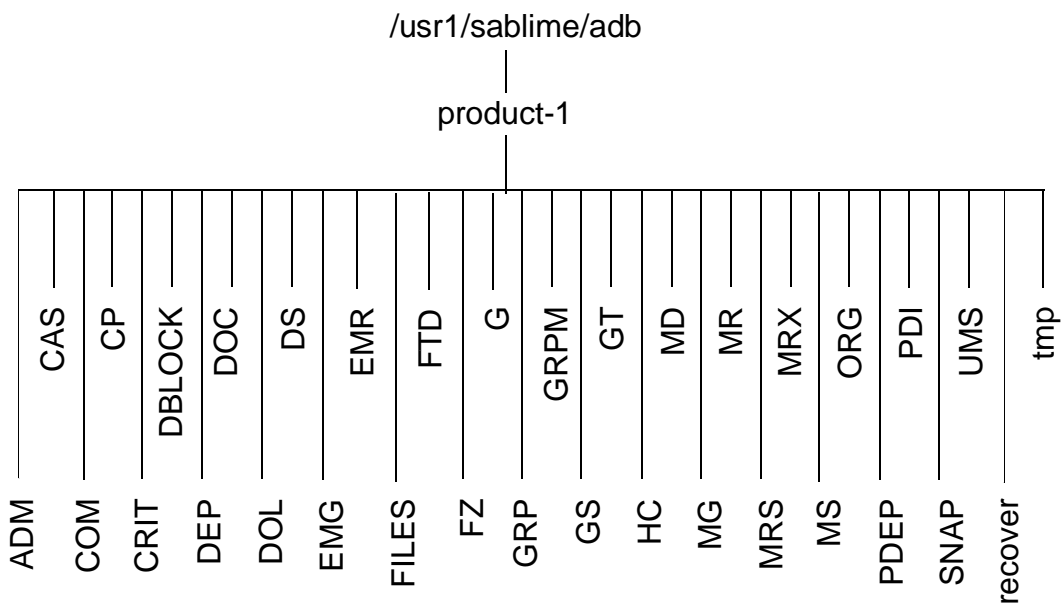


Figure 5-2. Active Database Directories and Relations

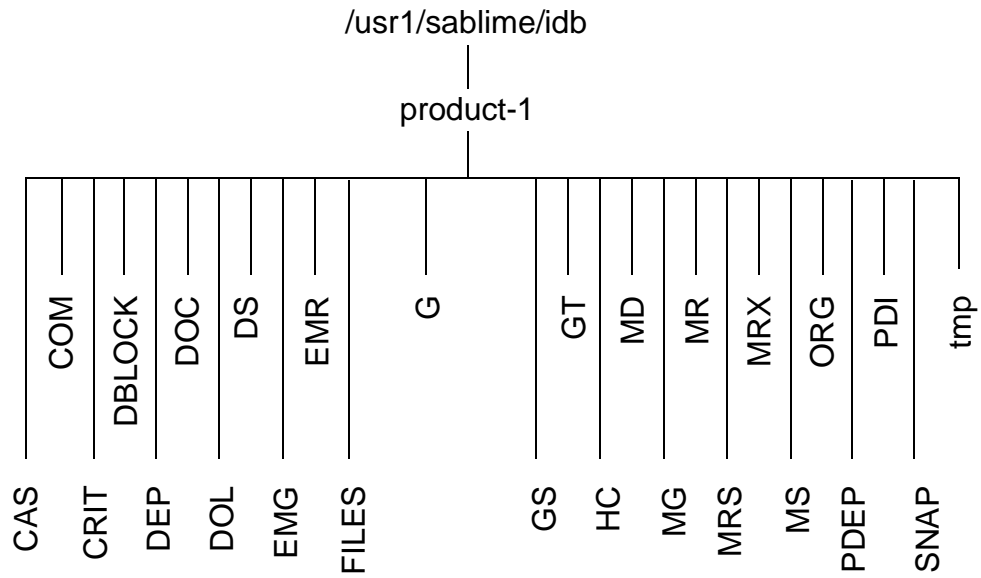


Figure 5-3. Inactive Database Directories and Relations

DActive/Inactive Database Directories

DBLOCK in the ADB Directory

The DBLOCK directory structure in the ADB is shown in Figure 5-4.

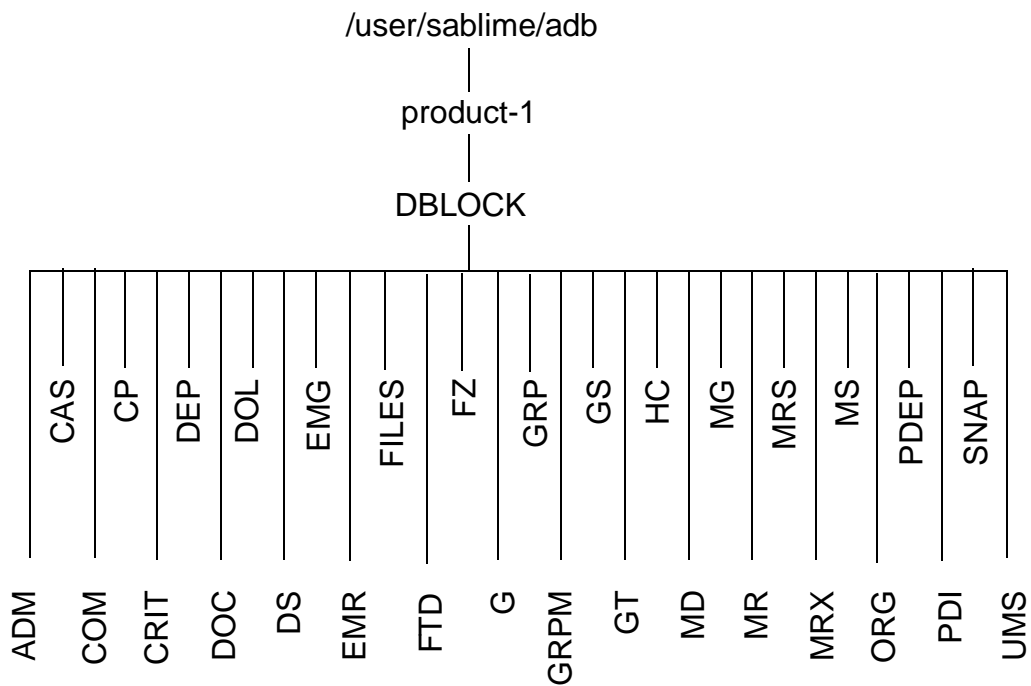


Figure 5-4. Active Database DBLOCK Directory

DBLOCK in the IDB Directory

The DBLOCK directory structure in the IDB is shown in Figure 5-5.

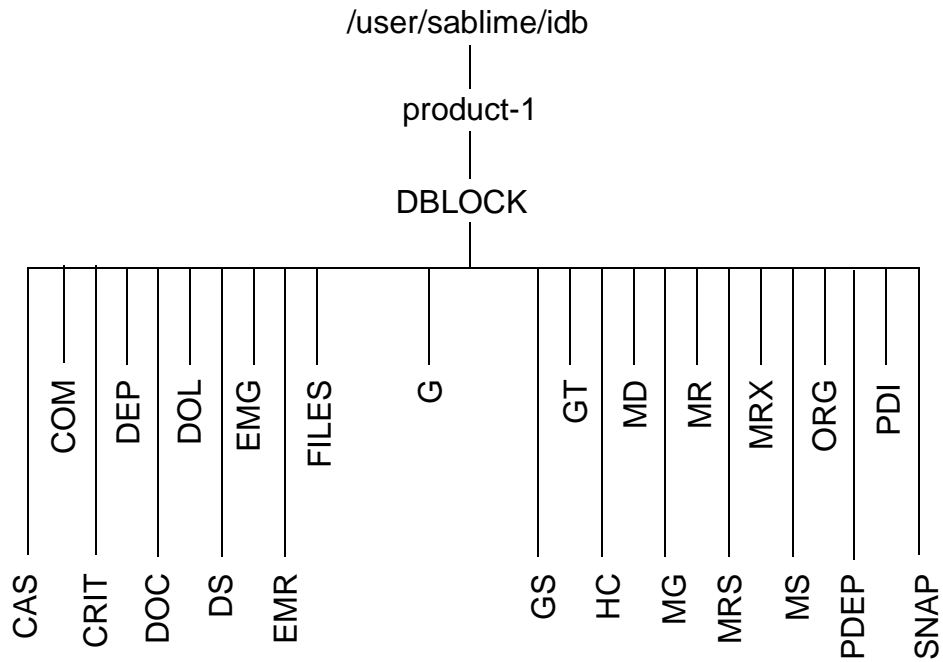


Figure 5-5. Inactive Database DBLOCK Directory

DBLOCK contains a set of directories matching the relations described below. This directory allows a tuple file to be locked when it is accessed or updated by a Sablime command.

FILES Directory

The FILES directory structure for the ADB is shown in Figure 5-6; the FILES directory structure for the IDB is shown in Figure 5-7.

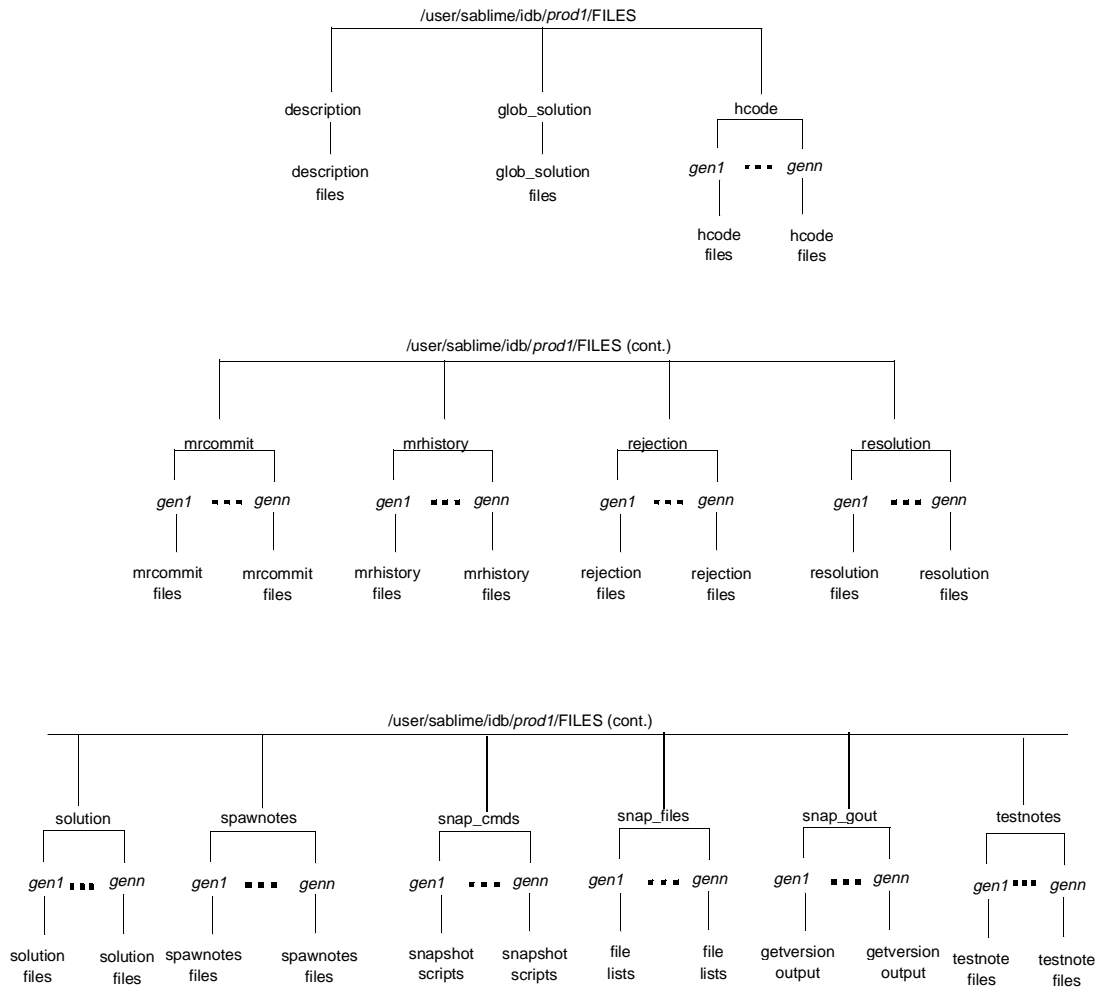


Figure 5-7. Inactive Database FILES Directory

FILES contains:

SAB.stopfile	A file to be used when a Sablime instance is shut down for administrative tasks. This file must always exist. If it has a zero length, the Sablime commands will execute. If the length is greater than zero, most Sablime commands stop execution and the contents of this file is displayed to the Sablime users.
description	Directory containing the MR description files. Each time create, fcreate, or review is processed and an MR is created, the user of the command is expected to enter information about the problem or enhancement for which the MR is being created. A description file with the same name as the MR number is created containing the information entered by the user. Notes can be added to this file using the <code>mrnote</code> command.
glob_solution	Directory containing the global solution files. Each time propose is processed before an MR is accepted in a generic, the developer using the command is expected to describe a proposed solution for an MR in the <i>mra_study</i> state. A global solution file with the same name as the MR is created containing the information entered by the developer. Notes can be added to this file using the <code>mrnote</code> command.
hcode	Directory containing a directory node with the same name as the generic for each active generic in the product. Under each generic node are the <code>hcode</code> description files for that generic. Each time the <code>hcode</code> command is processed, the user of the command is expected to enter information about the code ID that is being added, copied, or modified. The <code>hcode</code> description file has the same name as the code ID.
mrcommit	Directory containing a directory node with the same name as the generic for each active generic in the product. Under each generic node are the MR files for MRs committed for that generic. An MR file with the same name as the commitment ID defined with the <code>commit</code> command is created whenever MRs are committed.
mrhistory	Directory containing a directory node with the same name as the generic for each active generic in the product. Under each generic node are the history files for MRs accepted for that generic.

A history file with the same name as the MR number or MR spawn is created whenever an MR is accepted in a generic. Each time the status of the MR changes, a line is added to the file in chronological order to provide information about which command made the change, what change was made, and when the change occurred. These files are created only if the history flag is on in the ADM relation.

rejection

Directory containing a directory node with the same name as the generic for each active generic in the product. Under each generic node are the rejection files for MRs accepted for that generic.

Each time `reject` is processed, the test team member using the command is expected to state a reason for rejecting the MR. A rejection file with the same name as the MR or MR spawn is created containing the information entered by the test team member. Notes can be added to this file using the `mrnote` command.

resolution

Directory containing a directory node with the same name as the generic for each active generic in the product. Under each generic node are the resolution files for MRs accepted for that generic.

Each time `edput` or `submit` is processed, the user of the command is expected to describe how the MR was resolved. A resolution file with the same name as the MR or MR spawn is created containing the information entered by the developer. Each time `unedput` is processed, the unedputted files are stored in the resolution file. Notes can be added to this file using the `mrnote` command.

Each time `unedput` is processed, the unedputted files are stored in the resolution file.

solution

Directory containing a directory node with the same name as the generic for each active generic in the product. Under each generic node are the solution files for MRs accepted for that generic.

Each time `propose` is processed, the user of the command is expected to describe a proposed solution for an MR in the *understudy* state. A solution file with the same name as the MR or MR spawn is created containing the information entered by the developer. Notes can be added to this file using the `mrnote` command.

spawnnotes

Directory containing a directory node with the same name as

the generic for each active generic in the product. Under each generic node are the spawnnotes files for MRs accepted for that generic.

Each time `spawnmr` is processed, the user of the command is expected to enter information concerning the spawned MRs. A spawnnotes file with the same name as the original MR is created containing the information entered by the user. Notes can be added to this file using the `mrnote` command.

templates

Directory containing directories for the template files for Sablime commands, the SDF (System Development Framework), and Quality Plans, as well as other templates, including a document order form and enhancement request form, mostly in `troff` format. These template files are created when Sablime is installed. They can be customized for your project.

Other template directories and files can be added with the `tempset` and `template` commands.

testnotes

Directory containing a directory node with the same name as the generic for each active generic in the product. Under each generic node are the testnotes files for MRs accepted for that generic.

Each time `testpass` is run, the user of the command may enter comments regarding the testing for an MR in the *submitted* state or a testing state, `preitpassed` to `stpassed`. A testnotes file with the same name as the MR or MR spawn is created containing the information entered by the user. Notes can be added to this file using the `mrnote` command.

trace

Directory containing a file named with each Sablime user's login with a record created each time the user processes a transaction. Each record describes the transaction performed by that user. These records are stored in chronological order.

These files are created only if the trace flag is set to on in the ADM relation when Sablime is installed or through a modification of the ADM relation after installation using the `setrel` command.

Active/Inactive Database Relations

The Active/Inactive Database relations are presented on the following pages in alphabetical order, as shown in Table 5-8. The first column gives the name of the relation, the second shows the type of information stored in the relation, the third contains the name of the hash field for the relation, and the last lists the command or commands that create or update the information in the relation.

Table 5-8. Relations vs. Commands

Name	Information	Hash Field	Command
ADM	Administrative Product	DBA Group Name	create dbedit fcreate initsab setrel
CAS	Cascading Menu	Cascading Type	dbedit initsab setrel
COM	Commitment	Generic	commit closegen closemr dbedit initsab
CP	Command Permissions	Command Name	dbedit initsab setrel
CRIT	Automatic Routing and Automatic Assignment Criteria	Criteria Type	closegen dbedit initsab setrel
DEP	MRG Dependencies	MR Number	closemr dbedit depend edput initsab unedput
EMG	External MRG	External ID	closemr dbedit initsab killmr review

Table 5-8. Relations vs. Commands—Continued

Name	Information	Hash Field	Command
EMR	External and internal MR link	MR Number	closemr dbedit initsab killmr qmr review
ES			dbedit initsab setrel
FTD	Field Tracking Data	Command Name	dbedit ftd initsab
FZ	File to Snapshot		closegen dbedit getversion initsab snapedit source
G	Generic	Generic Name	addgen closegen dbedit initsab
GRP	Group Owner and Type	Group Name	dbedit initsab setgroup
GRPM	Group Members	Group Name	dbedit initsab setgroup
GS	Generic Source File	Source File Name	addsrc addisrc approve closegen common dbedit edget edput initsab reserve source uncommon unedget unreserve

Table 5-8. Relations vs. Commands—Continued

Name	Information	Hash Field	Command
GT	Test Teams Assigned in a Generic	Generic	addgen closegen dbedit initsab
HC	Hardware	Hardware Code ID	closegen dbedit hcode initsab submit
MD	MRG/Source File Deltas	MR Number	addgsr addisr approve closemr dbedit edget edput initsab reserve source unedget unedput unreserve

Table 5-8. Relations vs. Commands—Continued

Name	Information	Hash Field	Command
MG	MRG Information	MR Number	accept activate addgen assign approve closemr commit dbedit defer fcreate initsab mrgedit nochange propose qmr reject review spawnmr study submit testassign testpass unaccept unassign
MR	MR Creation and Completion	MR Number	accept activate closemr create dbedit defer fcreate initsab killmr mredit propose review spawnmr study unaccept

Table 5-8. Relations vs. Commands—Continued

Name	Information	Hash Field	Command
MRS	Spawned MRG	MR Number	accept closemr dbedit initsab spawnmr
MRX	MR status	MR Number	activate closemr dbedit defer initsab killmr propose study
MS	Source File Relationship to MRGs in a Generic	MR Number	addsrc addisrc approve closemr dbedit edget initsab source unedget
ORG	MR or MRG Origination	MR Number	approve closemr create dbedit fcreate initsab killmr mredit review spawnmr
PDEP	Files that cause MR Dependency	MR Number	closegen closemr dbedit depend edput initsab sget source unedput

Table 5-8. Relations vs. Commands—Continued

Name	Information	Hash Field	Command
PDI	Product Design Information	PDI Number	closegen dbedit initsab pdi submit
PR	Product Information	Product Name	dbedit initsab setrel
PRX	Extra Product Information	Product Name	dbedit initsab setrel
PTS	Personnel Information	PTS ID	dbedit initsab pts
SNAP	Snapshot Information	Snapshot ID	closegen closemr dbedit getversion initsab snapedit
TR	Generics for each Product	Generic Name	addngen dbedit initsab
UMS	Source File Relationship to Unapproved MRGs	Source File	addgsrc addisrc approve dbedit edget initsab source unedget unedput

A detailed description including field names, keys, and examples for each of these relations is given on the following pages.

The ADM Relation

The ADM (ADMINistrative) relation contains information related to the overall administration of a product installed under the Sablime instance. The key for this relation is Database Administrator Group Name.

There is one tuple file containing one record. The tuple file contains the product-related information shown in Table 5-9.

Table 5-9. ADM Relation Fields

Field	Information
1	Database Administrator Group Name
2	Modification Request Administrator Group Name
3	MR Suffix (number) to be assigned to the next MR to be created
4	Prefix String for MR numbers in the product
5	Turn on the creation of trace files (y or n)
6	Turn on the creation of history files (y or n)
7	Turn on mail for the Sablime environment (y or n)
8	Internal Field Separator for pop-up selection windows (should never change)
9	In-Process Metrics Flag
10	Not used
11	Automatic Routing Flag
12	Automatic Assignment Flag
13	Reassignment Flag
14	Automatic Dependency Specifier
15	Dependency Override Flag
16	Source Administrator Group Name
17	Mail Dispatch Interval
18	Hardware Administrator Group Name
19	Default version control tool for non-binary files
20	Not used
21	Branch Flags

Example:

```
dba_sab;mra_sab;970034;sab;y;y;y;;y;;y;y;y;none;y;sa;8;;SBCS;;y
```

The CAS Relation

The CAS (Cascading) relation contains information about the fields with menus that depend on data entered in other fields. The keys for this relation are Cascading Type and Upper Level Key.

There is one record for each upper-level key value in a cascade type. Each record contains the information shown in Table 5-10 gathered at installation or with the `setrel` command.

Table 5-10. CAS Relation Fields

Field	Information
1	Cascading Type
2	Upper Level Key
3	Lower Level Group Name

Example:

```
sysCASsub;lib;sablime3.1_libgrp  
subCASmod;lib:libCOM;libcom_grp1
```

The COM Relation

The COM (COMmit) relation contains information about committed MRs. The keys for this relation are the Generic Name and the Commitment ID.

There is one record for each commitment ID in a generic. Each record contains the information shown in Table 5-11 gathered through the commit command.

Table 5-11. COM Relation Fields

Field	Information
1	Generic Name
2	Commitment ID
3	Commitment Date
4	Number of Users Affected
5	Date and Time commitment was made

Example:

```
test2.0;000;09/13/88;5;08/01/88 19:46:09
test2.0;111;09/13/88;3;08/01/88 19:46:09
test2.0;ABC;07/07/88;1;06/20/88 08:22:59
test2.0;LJH;08/08/88;2;06/20/88 16:00:50
test2.0;esz;12/12/88;17;06/20/88 10:21:48
```

The CP Relation

The CP (Command Permissions) relation contains information about who may execute commands. The keys for this relation are the Command Name, the Generic Name, and the Type of Function.

There is at most one record for each command-function combination in a generic. Each record contains the information shown in Table 5-12 gathered through the `setrel` command.

Table 5-12. CP Relation Fields

Field	Information
1	Command Name
2	Generic Name
3	Type of Function
4	Executor List
5	Email Recipient List

Example:

```
assign;test2.0;;__DEFAULT,mkp;__DEFAULT
```

The CRIT Relation

The CRIT (CRITeria) relation contains criteria for the Automatic Routing and the Automatic Assignment capabilities. The keys for this relation are Criteria Type and Criteria Owner.

There is at most one record for each criterion owner for each criterion type. Each record contains the information shown in Table 5-13 gathered with the `setrel` command.

Table 5-13. CRIT Relation Fields

Field	Information
1	Criteria Type
2	Criteria Owner
3	Generic (for Automatic Assignment only)
4	MR Class (for Automatic Assignment only)
5	MR Subclass (for Automatic Assignment only)
6	MR Type (for Automatic Assignment only)
7	MR Subtype (for Automatic Assignment only)
8	System(s) to match
9	Subsystem(s) to match
10	Module(s) to match
11	Site(s) to match
12	Release(s) to match
13	Not used
14	Not used

Example:

```
assign;esz;tst3.1;software;;modification;;admin;;;;;
```

The DEP Relation

The DEP (DEPend) relation contains information about the dependencies established between MRGs. The keys for this relation are the Dependent MR Number, Generic Name, and the Depended-Upon MR Number.

There is one record for each association of dependent MRG and depended-upon MRG in a generic. Each record contains the information shown in Table 5-14 gathered through the `depend` command or the `edput` command if your project has chosen to use the Automatic Dependency option.

Table 5-14. DEP Relation Fields

Field	Information
1	Dependent MR Number
2	Generic Name
3	Depended-Upon MR Number
4	Logical Dependency Flag
5	Physical Dependency Flag
6	Reason for Dependency

Example:

```
sab960127;g2;sab960042;y;y;Same code change
sab960127;g1;sab960044;n;y:auto:file-level
sab960127;g1;sab960045;y;n:auto:line-level
```


The EMG Relation

The EMG (External MR in a Generic) relation contains information about External MRs in a Generic. The keys for this relation are the External ID, the External Generic, and the External Product Name.

There is one record for each external ID in an external generic for an external product. Each record contains the information shown in Table 5-15 gathered through the review command.

Table 5-15. EMG Relation Fields

Field	Information
1	External ID (assigned by external project)
2	External Generic Name
3	External Product Name
4	External Status
5	External Commitment ID
6	External Commitment Date
7	Last Message Type that updated this record
8	Reason
9	Time Stamp

Example:

```
tpr960002;tpr4.2;tpr;submitted;;;3;;06/23/96 15:02:23
tpr960007;tpr4.2;tpr;assigned;;;3;;05/24/96 17:02:29
tpr960011;tpr4.2;tpr;accepted;;;3;;07/19/96 10:44:45
tprtpr960085;tpr4.2;tpr;assigned;;;3;;06/09/96 14:06:01
tpr960086;tpr4.2;tpr;accepted;;;3;;06/09/96 14:06:13
```

The EMR Relation

The EMR (External MR) relation contains information about MRs shared with an External Project. The keys for this relation are the MR Number, the External Project, the External Product Name, and the External MR Number.

There is one record for each association of an MR with an External MR for an External Project/Product. Each record contains the information shown in Table 5-16 gathered through the review command and the qmr command.

Table 5-16. EMR Relation Fields

Field	Information
1	MR Number (internal)
2	External Project Name
3	External Product Name
4	External MR Number
5	Route (in or out, depending on whether the MR was received from or sent to an External Project)
6	Message Originator
7	Status (open, closed, or sent)
8	External MR Reason
9	Time Stamp
10	External MR Originator ID
11	External MR Origination Date
12	External MR Required Date
13	External MR Severity
14	External MR System
15	External MR Subsystem
16	External MR Release
17	External MR Site
18	External MR Category
19	External MR Module
20	External MR Phase Detected
21	External MR User-Definable Field 1
22	External MR User-Definable Field 2
23	External MR User-Definable Field 3

Table 5-16. EMR Relation Fields

Field	Information
24	External MR User-Definable Field 4
25	External MR User-Definable Field 5
26	Not Used

Example:

```
sab1960271;sablime;tst;tst960203;out;anil;open;;11/10/96 \  
11:47:36;anil;11/10/96;;3;xmrmgmt;sub1_xmrmgmt;not_applicable;00;\br/>testing;;unit_test;;;;;
```

The FTD Relation

The FTD (Field Tracking Data) relation contains information about the fields used for entry of data in the Command Line and Curses Forms interfaces. In addition, some of the fields are used by the Graphical User interfaces. The keys for this relation are Command Name and Internal Keyword.

There is one record for each field in a command. Each record contains the information shown in Table 5-17 gathered through the ftd command or at installation.

Table 5-17. FTD Relation Fields

Field	Information
1	Name of command in which field is used
2	Internal Keyword
3	Contains 3 subfields:
i.	Is field displayed (y or n)
ii.	Is field mandatory (y or n)
iii.	Is field hideable (y or n)
4	Contains four subfields that relate to the Command Line interface:
i.	Command Line help text
ii.	Not Used
iii.	Not Used
iv.	Verbose help message number
5	Contains three subfields:
i.	Code for type of field entry (i.e., 1=string, 2 =Pop-Up Selection Window that loads from group, 3=Pop-Up Selection Window that loads dynamically)
ii.	Maximum field length
iii.	Default value or Sablime reserved keyword
6	Group Name for a Type 2 Field
7	Contains eight subfields that relate to the Curses Forms interface:
i.	Starting row number of the field
ii.	Starting column number of the field
iii.	Fill character
iv.	Left/right scrolling buffer length
v.	Prompt position with respect to the field
vi.	Text for field name or prompt

Table 5-17. FTD Relation Fields—Continued

Field	Information
vii.	Attribute (Must be 0)
viii.	Number of menu choices allowed
8	User-Defined External Keyword

Example:

```
create;abst;y,y,n;Abstract of Request:,22,,2111;1,57,;;\  
17,22,_,60,left,Abstract of Request: ,0,0;abst
```

The FZ Relation

The FZ (File to Zsnapshot) relation tracks the occurrence of source files in configuration snapshots. Each time a configuration snapshot is created in a generic, an FZ record is created for each source file used in the snapshot. The first four fields are the key fields. The last field contains the identifier of the highest MR-branch delta for that file that is used in the snapshot. This field may be empty, if the snapshot uses no deltas from the generic's MR branch for the file in question.

Table 5-18. FZ Relation Fields

Field	Information
1	File name
2	Directory
3	Generic
4	Snapshot ID
5	SID

Example:

```
foo.c;src/lib/widgets;g1.0;scoobydoo;4.1.2.9
```

The G Relation

The G (Generic) relation contains information about each generic for the product. The key for this relation is Generic Name.

There is one record for each Generic Name. Each record contains the information shown in Table 5-18 gathered through the addgen command.

Table 5-19. G Relation Fields

Field	Information
1	Generic Name
2	Active or Inactive Status
3	Generic Source Database Identifier (GID)
4	Generic Administrator Group Name
5	Documentation Class Flag (y or n)
6	Firmware Class Flag (y or n)
7	Hardware Class Flag (y or n)
8	Software Class Flag (y or n)
9	Generic Creation time stamp
10	Generic Creator's PTS ID
11	Generic Closure time stamp
12	Generic Closer's PTS ID
13	Generic Release Flag (y or n)
14	Not Used
15	Not Used
16	Not Used

Example:

```
g1;active;1.1;ga;y;y;y;y;10/16/96 16:39:23;slc;;;y;;;
```

The GRP Relation

The GRP (GRouP) relation contains information about each group established in a Sablime database. The key for this relation is Group Name.

There is one record for each Group Name. Each record contains the information shown in Table 5-19 gathered through the `setgroup` command.

Table 5-20. GRP Relation Fields

Field	Information
1	Group Name
2	Owner of Group (creator)
3	Type of Group (ptsid, mr, other)

Example:

```
ittgrp;sdp;ptsid
```


The GRPM Relation

The GRPM (GRouP Members) relation contains information about the members of each group established in a Sablime database. The keys for this relation are Group Name and Group Member.

There is one record for each Group Member of a group. Each record contains the information shown in Table 5-20 gathered through the `setgroup` command.

Table 5-21. GRPM Relation Fields

Field	Information
1	Group Name
2	Group Member

Example:

```
ittgrp;cdf  
ittgrp;lsc
```

The GS Relation

The GS (Generic/Source file) relation contains information about the source files that have been added for a generic. The keys for this relation are Source File Name, Relative Directory Name, and Generic Name.

There is one record for each Source File Name associated with a relative directory in a generic. Each record contains the information shown in Table 5-21 gathered through the `addsrc` or the `addgsrc` command.

Table 5-22. GS Relation Fields

Field	Information
1	Source File Name
2	Logical Relative Directory Name with which the File is Associated
3	Generic Name
4	Physical Relative Directory Name in SDB where file is stored
5	Date and Time (space separated) the source file was last touched
6	Status of Source File (free or busy)
7	Generics (space separated) in which file is common
8	Binary File Flag—y for binary, n for non-binary
9	Language of Source File
10	Source Files Counted for Software Quality Assurance
11	Source File Owner
12	Version Control Tool Used for File (SBCS or SCCS)
13	Latest mr Branch Version Number in the SBCS File that is declared official
14	Not Used
15	Reserved MR Number (MR Number of the reserved check-out - if any)
16	Check Out Count

Example:

```
create.c;src/mrmgmt;v4.2;src/mrmgmt;09/18/96 11:03:07;free;;n;\
c++;y;;SCCS;;;0
lst_prod.c;src/lib/libPOST;v3.0;src/lib/libPOST;08/28/89 \
13:08:57;free;v3.0 v3.1 v5.0;n;c++;y;;SCCS;;;0
lst_prod.c;libPOST;v3.1;src/lib/libPOST;\
05/28/90 06:42:50;free;v3.0 v3.1 v5.0;n;c++;y;;SCCS;;;0
create.c;src/mrmgmt;v3.1;src/mrmgmt;02/21/96 \
17:41:14;busy;v3.1 v5.0;n;c++;y;;SCCS;;;sab001101;1
create.c;src/mrmgmt;1bproc;src/mrmgmt;05/02/91 \
16:52:59;free;;n;c++;y;;SCCS;;;0
```

The GT Relation

The GT (Generic/Test team) relation contains information about the Test Teams and States to be used in a generic. The keys for this relation are Generic Name and Class.

There is one record for each Class selected for a Generic. Each record contains the information shown in Table 5-22 gathered through the `addgen` command.

Table 5-23. GT Relation Fields

Field	Information
1	Generic Name
2	Class
3	Group Name of Test Team 1
4	Group Name of Test Team 2
5	Group Name of Test Team 3
6	Group Name of Test Team 4
7	Group Name of Test Team 5
8	Group Name of MR Approval Team
9	Not used
10	Group Name of Quality Assurance Team
11	Not used

Example:

```
test1.2;software;;;;;sat1.2;;qat;
test1.2;document;dpit_1.2;dit_1.2;dpst_1.2;dst_1.2;dpst_1.2;\
dat_1.2;;;
test1.2;firmware;fpit_1.2;fit_1.2;fpst_1.2;fst_1.2;fpst_1.2;\
fat_1.2;;;
test1.2;hardware;hpit_1.2;hit_1.2;hpst_1.2;hst_1.2;hpat_1.2;\
hat_1.2;;;
```

The HC Relation

The HC (Hardware Code) relation contains information about the hardware codes to be used for hardware quality assurance in a generic. The keys for this relation are Hardware Code Identifier and Generic/Product Release Number.

There is one record for each hardware/release configuration. Each record contains the information shown in Table 5-23 gathered through the hcode command.

Table 5-24. HC Relation Fields

Field	Information
1	Hardware Code Identifier
2	Generic/Product Release Number
3	Version Number of Hardware Code
4	Hardware Code Tuple File Status
5	Quantity of this Hardware Code in a fully equipped system
6	Flag: code is new or reused for this release
7	hcode Complexity
8	Flag: is this hcode used to calculate hardware fault density?
9	Flag: should MRs against this hcode be used to calculate hardware fault density?
10	Date this hcode first went to system test
11	Date this hcode was first released
12	Description of hcode
13	hcode Time Stamp Creation Date
14	hcode Time Stamp Change Date
15	hcode User-Definable Field 1
16	hcode User-Definable Field 2
17	Not Used
18	Not Used

Example:

```
120;2st3.1;;not_mr_linked;12;y;1.0;y;y;;;this is the abstract;\
05/18/96 10:30:41;05/18/96 10:32:25;;;;
```

The MD Relation

The MD (Modification request/Delta) relation contains information about source file changes in response to an MR. The keys for this relation are MR Number, Generic Name, Source File, Relative Directory Name, and Delta Identifier.

There is one record for each delta created in a source file.

When an edget command is processed for the source file, Field 7 is set to either the *reserved* or *unreserved*.

If an unedget or unedput command is issued, the record is deleted from the relation. If an edput command is issued, *reserved/unreserved* is changed to *delta* in Field 7.

Each record contains the information shown in Table 5-24 gathered through the addsrc, addisrc, edget, edput, reserve, or unreserve command.

Table 5-25. MD Relation Fields

Field	Information
1	MR Number
2	Generic Name
3	Source File
4	Logical Relative Directory
5	Delta Identifier of the actual or probable delta
6	Developer who created the delta
7	Status of Delta (i.e., is the file reserved or has a delta been created)
8	Date and Time of the last change in state
9	SID of parent delta

Example:

```
sab960032;g1;create.c;src/mrimgmt;1.1.2.1;lsp;delta;\
11/22/96 15:32:42;
sab960032;g1;create.c;src/mrimgmt;1.1.2.2;lsp;unreserved;\
11/24/96 09:43:18;1.1.2.1
sab960032;g2;create.c;src/mrimgmt;2.1.2.1;lsp;delta;\
11/29/96 17:06:35;
```

The MG Relation

The MG (Modification request/Generic) relation contains information about MRs accepted for a Generic. These MRs are referred to as MRGs (MRs in a Generic). The keys for this relation are MR Number and Generic Name.

There is one record for each MRG. The record is updated each time a command that changes the state of the MRG is processed. Each record contains the information shown in Table 5-25.

Table 5-26. MG Relation Fields

Field	Information
1	MR Number
2	Generic Name
3	MRG State (see Table 5-27)
4	Date and Time of Last State Change
5	Assigned Developer or group name of developers
6	Current Severity of MRG
7	Date MRG is currently due
8	Reason Code for State Change
9	Reason for Change if Code is other
10	System Code for Current MRG Status (see Table 5-27)
11	MRG Class (document, firmware, hardware, or software)
12	MRG Type (enhancement, initialization, or modification)
13	Number of Spawns for the MRG
14	Commitment ID
15	External MR Flag
16	Hardware Code Number
17	Hardware PDI Number
18	MR Subclass
19	MR Subtype
20	Release in which the problem was introduced
21	Phase in which the problem was introduced
22	Optimal Detection Phase
23	Root Cause
24	Root Cause Subcategory
25	Actual Effort
26	Estimated Effort

Table 5-26. MG Relation Fields—Continued

Field	Information
27	Test Team 1 Actual Effort
28	Test Team 2 Actual Effort
29	Test Team 3 Actual Effort
30	Test Team 4 Actual Effort
31	Test Team 5 Actual Effort
32	MRG User-Definable Field 1
33	MRG User-Definable Field 2
34	MRG User-Definable Field 3
35	MRG User-Definable Field 4
36	MRG User-Definable Field 5
37	Actual Study Time
38	Nochange Date
39	Date MR was activated from <i>nochange</i> state
40	Date MR was deferred from <i>accepted</i> state
41	Date MR was activated from <i>deferred</i> state
42	Date MR was moved to <i>understudy</i> state from <i>accepted</i> state
43	Date MR was proposed from <i>understudy</i> state
44	Date MR was accepted
45	Date MR was assigned
46	Date MR was submitted
47	Date MR was stat 1 (preitpassed, prefitpassed, prehitpassed, or preinspected)
48	Date MR was stat 2 (itpassed, fitpassed, hitpassed, or inspected)
49	Date MR was stat 3 (prestpassed, prefstpassed, prehstpassed, or republished)
50	Date MR was stat 4 (stpassed, fstpassed, hstpassed, or published)
51	Date MR was stat 5 (preapproved)
52	Date MR was approved
53	Fault Type
54	Non-Detection Cause
55	Non-Detection Cause Subcategory
56	Cost of Problem
57	Duplicate MR from nochange

Table 5-26. MG Relation Fields—Continued

Field	Information
58	Not Used
59	Not Used
60	Not Used
61	PTS ID or group name of Assigned Tester(s) of stat 1
62	PTS ID or group name of Assigned Tester(s) of stat 2
63	PTS ID or group name of Assigned Tester(s) of stat 3
64	PTS ID or group name of Assigned Tester(s) of stat 4
65	PTS ID or group name of Assigned Tester(s) of stat 5
66	Assigned Tester(s) date of stat 1
67	Assigned Tester(s) date of stat 2
68	Assigned Tester(s) date of stat 3
69	Assigned Tester(s) date of stat 4
70	Assigned Tester(s) date of stat 5

Table 5-27. MRG States and Corresponding System Codes

MRG State (field 3)	System Code (field 10)
<i>nochange</i>	1
<i>deferred</i>	6
<i>understudy</i>	8
<i>accepted</i>	10
<i>spawned</i>	15
<i>assigned</i>	20
<i>submitted</i>	30
Stat1	45
Stat2	50
Stat3	55
Stat4	60
Stat5	70
<i>approved</i>	80
<i>closed (after nochange)</i>	89
<i>closed (after approved)</i>	99

The MR Relation

The MR (Modification Request) relation contains information about MR status. The record is created with the `create`, `fcreate`, or `review` command and is updated each time a `killmr`, `accept`, `unaccept`, `mredit`, `closemr`, `study`, `propose`, `spawnmr`, `defer`, or `activate` command is processed. The key for this relation is MR Number.

There is one record for each MR created. Each record contains the information shown in Table 5-27.

Table 5-28. MR Relation Fields

Field	Information
1	MR Number
2	Creator PTS ID
3	MR State
4	MR Category
5	Abstract entered at time of creation
6	Date MR is required by the creator/originator
7	Requested Severity of MR
8	Total Number of Spawns for MR in All Generics
9	Reason if Reason Code is <i>other</i>
10	Reason Code
11	Date and Time Created
12	Date and Time MR was killed or completed
13	PTS ID of person who executed <code>killmr</code> or <code>closemr</code> command
14	Not used
15	Phase in which problem was detected
16	MR User-Definable Field 1
17	MR User-Definable Field 2
18	MR User-Definable Field 3
19	MR User-Definable Field 4
20	MR User-Definable Field 5
21	Duplicate MR from <code>killmr</code>

Example:

```
tst960861;sablme;created;field_mod;complete format of listmsgs \
does not align;;3;0;;;02/02/96 18:19:14;;; \
controlled_intro;mmist;;;
```

The MRS Relation

The MRS (Modification Request/Spawns) relation contains information about spawned MRGs. The keys for this relation are MR Number, Generic Name, and Spawned MR Number.

There is one record for each spawned MRG. Each record contains the information shown in Table 5-28 gathered through the `spawnmr` command.

Table 5-29. MRS Relation Fields

Field	Information
1	MR Number
2	Generic Name
3	Spawned MR Number

Example:

```
test330130;ljh1;test330130.00
test330130;ljh1;test330130.01
test330130;ljh1;test330130.02
test330130;ljh1;test330130.03
test330123;test1.1;test330123.00
test330010;test1.1;test330010.00
```

The MRX Relation

The MRX (Modification Request/eXtra) relation contains information about MRs that have been deferred or put under study before being accepted in a generic. The key for this relation is MR Number.

There is one record for each deferred or studied MR. Each record contains the information shown in Table 5-29 gathered through the defer command or the study command issued before the MR has been accepted.

Table 5-30. MRX Relation Fields

Field	Information
1	MR Number
2	Activate Date from defer
3	Reason Code from defer
4	Reason if Reason Code is <i>other</i>
5	Assigned Developer for study
6	Severity from study
7	Due Date from study
8	Actual Study Effort for In-Process Metrics in staff days
9	Estimated Effort for In-Process Metrics in staff days

Example:

```
tst960519;08/31/96;enhancement;;;5.0;6.0
tst960134;12/01/96;other;testing;;;1.1;2.2
```

The MS Relation

The MS (Modification request/Source file) relation contains information about source files associated with MRGs. The keys for this relation are MR Number, Generic Name, Source File, and Relative Directory Name.

There is one record for each source file associated with an MRG. Each record contains the information shown in Table 5-30 gathered through the `addsrc`, `addsrc`, `approve`, or `edget` command and updated by the `approve`, `closemr`, `source`, and `unedget` commands.

Table 5-31. MS Relation Fields

Field	Information
1	MR Number
2	Generic Name
3	Source File
4	Logical Relative Directory Name
5	Status of MRG (approved or unapproved)

Example:

```
sab960043;g1;create.c;src/mrmgmt;approved  
sab960043;g2;create.c;src/mrmgmt;approved  
sab960043;g1;defer.c;src/mrmgmt;approved
```

The ORG Relation

The ORG (ORiGination) relation contains origination information about MRs. The key for this relation is MR Number.

There is one record for each MR. Each record contains the information shown in Table 5-31 gathered through the create, fcreate, or review command.

Table 5-32. ORG Relation Fields

Field	Information
1	MR Number
2	MR Originator's PTS ID
3	MR Origination Date
4	Product Name with which MR is associated
5	System Name with which MR is associated
6	Subsystem Name with which MR is associated
7	Release in which MR was detected
8	Site Code where the MR originated
9	Module

Example:

```
sab960252;lam;11/28/96;sab;admin;setrel;1.2;LC;
```

The PDEP Relation

The PDEP (Physical Dependency) relation contains the source files which cause the dependencies created in the DEP relation. Each time `edput` is processed, dependencies are established between the MR number which made the change to a file and other MRs which touched the same file. The information about MR dependencies is stored in the DEP relation. The source files which caused the dependencies are stored in the tuples under the PDEP relation. Each tuple name is the result of hashing the MR number. Each time `unedput` or `depend` (to remove dependency) is processed, the related information in this relation will be removed.

Table 5-33. PDEP Relation Fields

Field	Information
1	MR Number
2	Generic
3	Dependent MR
4	Filename
5	Logical Relative Directory of field 4
6	File's SCCS/SBCS SID
7	Reason for automatic dependency

Example:

```
sab970127;g1;sab970044;jk42.c;src;1.1.2.13;auto:INITfile-level
sab970127;g2;sab970042;jk42.c;src;2.1.2.2;auto:SBCSfile-level
sab970012;g1;sab970023;my.c;src/admin;1.1.2.5;auto:line-level
```

The PDI Relation

The PDI (Product Design Information) relation contains product design information about hardware. The keys for this relation are PDI Number and Release.

There is one record for each PDI Number/Release combination. Each record contains the information shown in Table 5-32 gathered through the pdi command.

Table 5-34. PDI Relation Fields

Field	Information
1	PDI Number
2	Release
3	PDI Tuple File Status
4	Hardware Change Classification
5	Date Hardware Change was issued
6	Other Drawings or Documents affected by this PDI
7	Reason for Change
8	Cost of Change 1
9	Cost of Change 2
10	Cost of Change 3
11	Cost of Change 4
12	Cost of Change 5
13	Cost of Change 6
14	Cost of Change 7
15	Cost of Change 8
16	Cost of Change 9
17	Cost of Change 10
18	Cost of Change 11
19	Cost of Change 12
20	Total Cost of Change
21	PDI Tuple File Creation Date
22	PDI Tuple File Change Date
23	PDI User-Definable Field 1
24	PDI User-Definable Field 2
25	Not Used
26	Not Used

Example:

```
786;1st3.1;linked_unavail;A;01/01/96;;;;;;;;;;;;;\
0;06/18/96 15:17:15;11/30/96 15:07:23;;;;
```

The SNAP Relation

The SNAP (Snapshot) relation contains information about configuration snapshots, which are made with, and used by, the `getversion` command. The keys for this relation are Snapshot ID and Generic. Fields 6 through 13 are taken directly from the `getversion` call that created the snapshot. The only field that can be updated is field 5.

Table 5-35. SNAP Relation Fields

Field	Information
1	Snapshot ID
2	Generic
3	Creator
4	Creation Date
5	Comments
6	MRG States
7	Branch
8	Include Depended-Upon MRs
9	MRs for File Selection
10	MRs for Additional Changes
11	Source Directories to restrict to
12	Cutoff Date
13	Keyword Expansion flag

Example:

```
L2;g1.0;joe;06/02/97 17:54:21;Load 2\
build;submitted,stopped;ofc;no;;;y
```

The UMS Relation

The UMS (Unapproved Modification request/Source file) relation contains information about source files associated with unapproved MRGs. The keys for this relation are Source File, Relative Directory, Generic Name, and MR Number.

There is one record for each source file associated with an unapproved MRG. Each record contains the information shown in Table 5-33, gathered through the `addsrc`, `addgsrc`, `edget`, or `approve` command.

Table 5-36. UMS Relation Fields

Field	Information
1	Source File
2	Logical Relative Directory Name
3	Generic Name
4	MR Number

Example:

```
defer.c;src/mrmgmt;g3;sab960241  
defer.c;src/mrmgmt;g2;sab960352  
defer.c;src/mrmgmt;g2;sab960417
```

Source Database

All the source files for each product installed under a Sablime instance are stored in the Source Database. When `initsab` is run, a directory structure parallel to the product's directory structure is created under the `sdb/product_name` node (see Figure 5-8).

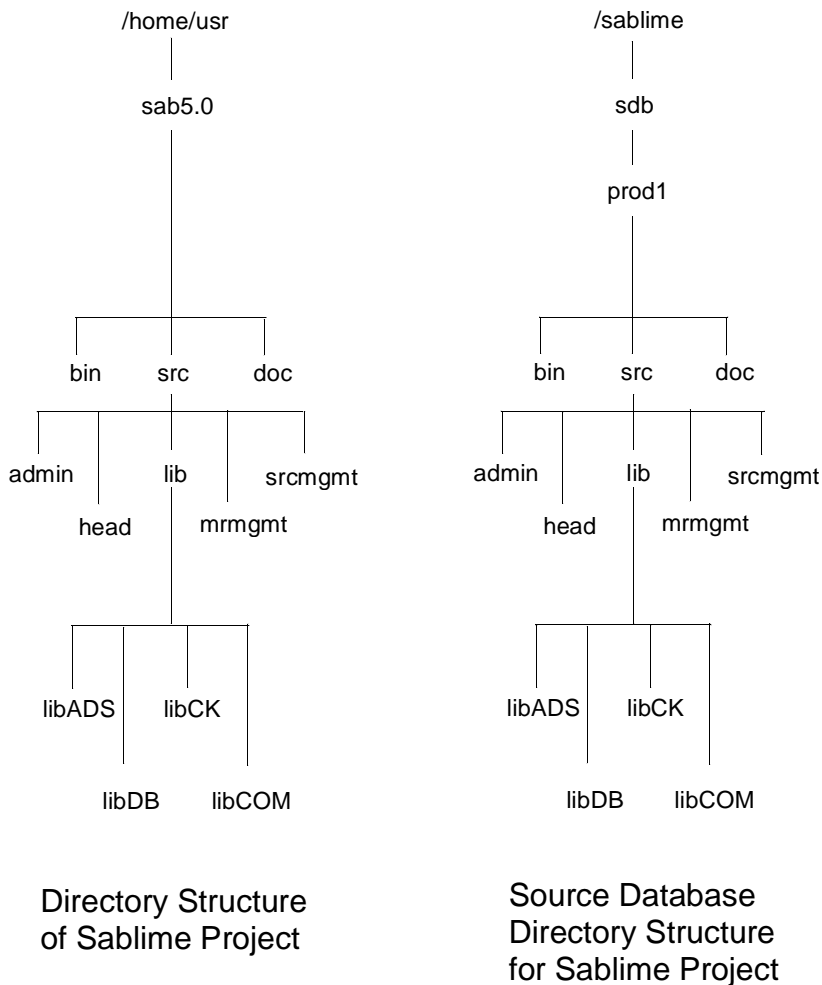


Figure 5-8. Directory Structure for the Sablime Instance

As more generics are added for the product, the corresponding directory structures are added under the `sdb/product_name` node. Thus, the Source Database represents the union of all directory structures for all generics for the product. Regardless of the number of generics defined for a product, Sablime keeps only one copy of a version-controlled source file for each unique file.

Delta Numbers

When a generic is created for a product, the release and level of the GID (Generic Identifier) for the Source Database is defined; it is called the generic ID number. A delta number is the combination of three separate elements (the generic ID number, the branch number, and the sequence number) separated from each other by a dot. An example is shown in Figure 5-9.

Generic ID Number	Branch Number 1=ofc, 2=mr	Sequence Number
4.1	1	2

Figure 5-9. Delta Specifiers

The first box in Figure 5-9 shows the generic ID for the Sablime generic. This number is incremented by one for each new generic in your product and has no relation to your internal generic specification. For example, your first generic is `sab1.0` and is given the generic ID 1.1. Your second generic is `sab1.1`, and it is given the generic ID 2.1, etc.

⇒ NOTE:
The generic ID number is never reused within a product even after a generic has been closed. In the generic ID number, the digit after the dot (.) is always 1.

The second box in Figure 5-9 shows the branch number. The `ofc` branch is indicated by the number 1. The `mr` branch is indicated by the number 2.

The third box in Figure 5-9 shows the sequence number. This number is incremented by 1 whenever a delta (change) is made to a file in the specified branch.

Contents

6	The Administrative Commands	1
n	Overview	1
n	Command Descriptions	3
n	Host-Only Commands	4
n	addgen	5
n	closegen	10
n	dbcross	12
n	dbdelta	13
n	dbedit	14
n	dbhash	16
n	dbstart	17
n	dbstop	18
n	dbxcross	20
n	ftd	21
n	hotline.ck	26
n	initsab	28
n	mhst	29
n	mmail	31
n	mredit	32
n	mrgedit	38
n	mtrace	45
n	mvgen	47
n	newgen	48
n	primfdb	49
n	screate	50
n	setperm	54
n	setrel	55
n	source	71
n	spacecheck	74
n	sreport	75

Contents

The Administrative Commands

6

Overview

This chapter contains manual pages for administrative commands. Some of these that can only be run by members of one of the four administrative groups - Database Administrator (DBA), Source Administrator (SA), Modification Request Administrator (MRA), and Generic Administrator (GA) - or by someone who has been given permission to run them by the command permissions feature. Some of these can only be run by the *sablime* login. A brief description of each command appears in the following table:

Table 6-1. The Sablime Administrative Commands

Command	Description
addgen	The <i>addgen</i> command allows the <i>sablime</i> login to add, modify, or view information about a generic.
closegen	The <i>closegen</i> command allows the <i>sablime</i> login to move information about a generic from the Active to the Inactive Database.
dbcross	This program does a cross-check of basic Active and Inactive Database relations by comparing the contents of records to the contents of other records.
dbdelta	This program compares relations containing information about files in the Active Database with the information in the Source Database where the files actually reside. If you do not use Sablime for file control, you do not need to run this audit program.
dbedit	The <i>dbedit</i> command allows the DBA to correct or modify the Global, Active, or Inactive Database.

Table 6-1. The Sablime Administrative Commands

Command	Description
dbhash	The dbhash command allows anyone to obtain the two-letter file name assigned to a tuple file in the Sablime database relation. This file name can then be used to access a desired record in the database. This command can be used by any Sablime user.
dbstop/dbstart	The dbstop command allows the DBA or SA to close down a set of Sablime databases to users and issue a message to inform them of this event. The DBA/SA can then take any necessary action. The DBA/SA issues the dbstart command to resume database processing.
dbxcross	This program does a cross-check of all the database relations for the Document Management, External Communications, Hardware Quality Assurance, and Software Quality Assurance features by comparing the contents of records to the contents of other records. If you are not using any of these features, you do not need to run this audit program.
ftd	The ftd command allows the DBA to modify or any user to view fields in Sablime commands.
hotline.ck	Run this program when the Sablime commands are not running or are generating system error messages. The program checks for permissions of commands, database directories, files, and several other items associated with Sablime.
initsab	The initsab script allows the <i>sablime</i> login to establish a new Sablime instance with its first product or add a new product to an existing instance.
mhist/nohist	The mhist/nohist commands allow the DBA to turn history file data collection on and off for the product. These commands are entered at the system prompt to toggle the flag set in the initsab program. The DBA can issue them at any time.
mmail/nomail	The mmail/nomail commands allow the DBA to turn mail on and off for the product. These commands are entered at the system prompt to toggle the flag set in the initsab program. The DBA can issue them at any time.
mredit	The MRA may want to change the data in database relations containing MR information. The mredit command allows the MRA to modify any information entered when the MR was created including the MR relation, ORG relation, or the Description File for a particular MR.
mrgedit	The GA may want to change the data in database relations containing MRG information. The mrgedit command allows the GA to modify the MG relation or the Rejection File, the Resolution File, the Solution File, or the Spawn Notes File for a particular MRG.
mtrace/notrace	The mtrace/notrace commands allow the DBA to turn the trace function on and off for the product. These commands are entered at the system prompt to toggle the flag set in the initsab program. The DBA can issue them at any time.

Table 6-1. The Sablime Administrative Commands

Command	Description
mvgen	The mvgen script allows convenient renaming of an existing generic. Only the <i>sablime</i> login can use mvgen.
newgen	The newgen script provides an automated way of adding a new generic to an existing project. It calls the addgen and setgroup commands and puts you in your favorite editor to update the xsablime.sh script. Only the <i>sablime</i> login can use newgen.
primsdb	The primsdb command provides a way for the DBA to add a large number of files to the Sablime Source Database.
screate	The screate command allows a Sablime customer DBA to enter a modification or enhancement request to the Sablime development team directly.
setperm	The setperm command allows the <i>sablime</i> login to reset the permissions of all the Sablime commands if they have been corrupted.
setrel	The setrel command allows the <i>sablime</i> login to work with the ADM relation; the DBA to work with the CAS, ES, PR, and PRX relations; the MRA, GA, and all users can work with the CRIT relation to route, assign, or view information.
source	The source command allows the SA to rename, move, or delete files from the Sablime Source Database (SDB) or to modify the specified file type (language) or file owner for any file in the SDB.
spacecheck	The spacecheck script checks the available space on the filesystems where the databases are stored.
sreport	The sreport command allows a Sablime customer DBA to produce a report about the state of all Sablime MRs that affect customers.

Command Descriptions

The manual pages for the administrative commands contain some or all the following parts:

- n a brief summary of what the command does
- n a synopsis, listing the keywords and options that may be used with the command, if any
- n a description of the command, which either provides detailed information about how the command functions, or refers the reader to the place in the documentation where more information is available..
- n a table listing the field names that appear in the Curses Forms interface, the keywords that may be used in the Command Line interface, and the values associated with each.

n a list of related commands.

Host-Only Commands

When you use Sablime in multi-machine mode, commands in the following table that are marked with a pound sign (#) can be run only on the host machine if the network type is TCP/IP. Commands marked with a double pound sign (##) are available on the host only, regardless of the network type..

Table 6-2. Command Availability

addgen#	dbxcross##	mvgen##
closegen#	ftd#	nohist
dbcross##	hotline.ck	nomail
dbdelta##	initsab##	notrace
dbedit#	mhist	primsdb#
dbstart#	mmail	setrel#
dbstop#	mtrace	source#

addgen Command

The `addgen` command is used by the *sablime* login to add, modify, or view information about a generic in the Sablime database.

**NOTE:**

If you are simply adding a new generic to a product, you can use the `newgen` command, which automates much of the process.

Synopsis

```
addgen [-help | -?]
```

Description

If you plan to build a particular version or release of your product, send it to one or more customers, and maintain it, you should define that version as a generic in Sablime.

**NOTE:**

Make sure that you are set up for the product for which you intend to add a new generic before running the `addgen` command.

When a new generic is added, the generic name is mapped to an internal generic ID (GID). The GID is a number of the form *n.m* where *m* is always 1 and *n* is incremented by one every time a new generic is added in your product.

**NOTE:**

The GID numbering system is independent of your generic naming convention. The GID is never reused within a product even after a generic has been closed.

For the *Accept MRs with States* field, the MRs in question are from the generic with the highest GID. Verify which generic has the highest GID by examining the tuples in the G relation for your product. Issue the following command to print all records from the G relation:

```
query relation=G prompt=n
```

Output looks like this:

```
v4.0;active;5.1;ga_v4.0;y;n;y;01/07/93 10:06:19;sablime;;;n;;;  
v4.2;active;6.1;ga_v4.2;y;n;y;04/05/96 16:28:57;sablime;;;n;;;  
v5.0;active;4.1;ga_v5.0;y;n;y;05/29/92 10:29:21;sablime;;;n;;;
```

The GID is the third field (fields are separated by semicolons). In this case, the highest GID (6.1) belongs to generic v4.2, not v5.0.

Up to five screens (an initial screen and a screen for each of the MR classes you choose to use in the generic) are used by the `addgen` command to establish the new generic with Generic Administrators and to set up the MR classes and test and approval teams and members to be used when working with MRs in the generic. The fields in three of the screens (the initial screen and two of the MR class screens) are described below.



NOTE:

Before you issue the `addgen` command to add a new generic, be sure to use the `setgroup` command to create a group for Generic Administrator and for each test team to be included in the generic. An Approval Team group is mandatory.

Generic names must be unique within an instance of Sablime (i.e., across all the products). Unique names allow Sablime to gather information about the product for which a user is set up at the same time as it gathers generic information.

If you decide to change the name of an existing generic, use the `mvgen` script.

If the name of an existing generic is entered in the *Name for New Generic* field, all existing field values are displayed on the screen. These values can be changed or deleted, or new values can be added. The database is modified accordingly.

This command can be used in single-machine mode or from the host machine in multi-machine mode.

Used By


`sablime login`


Sends Mail To


DBA, GA

Field Descriptions

The fields on the first addgen screen, and appropriate values for them, are explained below.

Fields	Values
<i>Generic Name</i>	Enter the name for the generic. If the generic already exists, the rest of the screen is populated with existing data and you are allowed to modify the entries. Generic names must be unique within a Sablime instance. The maximum length of the name is 14 characters; it can include dots (.).
<i>Generic Administrator</i>	Enter the name of the group identifying the GAs for this generic. (See the setgroup command in the <i>User's Reference Manual</i> for information about establishing groups.)
<i>Accept MRs with State(s)</i>	<p>Enter a menu item to indicate which active MRs (those that have been accepted into the old generic and not closed) should be included in the generic. You can specify a series of one or more states, all or none.</p> <ul style="list-style-type: none"> n If one or more states are specified, all active MRs from the previous generic (the one with the highest generic ID number) with those states are accepted in the new generic. n If all is selected, all active MRs from the previous generic are accepted in the new generic. n If none is selected, no MRs are accepted in the new generic. <p>Default: all</p> <p> NOTE: When you modify an existing generic, this field is protected and an entry of no_more is made automatically.</p>

 **NOTE:**
At least one MR class from the following four fields must be selected for each generic.

 **CAUTION:**
Do not change a **y** entry to **n** unless you want to delete that MR class from the database for the generic.

Fields	Values
<i>Are you going to have Document Class MRs</i>	Enter y or n to specify whether document class MRs will be used in this generic. If y is specified, MRs with a class of <i>document</i> can be accepted for this generic. Default: n .
<i>Are you going to have Firmware Class MRs</i>	Enter y or n to specify whether firmware class MRs will be used in this generic. If y is specified, MRs with a class of <i>firmware</i> can be accepted for this generic. Default: n
<i>Are you going to have Hardware Class MRs</i>	Enter y or n to specify whether hardware class MRs will be used in this generic. If y is specified, MRs with a class of <i>hardware</i> can be accepted for this generic. Default: n
<i>Are you going to have Software Class MRs</i>	Enter y or n to specify whether software class MRs will be used in this generic. If y is specified, MRs with a class of <i>software</i> can be accepted for this generic. Default: n
<i>Copy To</i>	Enter the PTS ID, email address, group name, or comma-separated list of PTS IDs, email addresses, or group names of people to be sent copies of mail. The mail sent as a result of confirming this command is sent to each person included in the Copy To list. This line scrolls to accept an entry of up to 256 characters.

The fields on the Software and Document Teams and States screens, and appropriate values for them, are explained below. Where they differ, the document field names are shown below the software field names.

Fields	Values
<i>Pre-Integration Test Team</i> <i>Pre-Inspection Team</i>	Enter a valid group name if this state is desired in the generic. Entry in this field is optional.
<i>Integration Test Team</i> <i>Inspection Team</i>	Enter a valid group name if this state is desired in the generic. Entry in this field is optional.
<i>Pre-System Test Team</i> <i>Pre-Publishing Team</i>	Enter a valid group name if this state is desired in the generic. Entry in this field is optional.
<i>System Test Team</i> <i>Publishing Team</i>	Enter a valid group name if this state is desired in the generic. Entry in this field is optional.
<i>Pre-Approval Test Team</i>	Enter a valid group name if this state is desired in the generic. Entry in this field is optional.

Fields	Values
<i>Approval Team</i>	Enter a valid group name. <i>This is a mandatory field.</i>
<i>Manufacturing Team</i>	This field is reserved for a later release of Sablime.
<i>Software QA Team</i>	Enter a valid group name if this state is desired in the generic. Entry in this field is optional.

Related Commands

closegen
initsab
mvgen
newgen
setgroup (in the *User's Reference Manual*)

closegen Command

closegen - close a generic.

Synopsis

closegen [*keyword=value* [*keyword=value...*]] [-**help** | -?]]

g = *generic*
copyto = *list*
prompt = *y* | *n*



CAUTION:

The default generic is the setup generic.

Description

The `closegen` command is used to close a generic when all work on the generic has been completed and no further work on the generic is anticipated. A generic can be closed only when all MRs associated with the generic are in the *closed* state.

During processing, the appropriate records in the G, GS, GT, COM, CRIT, HC, PDEP, PDI, and SNAP relations are moved to the Inactive Database. In addition, for files that are common to more than one generic, the closed generic is removed from the common field in the GS relation record.



NOTE:

Depending on the number of files and deltas in the generic, processing may take considerable time; consequently, we recommend that `closegen` be run as a background process, using the Command Line interface.



CAUTION:

Once a generic is closed, the only operations that can be performed on it are query, report, ssql, and retrieving a snapshot. A closed generic cannot be reopened.

Used By

sablime login

Sends Mail To

DBA, GA

Field Descriptions

The fields that appear in the Curses Forms interface, the keywords that may be used in the Command Line interface, and the values associated with each, are shown below.

Fields and Keywords	Values
<i>Generic</i> <i>g</i>	The name of the generic to be closed. Default: the setup generic.
<i>Copy To</i> <i>copyto</i>	The PTS ID, email address, group name, or comma-separated list of PTS IDs, email addresses or group names of people to be sent copies of mail. The mail sent as a result of confirming this command is sent to each person included in the list. You can enter up to 256 characters.
<i>prompt</i>	y n specifies whether prompts should be displayed and input requested.

Related Commands

addgen
closemr
initsab
mvgen
newgen

dbcross Command

dbcross - cross-check information in the Active and Inactive Databases.

Synopsis

```
dbcross [-a] [-g] [-rrel1[,rel2]] [ g=generic ]
```

Description



NOTE:

In multi-machine projects, the `dbcross` command must be run on the host where the databases are located.

The options mentioned below permit the user to limit the audit to the specified subsets of the database.

If no options are specified, `dbcross` will check all relations in the ADB, IDB, and GDB. The IDB will only be checked if no options are specified.

The options are as follows:

-a causes all of the relations in the Active Database to be checked.

-g causes all of the relations in the Global Database to be checked.

-rrel1[,rel2] causes the relations listed to be checked. The relations listed may be in the Active or Global Database.

g=generic causes the records related to the specified generic to be checked.

For a complete discussion of `dbcross` and the other the audit commands, see *Managing the Databases* in Chapter 4, *Other Administrative Procedures*.

Used By

Anyone

Sends Mail To

Sends no mail

dbdelta Command

dbdelta - compare information in the Active, Inactive, and Source Databases.

Synopsis

dbdelta [**-cms** *yymmdd*] [**g=generic**]

Description



NOTE:

In multi-machine projects, the `dbdelta` command must be run on the host where the databases are located.

-cms *yymmdd* Databases converted from “CMS” have some characteristics that `dbdelta` considers errors. This flag causes these to be ignored for deltas earlier than *yymmdd*.

g=generic Limits the audit to looking at records related to the specified generic.

For a complete discussion of `dbdelta` and the other audit programs, see *Managing the Databases* in Chapter 4, *Other Administrative Procedures*.

Used By

Anyone

Sends Mail To

Sends no mail

dbedit Command

dbedit - edit database information.

Synopsis

dbedit [-help | -?]

Description

This command locks each named database file (tuple) so that it cannot be accessed by other commands while it is being edited. It then puts the file into the administrator's favorite editor for changes. When the administrator leaves the screen, the file is unlocked.

The DBA should be familiar with the databases and the relation structures before attempting to edit the databases. See Chapter 5, *The Sablime Databases* for information about the databases.



NOTE:

In multi-machine projects, the dbedit command must be run on the host where the databases are located.



CAUTION:

Only highly experienced Sablime DBAs should use this command to change database information. If a change requires editing for an extended period of time, shut down the Sablime databases with the dbstop command before using dbedit to make the necessary change.



NOTE:

This command can only be run using the Curses Forms interface.

Used By

DBA, any user given permission by the command permissions feature

Sends Mail To

Sends no mail

Field Descriptions

The fields on the dbedit screen, and appropriate values for them, are explained below:

Fields	Values
<i>Database</i>	Enter active (default), global , or inactive to specify the desired database.
<i>Relation</i>	Enter the name of the relation containing the data to be changed. A menu is displayed showing the acceptable entries for the database you selected in the previous field.
<i>Key for Edit</i>	<p>Enter the data in the first field of the record to be edited.</p> <ul style="list-style-type: none"> n If the tuple for this key exists, the following message is displayed: HIT A <CR> IN THE 'TUPLE FILE TO EDIT' FIELD TO CONTINUE. The cursor moves to the <i>Tuple to Edit</i> field. n If the tuple for this key does not exist, the following message is displayed: THE TUPLE FILE TO EDIT DOES NOT EXIST, TO CONTINUE, HIT <CR>. The cursor remains in this field to allow you to change your entry.
<i>Key for Edit</i>	<p>Press RETURN. You are now put into your favorite editor with the information from the selected tuple displayed for edit. After making the necessary changes, exit the editor in the usual manner.</p> <p>Sablime checks that the tuple records contain the correct number of fields and that no blank lines have been included. If either of these conditions is not met, an error message is displayed. When the entries are confirmed, the tuple is updated in the selected database.</p>



CAUTION:

Be very careful when you edit the data, because Sablime does not perform any field verification checks on the modifications other than those noted above.

Related Commands

mredit
mrgedit
setrel

dbhash Command

dbhash - generate a two-character tuple file name from the first field of a record.

Synopsis

`dbhash [-s] record`

Description

Sablime uses this command internally to locate the tuple into which to put a new tuple file. The DBA can use dbhash to find a database record before using the dbedit command to change information in that record.

The -s option provides a shortened form of the output that is useful in scripts. For example, dbhash tpr970001 produces ip <-- tpr970001, while dbhash -s tpr970001 produces ip.

Used By

Any user

Sends Mail To

Sends no mail

Related Command

dbedit

dbstart Command

dbstart - restart a Sablime product after a dbstop command has been issued.

Synopsis

dbstart

Description

dbstart returns the SAB.stopfile in the FILES directory of the Active Database to zero length.



NOTE:

In multi-machine projects, the dbstart command must be run on the host where the databases are located.

As processing takes place, information like that shown below appears on the screen:

+ YOUR SABLIME PRODUCT [product] IS NOW OPERATIONAL
+ A Master Trace record has been generated for the Database Administrator.

If the product has not been stopped, the following message is displayed.

+ YOUR SABLIME PRODUCT IS ALREADY OPERATIONAL

Used By

DBA, any user given permission by the command permissions feature

Sends Mail To

Sends no mail

Related Commands

dbstop

dbstop Command

dbstop - shut down the Sablime databases for a product.

Synopsis

dbstop

Description

dbstop allows the DBA or SA to edit the SAB.stopfile stop file in the FILES directory of the Active Database for a particular product and shut down Sablime usage for that product. The stop file is a text file, and the DBA or SA should enter into it the reason for the shutdown. After you enter a message and save the file, Sablime is stopped and locked until the dbstart command is issued.



NOTE:

If your HMI command mode is set to **np**, the following standard message is written to the SAB.stopfile when you execute dbstop:

The Databases for Product [product_name] are presently shut down.

Sablime commands that update the database first examine the stop file. If the stop file has length greater than zero (i.e., there is data in the file), the contents of the file are printed and the Sablime command exits. The SAB.stopfile must always exist. If it doesn't, Sablime commands will not execute.

The commands dbhash, dbedit, dbstop, dbstart, query, dbcross, dbxcross, dbdelta, report, and ssql continue to work if the Sablime product databases are stopped. The dbstop command prints the contents of the file with a warning that the Sablime instance is stopped.



NOTE:

In multi-machine projects, the dbstop command must be run on the host where the databases are located.

**NOTE:**

This command may only be run using the Command Line interface.

As processing takes place, information like that shown below appears on the screen:

- + YOUR SABLIME PRODUCT [product] IS NOW STOPPED
- + A Master Trace record has been generated for the Database Administrator.

Used By

DBA, SA, any user given permission by the command permissions feature

Sends Mail To

Sends no mail

Related Command

dbstart

dbxcross Command

dbxcross - cross-check all the database relations related to the External MR Communications feature.

Synopsis

dbxcross [g=*generic*]

Description



NOTE:

In multi-machine projects, the dbxcross command must be run on the host where the databases are located.

g=*generic* Limits the audit to looking at records related to the specified generic.

For a complete discussion of dbxcross and the other audit programs, see *Managing the Databases* in Chapter 4, *Other Administrative Procedures*

Used By

Anyone

Sends Mail To

Sends no mail

ftd Command

ftd - customize or look at field information for a product.

Synopsis

ftd*keyword=value* [*keyword=value...*] [-help | -?]

fcn = add|modify|view|delete
cmmnd = *command name*
intkey = *key*
extkey = *string*
ftype = 1|2|3
mdtory = y|n
hide = y|n
sflag = y|n
scrnum = *number*
sccnum = *number*
mlength = *number*
nscpdg = *number*
scppos = left|right|above|below
lrsrbf = *number*
scfchr = *character*
mxppch = *number*
vhel = *number*
scpmpt = *name*
nscpmt = *name*
dvalue = *value*
value = *value*
copyto = *list*
prompt = y|n



CAUTION:

In the Command Line interface, ftd uses the default value, not the current value, for non-specified fields. When you use the Command Line interface, be sure to specify values for fields in which the default is not the desired value.



NOTE:

Be sure to put double quotation marks around any Command Line interface entry that contains spaces.

Description

⇒ **NOTE:**
See *Customizing Commands and Command Screens* in Chapter 3, *Customizing the User Interface*, for information about the kinds of modifications allowed to Sablime fields.

⇒ **NOTE:**
ftd changes to User-Definable Fields (UDFs) can be reflected by similar ftd changes made to the report and query commands. report and query cannot, however, reflect changes to fields other than UDFs.

You cannot change the size or placement of a Pop-Up Selection Window.

⇒ **NOTE:**
Because of the large number of keywords that you must specify on the command line and the complexity of the ftd screen, we recommend that you use the Curses Forms interface for the ftd command.

Used By

DBA, any user given permission by the command permissions feature

⇒ **NOTE:**
For some functions, no special permissions are needed.




Sends Mail To


Sends no mail


Field Descriptions

The fields that appear in the Curses Forms interface, the keywords that may be used in the Command Line interface, and the values associated with each are

listed below. Fields with an asterisk (*) must be included on the command line in the Command Line interface.

Fields and Keywords	Values
<p><i>Function</i> fcn</p>	<p>add modify view delete specifies the function you want to perform in PTS. (add and delete are reserved for the Sablime team.) When anyone other than a DBA executes this command, view is the default value and the only value accepted.</p> <p> NOTE: In the Command Line interface, if a value is not given, the default is modify. view does not work in the Command Line interface.</p>
<p><i>*Command</i> cmmnd</p>	<p>The name of the command for which the field is to be modified.</p>
<p><i>*Internal Key</i> intkey</p>	<p>The characters used as a key to the field name. They are shown in the Command Line interface descriptions of commands in the manuals. In the Curses Forms interface, a pop-up menu lists all the valid keys for the command specified.</p>
<p><i>*External Key</i> extkey</p>	<p>A character string to be used in the Command Line interface in the <i>keyword=value</i> format, which is set to a user-definable custom key field name. See <i>Customizing Commands and Command Screens</i> in Chapter 3 for further information.</p>
<p><i>Field Type</i> ftype</p>	<p>1 2 3 specifies the field type assigned.</p> <p>1 specifies that string entry or a Run-time Expansion Keyword will be used for this field.</p> <p> NOTE: If ftype=1, mxxpch is set to 0 and ignored in the Command Line interface.</p> <p>2 specifies that the field will display a Pop-Up Selection Window populated by a Sablime-Level or Product-Level Reserved Group.</p> <p>3 specifies that the field will display a Pop-Up Selection Window populated dynamically when the command is executed.</p> <p> NOTE: A field type cannot be changed from or to field type 3.</p>

Fields and Keywords	Values
<i>Mandatory</i> mdtory	<p>y n indicates whether data entry in the field is mandatory or optional.</p> <p> CAUTION: <i>You must not change a mandatory field to optional because Sablime needs that field to continue processing successfully. Most fields are necessary to ensure database integrity. If you want to make a field optional, call the Sablime hotline before you make the change. You can change optional fields to mandatory.</i></p>
<i>Hideable</i> hide	Indicates whether the current field can be hidden. This field is set by Sablime and is protected. It is ignored in the Command Line interface.
<i>Display</i> sflag	y n indicates whether the field should be displayed and used for data collection. The field is protected unless the field is hideable. The keyword is ignored if the field is not hideable.
<i>*Row Number</i> scrnum	An integer from 0 to 23 that indicates the row from the top of the screen where the field is to appear in the Curses Forms interface.
<i>*Column Number</i> sccnum	The column from the left of the screen where the field is to appear in the Curses Forms interface. This number must be greater than the length of the <i>Screen Label</i> (scpmpt) entry. The maximum combined length of the <i>Screen Label</i> (sccnum) and <i>Length</i> (mlngth) entries is 79 characters.
<i>*Length</i> mlngth	The maximum number of characters displayed by the fill characters in the Curses Forms interface. This number must be less than or equal to 80 minus the length of the <i>Screen Label</i> (scpmpt) entry. The maximum combined length of the <i>Length</i> (mlngth) and <i>Screen Label</i> (scpmpt) entries is 80 characters. If <i>L/R Scroll Size</i> (lrsrbf) is 0, <i>Field Length</i> (mlngth) is the maximum length allowed for the field.
<i>Attribute</i>	0 is the only acceptable entry.
<i>*Prompt Padding</i> nscpdg	An integer from 1 to 78 that indicates the amount of padding allowed in the Command Line interface.
<i>Prompt Position</i> scppos	<p>left right above below specifies the position of the screen label in relation to the data entry display in the Curses Forms interface.</p> <p>Default: left</p>

Fields and Keywords	Values
<i>L/R Scroll Size</i> lrsrbf	An integer that indicates the maximum number of characters that can be entered for this field. The number entered must be 0 (zero) or between 10 and 1024, inclusive. If the entry is 0, the maximum field size is the same as the <i>Length</i> (mLength) entry; if the entry is other than 0, it must be greater than the <i>Length</i> (mLength) entry.
<i>Fill Character</i> scfchr	The character to be used to indicate the data value entry area in the Curses Forms interface. The usual entry is the underscore character (_).
<i>Popup Selections</i> mxxpch	The number of menu entries the user can select as data entry. This number allows the user to select more than one item from a Pop-Up Selection Window.  NOTE: If <i>Field Type</i> (ftype) is set to 1, this field is set to 0 and ignored in the Command Line interface. For field types 2 or 3, an integer from 1 to the number of items in the menu can be specified.
<i>*Verbose Help Key</i> vhlp	An integer from 1 to 9999 to be used as the name of the file containing the verbose help message for this field. The verbose help messages for UDFs are 6001-6050.
<i>Screen Label</i> scpmpt	The prompt that will appear as the field label when the user is in the Curses Forms interface. You can enter up to 60 characters.
<i>*No Prompt Help</i> nscpmt	The string of characters that is displayed for the field in the Command Line interface help.
<i>Default Value</i> dvalue	A string or a Run-time Expansion Keyword to be used as a default entry for this field. This field is valid only if <i>Field Type</i> (ftype) is 1. The default entry must contain fewer characters than the field length limit set by the <i>Length</i> (mLength) keyword or by the <i>L/R Scroll Size</i> (lrsrbf) keyword.
<i>Group/File</i> fvalue	If the <i>Field Type</i> (ftype) for this field is 2, enter either a Sablime-Level or Product-Level Reserved Group name. This information is loaded into the Pop-Up Selection Window for the field.
<i>Copy To</i> copyto	The PTS ID, email address, group name, or comma-separated list of PTS IDs, email addresses or group names of people to be sent copies of mail. The mail sent as a result of confirming this command is sent to each person included in the list. You can enter up to 256 characters.
prompt	y n specifies whether prompts should be displayed and input requested. Default: y

hotline.ck Script

hotline.ck - check for common problems that have been reported to the Sablime hotline.

Synopsis

hotline.ck

Description

The main focus of this script is to detect incorrect permissions on such items as the Sablime executables and the Sablime databases, as well as several other key directories and tools that Sablime requires for daily operations.

Although `hotline.ck` is similar to the audit scripts (i.e., `dbdelta`, `dbcross`, `dbxcross`), it need not be executed daily, but only when your Sablime environment gives System Error Messages or the commands do not execute correctly.



NOTE:

`hotline.ck` should be run on both satellite(s) and host.

To run `hotline.ck`, the *sablime* login must set up for a generic in the appropriate product and execute the following command at the UNIX shell prompt:

```
nohup hotline.ck > hotline.out &
```

This command executes the `hotline.ck` script in the background and redirects the output to a file named `hotline.out`. Since this script may take a while to execute, we recommend that you execute it in `nohup` mode. (For further information on `nohup`, refer to the UNIX manual page.) You can monitor the output file to check for any errors. A complete listing of the possible error messages from this script appears in Table D4 in Appendix D *Audit Program Messages*.

The output of this script is in the following format:

Heading: ***** <*> ***** LIST OF ITEMS BEING CHECKED

Error Message: <*> List of possible error messages

If no errors are found, the output contains only a list of headings. If errors are detected, make the necessary changes and re-execute `hotline.ck`.

For a complete discussion of this script and of the audit programs, see *Managing the Databases* in Chapter 4, *Other Administrative Procedures*.

Used By

sablime login

Sends Mail To

Sends no mail

initsab Script

initsab - establish a new Sablime instance with its first product or add a new product to an existing Sablime instance.

Synopsis

initsab

Description

See *Running the Installation Script* in Chapter 2, *Installing Sablime*, for detailed information about the initsab script.

Used By

sablime login

Sends Mail To

Sends no mail

mhist/nohist Commands

mhist/nohist - keep/do not keep MRG history records.

Synopsis

mhist
nohist

Description

The *sablime* login can use the `mhist/nohist` commands to control whether MR history records are kept for the product for which the *sablime* login is currently set up. `mhist` turns on the history flag for the product; `nohist` turns off the history flag for the product. When the history flag is on, history records are kept of the changes through which an MR goes in its life cycle. We strongly recommend that you keep the history flag turned on.

The *MRG History* records are stored in the `adb/product/FILES/mrhistory/generic` directory. (See chapter 5, *The Sablime Databases*.) Each file in the directory is named for an MR or spawned MR and contains a record for each transaction affecting that MR, showing the date and time, the PTS ID, the command issued, and, when appropriate, some data that was entered. The records of a typical file are shown below.

File name: `sab970043`

Partial Contents:

```
04/16/97 15:10:32 [ljh] accept
04/16/97 15:11:45 [ljh] assign ljh 3
04/18/97 10:33:27 [ljh] submit n
```

The `mhist/nohist` commands provide a quick way to change the setting of the history flag. You can achieve the same effect with `setrel`. (See the ADM subcommand for details.)



NOTE:

In multi-machine projects, the `mhist/nohist` commands must be run on the host where the databases are located.

When you enter the `mhist` command, the following message is displayed:

+ The ADM Relation has been modified to turn on the History Flag

When you enter the nohist command, the following message is displayed:

+ The ADM Relation has been modified to turn off the History Flag

Used By

DBA

Sends Mail To

Sends no mail

Related Command

setrel (ADM)

mmail/nomail Commands

mmail/nomail -turn on/turn off Sablime mail processing.

Synopsis

mmail
nomail

Description

mmail turns on the mail flag for the product for which the *sablime* login is set up; nomail turns the mail flag off for that product. When the mail flag is on, mail is sent to the default recipients to inform them of the progress of an MR. We recommend that you keep the mail flag turned on.

The mmail/nomail commands provide a quick way to change mail processing. You can achieve the same result with setrel. (See the ADM subcommand for details.)

When you enter the mmail command, the following message is displayed:

+ The ADM Relation has been modified to turn on the Mail flag

When you enter the nomail command, the following message is displayed:

+ The ADM Relation has been modified to turn off the Mail flag

Used By

sablime login

Sends Mail To

Sends no mail

Related Command

setrel (ADM)

mredit Command

mredit - edit the data in the MR and ORG relations and in the Description file entered through the create, fcreate, review, and spawnmr commands.

Synopsis

mredit*keyword=value* [*keyword=value...*] [-help | -?]

mr = MR Number
org = name
sev = severity
odate = date
rdate = date
sys = system
sub = subsystem
mod = module
rel = release
pd = phase
site = site
acdate = activate date from defer
ase = actual study effort
ee = estimated study effort
rsn = reason if reason code is other
rsncode = reason code from defer
studydev = assigned developer for study
studydue = due date from study
studysev = severity from study
mrudf1 = string
mrudf2 = string
mrudf3 = string
mrudf4 = string
mrudf5 = string
cat = category
abst = abstract
desc = filename
gsfile = filename
copyto = list
prompt = yln



NOTE:

Unspecified fields retain their previous value. Any field that has no current value is overwritten by the default value (if any) for that field.

Description

Data can be edited for any MR that is not closed.



NOTE:

If the mredit command is used to change the MR data in a higher-level cascading field (i.e., System or Subsystem) and data exists for a lower-level field (i.e., Subsystem or Module), data must be entered for both the higher-level and the lower-level field.

Used By

MRA, any user given permission by the command permissions feature

Sends Mail To

MRA

Field Descriptions

The fields that appear in the Curses Forms interface, the keywords that may be used in the Command Line interface, and the values associated with each are listed below. Fields preceded by an asterisk (*) must be included on the command line.

Fields and Keywords	Values
<i>*MR Number</i> mr	The number of the MR for which the data will be changed. In the Curses Forms interface, when the number is entered the rest of the fields (except for <i>Request Desc File</i>) are populated with the current data.
<i>Originator PTS ID</i> org	The changed PTS ID. The system verifies that the ID is valid and that it exists in the PTS relation of the Active Database. Default: the current user's PTS ID.
<i>Request Severity</i> sev	The changed <i>Request Severity</i> . Possible values are 1,2,3, and 4. Default: 3.
<i>Origination Date</i> odate	The changed <i>Origination Date</i> . Default: the current date.
<i>Required Date</i> rdate	The changed <i>Required Date</i> .
<i>Product</i>	This field is protected.
<i>System</i> sys	The changed <i>System</i> name. In the Curses Forms interface, a menu displays the systems for the product.
<i>Subsystem</i> sub	The changed <i>Subsystem</i> name. In the Curses Forms interface, a menu displays the subsystems for the system.

Fields and Keywords	Values
<i>Module</i> mod	The changed <i>Module</i> . In the Curses Forms interface, a menu displays the modules for the system and subsystem.
<i>Release Detected</i> rel	The changed <i>Release Detected</i> . In the Curses Forms interface, a menu displays the releases for the product.
<i>Phase Detected</i> pd	The changed <i>Phase Detected</i> . In the Curses Forms interface, a menu displays the phases for the product.
<i>Site</i> site	The changed <i>Site</i> . In the Curses Forms interface, a menu displays the sites for the Sablime instance.
<i>Activate Date from Defer</i> acdate	The date by which the MR is to be reactivated.
<i>Actual Study Effort</i> ase	The actual effort in staff days to study the MR.
<i>Estimated Study Effort</i> ee	The estimated effort in staff days to study the MR.
<i>Reason if Reason Code is Other</i> rsn	The reason for deferring the MR.
<i>Reason Code from Defer</i> rsncode	The reason why the MR has been deferred.
<i>Assigned Developer for Study</i> studydev	The PTS ID of the developer assigned to study the MR.
<i>Due Date from Study</i> studydue	The date by which the assigned developer is expected to propose a solution to the MR.
<i>Severity from Study</i> studysev	The severity assigned to the MR.
<i>MRUDF1-MRUDF5</i> mrudf1-mrudf5	These are User-Definable Fields (UDFs) that can be used to store product-specific information. Your Sablime Database Administrator may set up these fields to store up to 256 characters. Pop-up menus may also have been designed for use in these fields. These fields will not be displayed on your screen if your product does not use them. Please determine if these fields have been customized for your product(s), and if so, what keywords you will have to use to refer to them in the Command Line interface. The valid keywords are all listed in the Command Line help message.
<i>Category</i> cat	The changed <i>Category</i> . In the Curses Forms interface, a menu displays the categories for the product

Fields and Keywords	Values
<i>Abstract of Request</i> abst	The changed <i>Abstract of Request</i> .
<i>Request Desc File</i> desc	<p>The file to be used for a full description of the requested change. The field accepts up to 140 characters for the path and filename.</p> <p>In the Curses Forms interface, press RETURN to edit the existing description file. Sablime will create a temporary file with your favorite editor to allow you to edit the description. Make any desired changes and save the file. If you do not want to change the file, simply exit the editor.</p> <p>If you want to replace the existing description file, enter the full path to the new file. Sablime will create a temporary file with your favorite editor to allow you to edit the new description. Make any desired changes and save the file. If you do not want to change the file, simply exit the editor.</p> <p>If you want to change the description file using the Command Line interface, you must create a file before you issue the command and include the filename as a value for the desc keyword on the command line. If this field is not specified, the previous file is untouched.</p>

Fields and Keywords	Values
<p><i>Global Solution File</i> <i>gsfile</i></p>	<p>The file to be used for a full description of the Global Solution file. The field accepts up to 140 characters for the path and file name (this is the file that is generated when the MR is studied and a proposal is made before the MR is accepted).</p> <p>In the Curses Forms interface, press RETURN to edit the Global Solutions file. Sablime will create a temporary file with your favorite editor to allow you to edit the file (this is the file that is generated when the MR is studied and a proposal is made before the MR is accepted). Make any desired changes and save the file in the usual manner. If you do not want to change the file, simply exit the editor.</p> <p>If you want to replace the existing Global Solution file, enter the full path to the new file. Sablime will create a temporary file with your favorite editor to allow you to edit the new description. Make any desired changes and save the file in the usual manner. If you do not want to change the file, simply exit the editor.</p> <p>If no Global Solution file exists, this field is protected.</p> <p>If you want to change the global solution file in the Command Line interface, you must create a file before issuing the command and include the file name as a value for the <i>gsfile</i> keyword on the command line. If this field is not specified, the previous file is untouched.</p> <p>If no Global Solution file already exists, this keyword is ignored.</p>
<p><i>Copy To</i> <i>copyto</i></p>	<p>The PTS ID, email address, group name, or comma-separated list of PTS IDs, email addresses, or group names of people to be sent copies of mail. The mail sent as a result of confirming this command is sent to each person included in the list. You can enter up to 256 characters. If this field is not specified, the previous list is untouched.</p>
<p><i>prompt</i></p>	<p>y n specifies whether prompts should be displayed and input requested. Default: y.</p>

Related Commands

create
dbedit
mrgedit
mrnote

mrgedit Command

mrgedit - edit the data in the MG relation and in the Rejection, Resolution, Solution, Spawn, and Test Notes files populated through the MR commands.

Synopsis

mrgedit*keyword=value* [*keyword=value...*] [-help | -?]

mr = *mr*
g = *generic*
class = *item*
subclass = *item*
dev = *developer*
type = *item*
subtype = *item*
sev = *severity*
due = *date*
rcode = *code*
dupmr = *duplicate*
status = *status*
chgdate = *date*
emrflag = *flag*
spawns = *spawn*
rsn = *reason*
copyto = *list*
npage = *y|n*
hcode = *item*
pdi = *item*
ri = *release*
rc = *item*
rcs = *item*
ndc = *item*
ndcs = *item*
flttype = *type*
ae = *number*
pi = *item*
odp = *item*
ee = *number*
ast = *number*
tte1 = *number*
tte2 = *number*
tte3 = *number*
tte4 = *number*
tte5 = *number*
file_rej = *y|n*
file_res = *y|n*
file_sol = *y|n*
file_spwn = *y|n*
file_tnotes = *y|n*

`mrgudf1 = string`
`mrgudf2 = string`
`mrgudf3 = string`
`mrgudf4 = string`
`mrgudf5 = string`
`prompt = yln`

Description

Data can be edited for any accepted MRG.

Used By

GA, any user given permission by the command permissions feature



Sends Mail To


GA, Assigned Developer (if the Assigned Developer is changed)

Field Descriptions

The fields that appear in the Curses Forms interface, the keywords that may be used in the Command Line interface, and the values associated with each are listed below. Fields with an asterisk (*) must be included on the command line.

Fields and Keywords	Values
*MR Number <i>mr</i>	Specifies the number of the MRG for which the data will be changed.
Generic <i>g</i>	Specifies the name the generic for which the given MRG is to be edited. Default: the setup generic.
MR Class <i>class</i>	Specifies the changed class. Your entry must match a menu option. This keyword is ignored if the original class is <i>mixed</i> and the MR has been spawned.

Fields and Keywords	Values
<i>MR Subclass</i> subclass	<p>Specifies the changed subclass. Your entry must match a menu option. If you change the class entry and there is no current entry in the subclass field, a default subclass may be provided. If you change the class entry and there is already a subclass entry, it is untouched.</p> <p> CAUTION: <i>If you do not specify both class and subclass, be careful not to create a mismatch between class and subclass.</i></p>
<i>Developer</i> dev	<p>Specifies the changed developer. The developer can be changed only if data already appears in the field. This keyword is ignored if the MR has not been assigned. Mail is sent to both the old and new developers.</p>
<i>Commitment ID</i>	<p>This field is protected.</p>
<i>MR Type</i> type	<p>Specifies the changed type.</p>
<i>MR Subtype</i> subtype	<p>Specifies the changed subtype. Your entry must match a menu option.</p> <p> CAUTION: <i>If you do not specify both type and subtype, be careful not to create a mismatch between type and subtype.</i></p>
<i>Severity</i> sev	<p>Specifies the changed severity. Your entry must match a menu option. This keyword is ignored unless the MR is at or beyond the <i>assigned</i> state or in the <i>understudy</i> state.</p>
<i>Due Date</i> due	<p>Specifies the changed due date. This keyword is ignored unless the MR is at or beyond the <i>assigned</i> state or in the <i>understudy</i> state.</p>
<i>Reason Code</i> rcode	<p>Specifies the changed reason code. Your entry must match a menu option. This keyword is ignored unless the MR has been rejected, deferred, or moved to the <i>nochange</i> state and the MRG is in the <i>assigned</i>, <i>accepted</i>, or <i>nochange</i> state.</p>
<i>Reason</i> rsn	<p>Specifies the changed reason. Entry is mandatory only if the rcode is set to other. This keyword is ignored unless the MR has been rejected, deferred, or moved to the <i>nochange</i> state.</p>
<i>Duplicate MR Number</i> dupmr	<p>The duplicate MR number.</p>

Fields and Keywords	Values
<i>MRG Status</i> status	Contains the current MRG state of the MR. This field is protected. If the MR class is changed and the MRG is in one of the test states, the MRG state is changed to the proper test state.
<i>Last Change</i> chgdate	The latest MRG state-changed date. This field is protected.
<i>EMR Flag</i> emrflag	The External MR Link flag. This field is protected.
<i>Spawns</i> spawns	The number of spawned MRs for the specified MR in the specified generic. This field is protected.
<i>Copy To</i> copyto	Specifies the PTS ID, email address, group name, or comma-separated list of PTS IDs, email addresses, or group names of people to be sent copies of mail. The mail sent as a result of confirming this command is sent to each person included in the list. You can enter up to 256 characters.
<i>Do you need to edit more fields</i> npage	<p>In the Curses Forms interface, enter y (the default) if you want to edit MG relation fields that do not appear on the first screen. Otherwise, enter n. When you change the class of an MR from hardware to another class, you must edit the second screen if there is a pdi or hcode associated with the MR.</p> <p>In the Command Line interface, y n specifies whether any of the following fields (other than prompt) will be specified. If n is specified and the Command Line interface is used, all of the following fields specified (other than prompt) are ignored. If the Command Line interface is not used and npage is not used on the command line, specifying any of the following keywords sets npage to y. The default is n.</p>
<i>Hardware Code</i> hcode	A valid product-specific hardware code. This field is protected (ignored) unless the specified MR is a spawned hardware MR.
<i>PDI Number</i> pdi	A product-specific product development information number. This field is protected (ignored) unless the specified MR is a parent hardware MR.
<i>Release Introduced</i> ri	The changed release introduced.
<i>Root Cause</i> rc	The changed root cause.
<i>Root Cause Subcategory</i> rcs	The changed root cause subcategory.  CAUTION: <i>If you do not specify both rc and rcs, be careful not to create a mismatch between rc and rcs.</i>

Fields and Keywords	Values
<i>Non-Det Cause</i> ndc	The reason why the problem was not detected in the optimal detection phase. The entry must match one of the menu items.
<i>NDC Subcat</i> ndcs	A more detailed reason why the problem was not detected in the optimal detection phase. The entry must be appropriate for the item specified for non-detection cause. In the Curses Forms interface, a set of subcategories based on the preceding field appears. The entry must match one of the menu items.
<i>Fault Type</i> fltype	The type of fault that best categorizes the problem. The entry must match one of the menu items.
<i>Actual Effort</i> ae	The changed actual effort.
<i>Phase Introduced</i> pi	The changed phase introduced.
<i>Optimal Detection Phase</i> odp	The changed optimal detection phase.
<i>Estimated Effort</i> ee	The changed estimated effort.
<i>Actual Study Effort</i> ast	The changed actual study effort.
<i>Test Team 1 Effort</i> tte1	The changed test team 1 effort.
<i>Test Team 2 Effort</i> tte2	The changed test team 2 effort.
<i>Test Team 3 Effort</i> tte3	The changed test team 3 effort.
<i>Test Team 4 Effort</i> tte4	The changed test team 4 effort.
<i>Test Team 5 Effort</i> tte5	The changed test team 5 effort.
<i>Edit Rejection File</i> file_rej	Enter y or n to indicate whether you want to edit the existing Rejection File. If the MR has never been rejected and you enter y , an error message is displayed indicating that the file does not exist. If you change the entry from y to n at any time before confirming the screen, no update is made to the original file. Default: n

Fields and Keywords	Values
<i>Edit Resolution File</i> <i>file_res</i>	Enter y or n to indicate whether you want to edit the existing Resolution File. If you change the entry from y to n at any time before confirming the screen, no update is made to the original file. Default: n
<i>Edit Solution File</i> <i>file_sol</i>	Enter y or n to indicate whether you want to edit the existing Solution File. If a solution has never been proposed for this MR and you enter y , an error message is displayed indicating that the file does not exist. If you change the entry from y to n at any time before confirming the screen, no update is made to the original file. Default: n
<i>Edit Spawn Notes</i> <i>file_spwn</i>	Enter y or n to indicate whether you want to edit the existing Spawn Notes. If the MR has never been spawned and you enter y , an error message is displayed indicating that the file does not exist. If you change the entry from y to n at any time before confirming the screen, no update is made to the original file. Default: n
<i>Edit Test Notes</i> <i>file_tnotes</i>	Enter y or n to indicate whether you want to edit the existing Test Notes. If there are no existing Test Notes and you enter y , an error message is displayed indicating that the file does not exist. If you change the entry from y to n at any time before confirming the screen, no update is made to the original file. Default: n
<i>MRGUDF1-MRGUDF5</i> <i>mrgudf1-mrgudf5</i>	These are User-Definable Fields (UDFs) that can be used to store product specific information. Your Sablime Database Administrator can set up these fields to store up to 256 characters. Pop-up menus may have been designed for use in these fields. In the Curses Forms interface, these fields will not appear on your screen if your product does not use them. Please ascertain any customization that has been performed for your product(s).
<i>prompt</i>	y n specifies whether prompts should be displayed and input requested. You are not allowed to modify files in the Command Line interface.

Related Commands

accept
activate
approve
assign

dbedit
defer
fcreate
mredit
mrnote
propose
reject
study
submit
testassign
testpass

mtrace/notrace Commands

mtrace/notrace - keep/do not keep command trace records for a product that is under Sablime control.

Synopsis

mtrace
notrace

Description

The mtrace command turns on the trace flag; notrace turns off the trace flag for the product. When the trace flag is on, records are kept of the execution of every command for the product in the `adb/product_name/FILES/trace` directory in files that are named for each user (i.e., PTS ID).

The *MR Trace* records are stored as one file named for each user with an individual record for each transaction performed by that user showing the date and time, the PTS ID, the command issued, and, the condition of the transaction when ended. Each record is similar to the one shown below.

File name: stc

Partial Contents:

```
02/16/96 15:35:41 [stc] create CONDITION AT TIME OF ERROR ERR : \  
%Z% %M% [Version %I%] The [fname]  
parameter represents an invalid value.  
02/17/96 10:21:34 [stc] edget USER TERMINATION REQUESTED  
02/17/96 11:32:14 [stc] edget COMPLETED
```



NOTE:

The trace files can grow quite large over time. Periodically, the *sablime* login should remove them.

The mtrace/notrace commands provide a quick way to change the setting of the trace flag. You can achieve the same effect with setrel. (See the ADM subcommand for details.)

When you enter the mtrace command, the following message is displayed:

+ The ADM Relation has been modified to turn on the Trace flag

When you enter the notrace command, the following message is displayed:

- + The ADM Relation has been modified to turn off the Trace flag
- + A Master Trace Record has been generated for the Database Administrator

Used By

DBA

Sends Mail To

Sends no mail

Related Command

setrel (ADM)

mvgen Script

mvgen - rename an existing generic in the Sablime databases.

Synopsis

mvgen

Description

For detailed information about the script, see *Renaming a Generic* in Chapter 4, *Other Administrative Procedures*.



NOTE:

This script must be executed on the host machine.

Used By

sablime login

Sends Mail To

Sends no mail

Related Commands

dbstart
dbstop

newgen Script

newgen - add a new generic to an existing product in the Sablime databases.

Synopsis

newgen

Description

The newgen script comprises the five steps required to add a new generic to an existing product in the Sablime databases; each step is presented as a separate screen. The script starts by prompting you for the information required to define the new generic. Then it prompts you for the new directory structure file and updates the Source Database. Finally, it puts you into your favorite editor to update the xsablime.sh script, which will be modified as a result of the changes you make.



NOTE:

This script must be executed on the host machine or an NFS satellite machine.

For detailed information on newgen, see *Adding a Generic* in Chapter 4, *Other Administrative Procedures*.

Used By

sablime login

Sends Mail To

Sends no mail

Related Commands

addgen
closegen
initsab
mvgen
setgroup

primsdb Command

primsdb - add a set of files to the Sablime databases.

Synopsis

primsdb

Description

primsdb is a shell interface to a batch process running addgsrc or addisrc. As shipped, use of primsdb is restricted to the *sablime* login because it is a powerful command that can make major changes to the Source Database. If you want to give access to other users, change the permissions on primsdb to 755. However, we recommend that only very experienced users be allowed to run primsdb.



NOTE:

You cannot use primsdb to add files to Sablime at the top of your product (generic) directory structure (i.e., the directory cannot be "."). To add files to this directory, you must use the addisrc or addgsrc command.



NOTE:

In multi-machine projects, the primsdb command must be run on the host where the databases are located.

For detailed information about primsdb, see *Adding Files to a Generic* in Chapter 4, *Other Administrative Procedures*.

Used By

DBA

Sends Mail To

Sends no mail

Related Commands

addgsrc (in the *User's Reference Manual*)

addisrc (in the *User's Reference Manual*)

fcreate (in the *User's Reference Manual*)

setnode (in the *User's Reference Manual*)

setrel

screate Command

screate - identify problems with or request enhancements to Sablime.

Synopsis

`screate keyword=value [keyword=value...] [-help | -?]`

prodid = *product ID*
name = *name*
phone = *number*
site = *location*
odate = *date*
rdate = *date*
sev = *1|2|3|4*
sys = *system*
cmd = *command*
type = *modification|enhancement*
rel = *release detected*
abst = *abstract*
desc = *filename*
copyto = *list*
email = *return e-mail address*
sabaddr = *address*
prompt = *y|n*

Description

The `screate` command can be used to send an MR to the Sablime team from any machine that can send email to Lucent Technologies.

Four of the fields (*Product ID, Name, Telephone No, and Your Return Email Address*) of the `screate` command contain data about you and your product and are populated by Sablime from your PTS and PRX records. This information is used by the Sablime team when they respond to your MR.

When Sablime receives an MR from a customer, a Sablime MRA decides what action should be taken, and a message is sent to the customer explaining the action taken. The MR can be accepted for work or it can be rejected.

If it is accepted, a message like the following is sent.

```
From sablime Tue Apr 3 15:00 EDT 2002
Subject: Created MR reported by customer [Customer's Product ID]

Initiator: sablime
This mail is being sent to you in your capacity as:
  Copy-To List Member
MR [mr #] has been created by Team Sablime administrator.
<----- INFORMATION FOR MR sab020030 ----->
  Product:  sab++  Release:   v6.0
  System:   admin  MR Severity: 3
  Subsystem: setgroup  Required Date:
Abstract: Abstract text

[You have chosen not to include the MR description in your mail message]
<----->
```

If it is rejected, a message like the following is sent.

```
From sablime Tue Apr 3 15:00 EDT 2002
Subject: Killed MR reported by customer [Customer's Product ID]

Initiator: sablime

This mail is being sent to you in your capacity as:
  Copy-To List Member
MR [mr #] has been killed by Team Sablime Administrator.
Reason: reason text
<-----INFORMATION FOR MR sab0200086 ----->
  Product:  sab++  Release:   6.0
  System:   admin  MR Severity: 3
  Subsystem: setgroup  Required Date:

[You have chosen not to include the MR description in your mail message]
<----->
```



NOTE:

The screate response mail from the Sablime team does not include the description file.

Used By

DBA, any user given permission by the command permissions feature

Sends Mail To

Sends no mail

Field Descriptions

The fields that appear in the Curses Forms interface, the keywords that may be used in the Command Line interface, and the values associated with each are listed below. Fields with an asterisk (*) must be included on the command line.

Fields and Keywords	Values
<i>Product ID</i> prodid	The product for which you are reporting a problem.
<i>Name</i> name	<i>name</i> specifies the PTS ID of the person originating the MR. You can enter up to 65 characters. Default: the user's PTS ID.
<i>Telephone No</i> phone	<i>number</i> specifies the phone number of the person originating the MR. You can enter up to 15 characters. Default: the user's phone number from the PTS record.
<i>Site</i> site	The location of your project. If this field is specified, it can be used as a selection criterion in <i>sreport</i> . You can enter up to 40 characters; spaces are not allowed.
<i>Origination Date</i> odate	The date when the MR is originated. Default: the current date.
<i>Required Date</i> rdate	The date when changes in response to the MR are requested. The date must be equal to or greater than the current date. This field is optional.
<i>Request Severity</i> sev	1 2 3 4 indicates the severity of the request; critical= 1 , high= 2 , medium= 3 , and low= 4 . Default: 3
<i>*System</i> sys	The name of the component of Sablime in which the change is requested. You can enter up to 30 characters. In the Curses Forms interface, a menu displays the acceptable systems.
<i>*Command</i> cmd	The name of the command in which the change is requested. You can enter up to 30 characters. In the Curses Forms interface, a menu displays the available commands.

—Continued

Fields and Keywords	Values
<i>Type type</i>	modification enhancement specifies the type of change requested. Default: enhancement
<i>Release Detected rel</i>	The release of Sablime called by the setup script. This field is protected.
<i>*Abstract of Request abst</i>	A short synopsis of the problem or enhancement. You can enter up to 60 characters.
<i>*Request Desc File desc</i>	The file to be created for a full description of the requested change. You can enter up to 140 characters. If the MR is reporting a problem, you should supply a method of recreating the problem. In the Curses Forms interface, press RETURN and you are placed in a temporary file displaying the enhancement request template. When you leave your editor, the screate screen places the temporary file name in this field. In the Command Line interface, you must create a file for the desc keyword before you issue the command and include the file name as a value for the desc keyword on the command line.
<i>Copy To copyto</i>	The PTS ID, email address, group name, or comma-separated list of PTS IDs, email addresses, or group names of people to be sent copies of mail (without the description section). The mail sent as a result of confirming this command is sent to each person included in the list. You can enter up to 256 characters. If the project mail flag is on, mail is sent to the appropriate Database Administrator automatically.
<i>Your Return Email Addr email</i>	The email address of the person originating the MR. This field should include necessary gateway information. You can enter up to 50 characters. Default: the user's email address from the PTS record.
<i>Email Addr to Sablime sabaddr</i>	The email address of the Sablime team. This field should include the necessary gateway information. Default: sabmail@jaguar.stc.lucent.com
<i>prompt</i>	y n specifies whether prompts should be displayed and input requested. Default: y

Related Command

sreport

setperm Command

setperm - set correct file permissions for all Sablime commands.

Synopsis

setperm

Description

setperm is used to reset file permissions if they are incorrect. setperm uses information from the Sablime environment established by the *dot sablime* command. If you do not have a Sablime environment set, setperm prompts for the necessary information.

Used By

sablime login

Sends Mail To

Sends no mail

setrel Command

The `setrel` command allows the administrator to modify or view the data in the ADM, CAS, CP, CRIT, ES, PR, and PRX relations.

Synopsis

```
setrel [-help | -?]
```

Description

For information about the relations that can be modified by `setrel`, see the sections on the individual subcommands, below, and Chapter 5, *The Sablime Databases*.

The `setrel` command can be used in single-machine mode or from the host machine in multi-machine mode.

Used By

DBA, MRA, GA, any user given permission by the command permissions feature, as follows:


- n Only the *sablime* login may view or modify the ADM relation.
- n The DBA can modify or view any of the other relations controlled using this command.
- n The MRA can create, delete, or modify CRIT records for Automatic Routing.
- n The GA can create, delete, or modify CRIT records for Automatic Assignment.
- n Any user can view CRIT records.

Sends Mail To

DBA

Field Descriptions

The screen fields and the values associated with each are listed below:

Fields	Values
<i>Relation rel</i>	Enter the menu item for the relation to be accessed.  NOTE: A second screen is displayed in response to your selection. See the following sections for information about each of the subcommands.
<i>Copy To copyto</i>	Enter the PTS ID, email address, group name, or comma-separated list of PTS IDs, email addresses, or group names of people to be sent copies of mail. The mail sent as a result of confirming this command is sent to each person included in the Copy To list. This line scrolls to accept an entry of up to 256 characters.

ADM Subcommand (setrel)

The ADM relation controls the groups for DBAs, MRAs, and SAs, standardizes MR numbers for the project, establishes the kind of tracking and mail distribution that will take place, and sets various product-level flags that determine the use of various Sablime capabilities for the project.

The ADM relation is created when Sablime is installed. The *sablime* login can modify or view the single record contained in the ADM relation; it cannot delete it.

The DBA can toggle the **yes** or **no** flag for the *hist*, *mail*, or *trace* fields by entering commands at the system prompt in the Command Line interface. It is not necessary to be logged in as *sablime* to use these quick commands. The commands and their effects are shown in the following table.

Table 6-1. ADM Quick Commands


Command	Result
mhist	MR History records will be kept.
nohist	MR History records will not be kept.
mmail	Mail will be sent by Sablime.
nomail	Mail will not be sent by Sablime.
mtrace	Command trace records will be kept.
notrace	Command trace records will not be kept.

**NOTE:**

In multi-machine projects, these commands must be run on the host where the databases are located.

Field Descriptions

The **ADM** screen fields, and the values associated with each, are listed below:

Fields	Values
<i>Function</i> <i>srlfcn</i>	<p>Enter the type of function you want to perform. The allowable entries are:</p> <ul style="list-style-type: none"> n modify (default)—change an existing field n view—look at a relation without making any changes. <p> NOTE: The following information assumes you are modifying the ADM relation.</p>
<i>DBA Group</i> <i>sadbadm</i>	<p>Usually the group name is <i>dba</i> or <i>dba_productname</i> and should never be changed because the <i>DBA Group</i> field identifies the product for the ADM relation. The system verifies that the name is valid and that it exists in the GRP relation of the Active Database. When you fill in this field, the rest of the fields on this screen are automatically populated with their current values.</p>
<i>MRA Group</i> <i>samradm</i>	<p>Enter the name of the MRA group that is valid for the current product. The system verifies that the name is valid and that it exists in the GRP relation of the Active Database. The name is usually <i>mra</i> or <i>mra_productname</i>.</p>
<i>SA Group</i> <i>sasrc</i>	<p>Enter the name of the SA group that is valid for the current product. The system verifies that the name is valid and that it exists in the GRP relation of the Active Database. The name is usually <i>sa</i> or <i>sa_productname</i>.</p>
<i>HA Group</i> <i>sahadm</i>	<p>Enter the name of the HA group that is valid for the current product. The system verifies that the name is valid and that it exists in the GRP relation of the Active Database. The name is usually <i>ha</i> or <i>ha_productname</i>.</p>

Fields	Values
<i>MR Prefix</i> <i>saprefix</i>	<p>Enter the prefix to be used as part of the MR number. The field accepts up to five alphanumeric characters; no special characters are allowed.</p> <p>Because MRGs that have been spawned use a decimal point (.) in the MR number, the MR prefix assigned in the ADM relation should not contain a period.</p>
<i>MR Suffix</i> <i>sanum</i>	<p>Enter the number that Sablime will use to number the next MR created. This field accepts one to six integers (i.e., integers between 0 and 999999).</p> <p>The recommended code for a numbering system is <i>yy</i><i>n</i>, where <i>yy</i> represents the last two digits of the year in which tracking begins and <i>n</i> is the number to be used for the next MR created (e.g., 960000).</p> <p>The maximum combined length of the MR prefix and MR suffix is 11 characters.</p> <p>The length of the MR suffix can be increased to eight; use the <i>ftd</i> command to modify the length of the <i>setrel</i> command's <i>sanum</i> field; then use the ADM subcommand of the <i>setrel</i> command to reset the <i>MR Suffix</i> field to accommodate the new length. If you increase the MR suffix to eight, the maximum length of the MR prefix is three characters.</p>
<i>Auto Dependency</i> <i>sadep</i>	<p>Enter file-level, line-level, or none to indicate the default for automatically creating MR dependencies while executing the <i>edput</i> command for your product. This field is applicable to non-binary files only. For binary files, there is implicit file-level dependency.</p> <p>If you select file-level, any unapproved MRs that touch the same file are made dependent automatically.</p> <p>If you select line-level, any unapproved MRs that touch the same line in a file are made dependent automatically.</p> <p>If you select none, no dependencies are automatically created.</p>
<i>Override Dep Flag</i> <i>sadover</i>	<p>Enter y or n to indicate whether the AD should be allowed to override the default for automatically establishing MR dependencies.</p>
<i>IPM Flag</i> <i>saipm</i>	<p>Enter y or n (default) to indicate whether your project wants to collect In-Process Metrics data for this product.</p>
<i>Auto Route Flag</i> <i>saarte</i>	<p>Enter y or n (default) to indicate whether your project wants to route MRs to the appropriate MRAs automatically for this product.</p>
<i>Auto Assign Flag</i> <i>saasgn</i>	<p>Enter y or n (default) to indicate whether your project wants to assign MRs to the appropriate ADs automatically for this product.</p>

Fields	Values
<i>Reassign Flag</i> <i>sarasgn</i>	Enter all , y , or n (default) to indicate who is allowed to reassign or unassign MRs for this product. y allows only the AD or GA or the owner of the AD or GA group to reassign or unassign an MR. n allows only the GA or owner of the GA group to reassign or unassign an MR. all allows any member of the project to reassign or unassign MRs.
<i>Trace Flag</i> <i>satrace</i>	Enter y (default) or n to indicate whether command use should be traced. Trace files are per-user and track each command invocation and termination status.
<i>Mail Flag</i> <i>samail</i>	Enter y (default) or n to indicate whether mail should be sent.
<i>History Flag</i> <i>sahist</i>	Enter y (default) or n to indicate whether MR History records should be kept for a given generic. History files are per-MRG and track MRG state changes and dependencies.
<i>Mail Interval</i> <i>samint</i>	obsolete
<i>Version Control Tool for Non-Binary Files</i> <i>sadvct</i>	Enter SBCS , SCCS , or Either to specify a version control tool for non-binary files. If Either is specified, whenever users are adding files with addisrc or primsdbs, the <i>Control Tool</i> field allows a choice of SBCS or SCCS. If a default is chosen here, the addisrc and primsdbs <i>Control Tool</i> field choice is protected for non-binary files.

Related Commands

addisrc (in the *User's Reference Manual*)
 create (in the *User's Reference Manual*)
 fcreate (in the *User's Reference Manual*)
 primsdbs
 review (in the *User's Reference Manual*)

CAS Subcommand (setrel)

The CAS relation contains information about the relationship between several pairs of menus (e.g., system and subsystem, class and subclass, etc.).

The **CAS** screen fields, and the values associated with each, are listed below:

Fields	Values
<i>Function srlfcn</i>	<p>Enter the type of function you want to perform. The allowable entries are:</p> <ul style="list-style-type: none"> n add—establish a new group name for a cascading menu n delete—delete an existing group name for a cascading menu n modify (default)—change an existing group name for a cascading menu n view—look at group name for cascading menu information without making any changes.



NOTE:

The following information applies when you add a new group name.

<i>Type sctype</i>	Enter the code for the cascading menu for which a group name is to be added. A menu showing allowable entries and an explanation of each is displayed.
<i>Upper Level Key sckey</i>	Enter the name of the entry for the upper-level field (e.g., document for the <i>Class</i> field).
<i>Lower Level Group Name scgrp</i>	<p>Enter the name of the group that contains the lower-level entries for the Upper Level Key specified.</p> <p> NOTE: This group must be created with the <code>setgroup</code> command before entry here.</p>

Related Command

`setgroup` in the *User's Reference Manual*

CP Subcommand (setrel)

The CP relation contains information about the users who can execute a particular command and the users who are recipients of the email generated by the execution of that command. The information stored includes the command name, the generic name, the function name, and lists of executors and email recipients. For further information about this relation, see the section *Customizing Command Executors and Email Recipients* in Chapter 4, *Other Administrative Procedures*.

The **CP** screen fields, and the values associated with each, are listed below:

Fields	Values
<i>Function</i> <i>srlfcn</i>	<p>Enter the type of function you want to perform. The allowable entries are:</p> <ul style="list-style-type: none"> n add—create a new command-permissions record. n delete—delete an existing command-permissions record. n modify (default)—change an existing command-permissions record. n view—look at command-permissions record without making any changes. <p>The following information applies when you add a new command-permissions record or modify an existing one.</p>
<i>Command</i> <i>cpcmd</i>	Enter the name of the command-permissions record you want to change.
<i>Command-Function</i> <i>cpcmd_fcn</i>	Enter the function of the command for which you are adding or changing information.
<i>Generic</i> <i>cpgen</i>	Enter the name of the generic. The default is the setup generic.
<i>Executor(s)</i> <i>cpexec</i>	Enter a comma-separated list, a group, or a keyword (see the table of <i>Executor Field Keywords</i> , below).
<i>Email Recipient(s)</i> <i>cpemail</i>	Enter a comma-separated list, a group, or a keyword (see the table <i>Email Recipient Field Keywords</i> , below).

Table 6-2. Executor Field Keywords

Keyword or Value	Meaning
__ALL	All internal, product-authorized PTSid's
__NONE	No one may execute
__DEFAULT	As-delivered permissions
PTS ID	PTS ID
__AD	Assigned Developer
PTSid Group	All group members (group type ptsid)

Table 6-3. Email Recipient Field Keywords

Keyword or Value	Meaning
__NONE	No email recipients
__DEFAULT	As-delivered email recipients (default)
__AD	Assigned Developer (MG/MRX assignee)
__ORIG	MR originator
PTS ID	PTS ID
PTS ID Group	All group members (group type ptsid)
Email Group	All group members (group type other)
Email Address	Email addressee

CRIT Subcommand (setrel)

The CRIT relation contains the Automatic Routing Criteria and the Automatic Assignment Criteria information for a product. Automatic Routing Criteria can be established only by a DBA or MRA; Automatic Assignment Criteria can be established only by a DBA or GA. Anyone can view the established criteria for either capability.

- n Once criteria have been established for Automatic Routing of MRs and the *Auto Route Flag* field in the ADM relation has been set to **y**, the established criteria are compared to data fields for the created MR. If the MR data matches all the criteria, mail informing the MRA of the creation of the MR is sent automatically to the subset of MRAs or group of MRAs named in the Criteria Owner field.
- n Once criteria have been established for Automatic Assignment of MRs, the *Auto Assign Flag* field in the ADM relation has been set to **y**, and the *Auto Assign* field in the command used to accept the MR (*accept* or *spawnmr*) is set to **y**, the established criteria are compared to data fields for the accepted MRs. If the MR data matches all the criteria, the MR is assigned



automatically to the developer or group of developers named in the *Criteria Owner* field and a mail message informing the developer of the assignment is sent to each person.

⇒ NOTE:
 If the criteria match for more than one owner (i.e., there is more than one Criteria Owner with the same set of matching criteria) for the Auto Assign function, the MR is assigned to only the first owner matched, and a message is sent stating that more than one match was found. If a group name is entered as the criteria owner, all members of that group receive mail messages.

⇒ NOTE:
 If you want to establish multiple criteria for the same field, you must create a group that contains the criteria and use that group name for the field.

The **CRIT** screen fields, and the values associated with each, are listed below:

Fields	Values
<i>Criteria Type</i> <i>crtype</i>	Enter route or assign to specify the type of criteria to be established. <ul style="list-style-type: none"> n If route is entered, you must be a DBA or MRA to add, modify, or delete criteria. Anyone can view criteria. n If assign is entered, you must be a DBA or GA to add, modify, or delete criteria. Anyone can view criteria.
<i>Function</i> <i>crfcn</i>	Enter the type of function you want to perform. The allowable entries are: <ul style="list-style-type: none"> n add—establish a new set of criteria n delete—delete an existing set of criteria n modify—change an existing set of criteria n view (default)—look at a set of criteria without making any changes. view is not available in the Command Line interface. <p>⇒ NOTE: The following information applies when adding new criteria.</p>
<i>Criteria Owner</i> <i>crowner</i>	Enter the PTS ID or group name to whom MRs should be automatically routed or assigned.
<i>System</i> <i>crsys</i>	Enter the name or group name of systems to be matched.

Fields	Values
<i>Subsystem</i> <i>crsubsys</i>	Enter the name or group name of subsystems to be matched.
<i>Module</i> <i>crmod</i>	Enter the name or group name of modules to be matched.
<i>Site</i> <i>crsite</i>	Enter the name or group name of sites to be matched.
<i>Release Detected</i> <i>crrel</i>	<p>Enter the name or group name of releases to be matched.</p> <p> NOTE: At least one of the above fields must be entered for Automatic Routing.</p> <p> NOTE: The following fields are used only for Automatic Assignment.</p>
<i>Generic</i> <i>crgen</i>	Enter the name or a group name of generics to be matched. This is a mandatory field for Automatic Assignment.
<i>MR Class</i> <i>crclass</i>	Enter the name or group name of MR classes to be matched.
<i>MR Subclass</i> <i>crsclass</i>	Enter the name or group name of MR subclasses to be matched.
<i>MR Type</i> <i>crmrtype</i>	Enter the name or group name of MR types to be matched.
<i>MR Subtype</i> <i>crstype</i>	Enter the name or group name of MR subtypes to be matched.

Related Commands

accept (in the *User's Reference Manual*)
 assign (in the *User's Reference Manual*)
 create (in the *User's Reference Manual*)
 fcreate (in the *User's Reference Manual*)
 review (in the *User's Reference Manual*)
 spawnmr (in the *User's Reference Manual*)



ES Subcommand (setrel)

The ES relation contains information about external projects with which the Sablime product will communicate. The DBA specifies the method of communication and indicates the state-change information that will be queued automatically for transmission to an external project. Once an external link is established for an MR, the MR state-change information is automatically queued for sending to the external project based on the DBA's decisions.

The state changes that can be communicated result from the following commands: accept, approve, assign, commit, defer, nochange, submit, study, and testpass.

Whenever a command asks for a project or product name, you must enter the name exactly as it is entered in the ES relation. See the *User's Guide* for more information about communication with external projects.

The **ES** screen fields, and the values associated with each, are listed below:

Fields	Values
<i>Function</i> <i>srlfcn</i>	<p>Enter the type of function you want to perform. The allowable entries are:</p> <ul style="list-style-type: none"> n add—add information to communication with an external project n delete—delete information about MR communications with an external project n modify (default)—change an existing external project n view—look at external project information without making any changes. view does not work in the Command Line interface. <p> NOTE: The following information applies when you add a new external project.</p>
<i>External Project</i> <i>esname</i>	<p>Enter the name of the external project with which you want to communicate. In the case of a Sablime project, lowercase must always be used (i.e., sablime). We recommend that lowercase be used for all other project names to avoid confusion. You can enter up to 15 characters.</p>
<i>External Product</i> <i>esprod</i>	<p>Enter the name of the external project with which you want to communicate. In the case of a Sablime project, lowercase must always be used (i.e., sablime). We recommend that lowercase be used for all other project names to avoid confusion. You can enter up to 15 characters.</p>
<i>Remote Machine</i> <i>eshost</i>	<p>Enter the name of the host machine used by the external product. You can enter up to 15 characters.</p>
<i>Network Type</i> <i>esnet</i>	<p>Enter the number for the type of network over which communication will be established. The possible values are shown in the table below. The default is 7.</p> <p> NOTE: IPC is used when the two communicating products are on the same machine and no network is involved.</p>
<i>Remote Program</i> <i>esprog</i>	<p>Enter the name of the program for the external project that will receive messages and put them in the receive queue. You can enter up to 15 characters. If you use UUCP as your network type, leave this field blank, otherwise for Sablime-to-Sablime communication, the program name should be rcv_msgs.</p>



Fields	Values
<i>Parameters esparm</i>	If you use UUCP as the network type, enter the login of the database owner, e.g., sablime (for more information, see Sablime in Chapter 2, <i>Installing Sablime</i>); otherwise, leave this field blank.
<i>Do you want to communicate MR Status Changes esflag</i>	<p>Enter y (default) or n to indicate whether state changes for MRs linked with external MRs should be automatically transmitted to the external product.</p> <p>If you select y, you must enter a y (default) or an n in the rest of the fields on this screen (<i>Accept, Defer, Commit</i>, etc.) to indicate whether a change to that particular state should cause a message to be queued for transmission to the external product.</p> <p> NOTE: The numbered <i>Status</i> fields refer to the five optional test states selected (by MR Class, e.g., <i>preitpass, itpass, prestpass, stpass, and preapprove</i> for the Software Class) in the <i>addgen</i> command.</p>
<i>Do you want to communicate MR Description changes to the originating project stmrnote</i>	<p>Enter y or n (default) to indicate whether <i>mrnote</i> description additions entered on the receiving project should be automatically transmitted to the external project.</p> <p> NOTE: This flag only applies MRs that originated from an external project.</p>
<i>Do you want to communicate MR State at Linkage stsend</i>	Enter y or n (default) to indicate whether the state of MRs at the time of linkage should be automatically transmitted to the external product.


Table 6-4. Network Types

Network	Number
Inter-Process Communication (IPC) and NFS/RFS	6
TCP/IP	7
UUCP	8

PR Subcommand (setrel)

This relation contains information about the products within the Sablime instance. It contains the directory information about the locations of the Sablime databases and commands.

The **PR** screen fields, and the values associated with each, are listed below:

Fields	Values
<i>Function</i> <i>srlfcn</i>	Enter the function you want to perform. The allowable entries are: <ul style="list-style-type: none"> n modify (default)—change the product information for an existing product n view—look at product information without making any changes. <p> NOTE: The following information applies when you modify information for a product.</p>
<i>Product</i> <i>spprod</i>	Enter the name of the product to be modified. The default is the product for which you are set up. You can enter up to 14 characters.
<i>Product Type</i> <i>sptype</i>	Enter internal (default).
<i>Multi-Machine Type</i> <i>sppmm</i>	Enter the mode used by the specified product. The acceptable codes are shown in the table below.
<i>Host Machine</i> <i>sphost</i>	Enter the name of the Host Machine for this product. The left/right scroll buffer has been turned on for this field; you can enter up to 25 characters.
<i>Master Bin</i> <i>spmcb</i>	Enter the full directory path for the location of the Master Control Bin for the product. This is the location where the Sablime commands are stored for this product on the host machine. The default is the directory established for the Master Control Bin when Sablime was installed.


Fields	Values
<i>Active DB</i> <i>spadb</i>	Enter the full directory path for the location of the Active Database (ADB) for the product. The default is the directory established for the ADB when Sablime was installed.
<i>Inactive DB</i> <i>spidb</i>	Enter the full directory path for the location of the Inactive Database (IDB) for the product. The default is the directory established for the IDB when Sablime was installed.
<i>Control DB</i> <i>spadb</i>	Enter the full directory path for the location of the Source Database (SDB) for the product. The default is the directory established for the SDB when Sablime was installed.  NOTE: Never install in or move your product databases to /usr/tmp or /tmp.

Table 6-5. Machine Modes

Mode	Number
Single-Machine Mode	0
Multi-Machine Mode	1

Related Commands

initsab

PRX Subcommand (setrel)

The PRX relation contains information about the products that are participating in a Software Quality Assurance program. The information stored includes the name of the product and the programming language used, customer information, and the name and organization number of the project developing the product.

The **PRX** screen fields, and the values associated with each, are listed below:

Fields	Values
<i>Function srlfcn</i>	<p>Enter the type of function you want to perform. The allowable entries are:</p> <ul style="list-style-type: none"> n modify (default)—change an existing product that will participate in Quality Assurance n view—look at product that will participate in Quality Assurance information without making any changes. <p>The following information applies when you modify information about an existing product that will participate in Quality Assurance.</p>
<i>Product spprxd</i>	Enter the name of an existing product in a Quality Assurance program. The default is the product for which you are set up. You can enter up to 14 characters.
<i>Prog. Language splang</i>	Enter the principal programming language used to develop the product. You can enter up to 15 characters.
<i>First B. Customer spfcust</i>	Enter the name of the first billable customer of the product. You can enter up to 15 characters.
<i>Project Name sppname</i>	Enter the full name of the project developing the product. You can enter up to 50 characters.
<i>Dev. Org. Number sporgnum</i>	Enter the organization number of the project development group. You can enter up to 50 characters.

Related Commands

initsab

source Command

source - change characteristics of a source file.

Synopsis

`sourcekeyword=value [keyword=value...] [-help | -?]`

`*g` = *generic*
`cdir` = *directory*
`*cfile` = *filename*
`delete` = *y|n*
`ndir` = *directory*
`nfile` = *filename*
`nown` = *name*
`nftype` = *type*
`nsq` = *y|n*
`copyto` = *list*
`prompt` = *y|n*



NOTE:

If you want to retain a value, do not enter the parameter associated with it on the command line. You must specify a new value to change the current value.



NOTE:

If a you want to delete a value, at least one space character must be specified in double quotes (e.g., `nown=" "`). This is the only Sablime command that operates this way.

Description

The source command can be used to rename and/or move a file in the Sablime databases, to delete a file from the databases, or to modify the file type (language), file owner, or QA flag information of a file in the databases.



NOTE:

In multi-machine projects, the source command must be run on the host where the databases are located.

The source command can be used to modify or delete only one file at a time. However, it does accept a list of generics in which the operations will be performed on that file. In the case of multiple generics, any request to modify the file information applies to that file as it appears in all the specified generics.

If the requested file is out for edit, no operations are allowed.

Used By

sablime login, SA, any user given permission by the command permissions feature, as follows:

- n Only the *sablime* login is permitted to remove, rename, or relocate files. The SA and other permitted users may change the various file attributes.

Sends Mail To

SA

Field Descriptions



NOTE:

In the Curses Forms interface, if a value is to be retained, simply skip over the field by pressing the RETURN key. If a value is to be nulled out, then at least one space character must be entered.

The fields that appear in the Curses Forms interface, the keywords that may be used in the Command Line interface, and the values associated with each are listed below. Fields preceded by an asterisk (*) must be included on the command line.

Fields and Keywords	Values
* <i>Generics</i> <i>g</i>	The name of one or more comma-separated lists of generics in which the file operation will be performed. You can enter up to 128 characters.
<i>Current Directory</i> <i>cdir</i>	The relative path to the directory in which the file for which the operations will be performed currently resides. You can enter up to 140 characters. Default: the relative path for the setup generic in the user's current directory.
* <i>Current File Name</i> <i>cfile</i>	The current name of the file for which the requested operations will be performed. You can enter up to 12 characters.
<i>Current File Owner</i>	This field is populated automatically and is protected.
<i>Current File Type</i>	This field is populated automatically and is protected.
<i>Current QA Flag</i>	This field is populated automatically and is protected.

Fields and Keywords	Values
<i>Delete</i> delete	y n indicates whether the specified file is to be deleted from the named generics. If this field contains a y , no other actions can be performed on the specified file. Default: n See <i>Restoring Deleted Files</i> in Chapter 4, <i>Other Administrative Procedures</i> , for information about restoring a file that has been deleted accidentally.
<i>New Directory</i> ndir	A valid relative path to move the specified file to a different directory. You can enter up to 140 characters.
<i>New File Name</i> nfile	A new file name. You can enter up to 12 characters. If your UNIX system allows file names longer than 14 characters, the Database Administrator can increase this default using the <code>ftd</code> command.
<i>New File Owner</i> nown	A new file owner name. If a value is to be retained, press the RETURN key or skip this parameter. If a value is to be deleted, at least one space must be entered or specified in double quotes (e.g., <code>nown=" "</code>). This is the only Sablime command that operates this way.
<i>New File Type</i> nftype	A new file type. If a value is to be retained, press the RETURN key or omit this parameter. If a value is to be deleted, at least one space must be entered or specified in double quotes (e.g., <code>nftype=" "</code>). This is the only Sablime command that operates this way.
<i>New QA Flag</i> nsq	y n changes the current QA flag. If you want to retain the value, skip this field or omit this parameter. If the file type is binary, this flag is automatically set to n and protected.
<i>Copy To</i> copyto	The PTS ID, email address, group name, or comma-separated list of PTS IDs, email addresses, or group names of people to be sent copies of mail. You can enter up to 256 characters.
prompt	y n specifies whether prompts should be displayed and input requested. Default: y

Related Commands

dbstop
ftd
snapedit

spacecheck Script

spacecheck - monitor space conditions on the filesystems where the databases are stored.

Synopsis

`spacecheck`

Description



NOTE:

In multi-machine projects, the `spacecheck` command must be run on the host where the databases are located.

For a complete discussion of `spacecheck` and the other the audit commands, see *Managing the Databases* in Chapter 4, *Other Administrative Procedures*

Used By

`sablime login`

Sends Mail To

Sends no mail

sreport Command

sreport - request reports on customer-affecting MRs from the Sablime team.

Synopsis

sreport*keyword=value* [*keyword=value...*] [-help | -?]

prodid = *name*|all
mr = *mr*
sev = 1|2|3|4|all
sys = *system*|all
cmd = *command*|all
type = **modification**|**enhancement**|all
stat = *mrstate*
site = *location*
rel = v5.1|v5.2||v6.0|v6.1
rname = **summary**|**long**
sabaddr = *address*
email = *address*
prompt = *yn*

Description

You can use the `sreport` command to request reports if you can send email from your host machine to Lucent Technologies.

When you request a report, the request is sent to the Sablime development machine. The report output is then mailed to the email address that is specified when the `sreport` command is run. The automatic report process runs periodically, so responses are not immediate.

Two types of report, a summary report and a long report, are available.

Used By

DBA

Sends Mail To

Sends no mail

Field Descriptions

The fields that appear in the Curses Forms interface, the keywords that may be used in the Command Line interface, and the values associated with each are listed below. Fields preceded by an asterisk (*) must be included on the command line.

Fields and Keywords	Values
<i>*Product ID</i> prodid	<i>name</i> all specifies the products to be included in the report. Enter your product ID (default from the PRX relation) to select only MRs originated by your product or all to select all MRs customer-affecting MRs.
<i>MR Number</i> mr	Specifies the MR numbers to be included in the report. You can enter up to 50 characters. If you specify values with this keyword, any values specified with the <i>sev</i> , <i>sys</i> , <i>cmd</i> , <i>type</i> , <i>stat</i> , or <i>site</i> keywords are ignored.
<i>Severity</i> sev	1 2 3 4 all specifies the severities to be included in the report. A menu is supplied. You can specify up to three entries in a comma-separated list.
<i>*System</i> sys	<i>system</i> all specifies the systems to be included in the report. At least one system must be specified if no MRs have been specified. You can specify up to seven entries in a comma-separated list. If more than one system or all or other is specified, any value specified with the <i>cmd</i> keyword is ignored.
<i>Command</i> cmd	Specifies the commands to be included in the report. You can specify up to 50 entries in a comma-separated list. This field is mandatory if you have specified only one system with the <i>sys</i> keyword.
<i>Type</i> type	modification enhancement all specifies the type to be included in the report. Default: enhancement
<i>Status</i> stat	Specify the state of the MRs as selection criteria for the report. You can enter a comma-separated list.
<i>Site</i> site	Specify the site where the MRs originated as selection criteria for the report.
<i>Release Detected</i> rel	v5.1 v5.2 v6.0 v6.1 specifies the Sablime release on which you want information. Default: v6.0
<i>Report Name</i> rname	summary long specifies the report you want to produce. Default: long

Fields and Keywords	Values
<i>*Sablime Email Address</i> sabaddr	Specifies the email address of the Sablime team. Default: sabmail@jaguar.stc.lucent.com
<i>*Your return email</i> email	Specifies the email address of the sender.
<i>Copy To</i> copyto	The PTS ID, email address, group name, or comma-separated list of PTS IDs, email addresses, or group names of people to be sent copies of mail. You can enter up to 256 characters.
prompt	y n specifies whether prompts should be displayed and input requested. Default: y

Related Command

screate

Web_create (in the *User's Reference Manual*)

Web_report (in the *User's Reference Manual*)

Sablime-Level Reserved Groups



This appendix contains a listing of all the Sablime-Level Reserved Groups and their members. It also indicates the standard default value for each group. Notes for each group indicate which commands use the group and what changes can be made to the group.

Table A-1. Sablime-Level Reserved Groups

Group	Members	Notes
__ACCPT	accepted approved assigned deferred fitpassed fstpassed hitpassed hstpassed inspected itpassed nochange preapproved prefitpassed prefstpassed prehitpassed prehstpassed preinspected preitpassed prepublished prestpassed published stpassed submitted understudy ^all none	Used in the <i>Accept MRs with Status</i> field of the addgen command to determine the MR states that are to be accepted into the new generic. Default can be changed. Comments can be added.

Table A-1. Sablime-Level Reserved Groups—Continued

Group	Members	Notes
__ADBREL	ADM CAS COM CP CRIT DEP DOC DOL DS EMG EMR FTD FZ G GRP GRPM GS GT HC MD MG MR MRS MRX MS ORG PDEP PDI SNAP UMS	All valid items used by the query command in the <i>Relation</i> field if active is entered in the <i>Database</i> field. Default and comments can be specified.
__ADEP	file-level line-level none	All valid items used in the <i>Auto Dependency</i> field of the ADM subcommand of the setrel command. Default and comments can be specified.
__ADMIN	addgen closegen dbcross dbdelta dbedit dbhash dbstart dbstop dbxcross	All valid items used in the <i>Command</i> field in the screate command if admin is entered in the <i>System</i> field. Default and comments can be specified.

Table A-1. Sablme-Level Reserved Groups—Continued

Group	Members	Notes
__ADMIN (cont.)	delay ftd hotline.ck initftd initsab mhist mmmail mtrace mvgen net_recv newgen nohist nomail notrace novhelp novinfo novprom pts ptsaudit sablmed sablime.sh sab6.0clean sab6.0conv sab6.0rollbk screate setgroup setnode setperm setrel shell_check sh2x spacecheck sreport xsablme.sh vhelp vinfo vprom	All valid items used in the <i>Command</i> field in the <i>screate</i> command if admin is entered in the <i>System</i> field. Default and comments can be specified.
__AMODE	^internal external	All valid items used in the <i>Access Mode</i> field of the <i>pts</i> command. Default can be changed. Comments can be specified.
__BINARY	yes no auto-detect	All valid items used in the <i>Binary File?</i> field of the <i>addisrc</i> command.

Table A-1. Sablime-Level Reserved Groups—Continued


Group	Members	Notes
__BROFC	mr ^ofc	All valid items used in the <i>Branch</i> field of the <i>addsrc</i> , <i>getversion</i> , and <i>sget</i> commands. Default cannot be changed. Comments can be added.
__CASTYPE	clsCASscls ndcCASndcs rcCASrcs subCASmod sysCASsub ssysCASsub typCASstyp	All valid items used in the <i>Type</i> field of the CAS subcommand of the <i>setrel</i> command. Default can be specified. Comments are supplied and can be changed.
__CLASSPDI	A M	All items available in the <i>Change Class</i> field of the <i>pdi</i> command. A=non-billable, customer-affecting changes intended to correct inoperative conditions or extremely unsatisfactory operating or maintenance conditions. M=internal, non-customer-affecting product changes intended to improve the product or factory process, or to correct the product prior to customer shipment. Default and comments can be specified.
__CSEV	1 2 ^3 4	All valid items used in the <i>Severity</i> field of the <i>create</i> , <i>fcreate</i> , <i>mrgedit</i> , <i>mredit</i> , <i>review</i> , and <i>spawnmr</i> commands. Default can be changed. Comments can be added.
	 CAUTION: <i>Severities are limited to a single alphanumeric character. The query command sorts only single-character severities.</i>	
__DB	active inactive global	All valid items used in the <i>Database</i> field of the <i>dbedit</i> command. Default and comments can be specified.
__DBAI	^active inactive both	All items used in the <i>Database</i> field of the <i>report</i> command. Default can be changed. Comments can be specified.

Table A-1. Sablime-Level Reserved Groups—Continued


Group	Members	Notes
__DELAY	^0 1 2 3 4 5 6 7 8 9	All time intervals (in seconds) used in the <i>Popup Delay</i> field of the <i>pts</i> command. Default can be changed. Comments can be added.
__DEVSEV	1 2 ^3 4	All valid items used in the <i>MRG Severity</i> fields of the <i>assign</i> and <i>accept</i> commands. Default can be changed. Comments can be added.
	 CAUTION: <i>Severities are limited to a single alphanumeric character. The query command sorts only single-character severities.</i>	
__DOCMGMT	template tempset	All valid items used in the <i>Command</i> field in the <i>screate</i> command if docmgmt is entered in the <i>System</i> field. Default and comments can be specified.
__DOCUMENT	Admin_Guide User's_Guide User's_Ref_Manual	All valid items used in the <i>Command</i> field in the <i>screate</i> command if document is entered in the <i>System</i> field. Default and comments can be specified.
__DVCT	SCCS SBCS Either	All valid items used in the <i>Default Version Control Tool for Non-Binary Files</i> field for the <i>ADM</i> subcommand of the <i>setrel</i> command. Default and comments can be specified.
__EMR_RNAME	emr80 emr132 ^complete extract_file emr_html	All valid items used in the <i>Name of Report</i> field of the <i>report</i> command if External_MR is entered in the <i>Class of Report</i> field. Default can be changed. Comments can be added.
__ESNET	^1 6 7 8	All valid items in the <i>Network Type</i> field for the <i>ES</i> subcommand of the <i>setrel</i> command. Default can be changed; comments can be added.

Table A-1. Sablime-Level Reserved Groups—Continued

Group	Members	Notes
__FCNDBA	add delete modify ^view	All valid items in the <i>Function</i> field for the commit, and setrel commands. Default can be changed. Comments can be added.
__FCNTMPL	get ^view	All valid items in the <i>Function</i> field for the template command. Default can be changed. Comments can be added.
__FCNTMPS	add delete copy ^modify	All valid items in the <i>Function</i> field for the tempset command. Default can be changed. Comments can be added.
__FCNSNP	delete edit ^view	All valid items in the <i>Function</i> field for the snapedit command. Default can be changed. Comments can be added.
__FCNTT	delete ^modify	All valid items in the <i>Function</i> field for the testassign command.
__FMST	^header complete	All valid items in the <i>Format</i> field for the listmsgs command. Default can be changed. Comments can be added.
__FTYPE	^1 2 3 4	All valid items in the <i>Field Type</i> field for the ftd command. Default can be changed. Comments can be added.
__GDBREL	ES PR PRX PTS TR	All valid items in the <i>Relation</i> field if global is entered in the <i>Database</i> field of the query command. Default and comments can be specified.

Table A-1. Sablime-Level Reserved Groups—Continued

Group	Members	Notes
__GETVSTAT	submitted preitpassed itpassed prepassed stpassed prehitpassed hitpassed prehstpassed hstpassed prefitpassed fitpassed prefstpassed fstpassed preinspected inspected prepublished published preapproved	All valid items in the <i>MRG Test State</i> field for the <code>getversion</code> command. Default and comments can be specified.
__GRP_RNAME	^summary aggregate extract_file	All valid items in the <i>Name of Report</i> field for the <code>report</code> command if group is entered in the <i>Class of Report</i> field. Default can be changed. Comments can be added.
__GRPTYPE	ptsid mr other	All valid items in the <i>Group Type</i> field for the <code>setgroup</code> and <code>report</code> commands. Default can be specified. Comments can be added.
__HMI	^fs np	All valid entries in the <i>HMI Command Mode</i> field of the <code>pts</code> command.

Table A-1. Sablime-Level Reserved Groups—Continued

Group	Members	Notes
__IDBREL	CAS COM CRIT DEP DOC DOL DS EMG EMR G GS GT HC MD MG MR MRS MRX MS ORG PDEP PDI SNAP	All valid items in the <i>Relation</i> field for the query command if inactive is entered in the <i>Database</i> field. Default and comments can be specified.
__INCLDEP	inmrs inumrs ^no	All valid items in the <i>Include Missing Depended-Upon MRs</i> field for the <i>getversion</i> command.
__INFORET	query report ssql	All valid items in the <i>Command</i> field for the <i>screate</i> command if infoRET is entered in the <i>System</i> field. Default and comments can be specified.
__MM	^0 1	All valid items in the <i>Multi-Machine Type</i> field for the PR subcommand of the <i>setrel</i> command. Default can be changed. Comments can be added.

Table A-1. Sablime-Level Reserved Groups—Continued

Group	Members	Notes
__MR_RNAME	ALL LONG ^SHORT CUSTOM bar bycategory byclass bydeveloper byseverity bysite bystatus bysystem bytype extract_file pie stat	All valid items in the <i>Name of Report</i> field for the report command if MR is entered in the <i>Class of Report</i> field. Default can be changed. Comments can be added.

Table A-1. Sablime-Level Reserved Groups—Continued

Group	Members	Notes
__MRMGMT	accept activate approve assign closemr commit create defer depend fcreate fitpass fstpass hitpass hstpass inspect itpass killmr mredit mrgedit mrnote nochange preapprove prefitpass prefstpass prehitpass prehstpass preinspect preitpass prepublish prestpass propose publish reject spawnmr stpass study submit testassign testpass unaccept	All valid items in the <i>Command</i> field for the <i>screate</i> command if mrmgmt is entered in the <i>System</i> field. Default and comments can be specified.
__MvF_RNAME	^xref extract_file	All valid items in the <i>Name of Report</i> field for the report command if mrVSfile is entered in the <i>Class of Report</i> field. Default can be changed. Comments can be added.

Table A-1. Sablime-Level Reserved Groups—Continued

Group	Members	Notes
__PROMPT	above below ^left right	All valid items in the <i>Prompt Position</i> field for the <i>ftd</i> command. Default can be changed. Comments can be added.
__PTYPE	^internal external	All valid items in the <i>Product Type</i> field for the <i>setrel</i> command. Default can be changed. Comments can be added.
__QMMGMT	hcode ncsl ncsldiff pdi	All valid items in the <i>Command</i> field for the <i>screate</i> command if qmmgmt is entered in the <i>System</i> field. Default and comments can be specified.

Table A-1. Sablime-Level Reserved Groups—Continued

Group	Members	Notes
__QREL	ADM CAS COM CP CRIT DEP DOC DOL DS EMG EMR ES FTD FZ G GRP GRPM GS GT HC MD MG MR MRS MRX MS ORG PDEP PDI PR PRX PTS SNAP TR UMS	All valid items in the <i>Relation</i> field for the query command. Default can be specified. Comments can be added.
__QUEUES	send_queue ^receive_queue	All valid items in the <i>Queue</i> field for the listmsgs command. Default can be changed. Comments can be added.
__RA	assign route	All valid items in the <i>Criteria Type</i> field for the CRIT subcommand of the setrel command and the query command. Default can be changed. Comments can be added.

Table A-1. Sablime-Level Reserved Groups—Continued


Group	Members	Notes
__REPORTCLASS	External_MR ^MR group source mrVSfile	All valid items in the <i>Class of Report</i> field for the report command. Default can be changed. Comments can be added.
__REVACT	enter ^hold link remove quit	All valid items in the <i>Action</i> field for the review command. Default can be changed. Comments can be added.
__RTE	in out	All valid items in the <i>Route</i> field for the report command. Default can be specified. Comments can be added.
__SABREL	v5.0 v5.1 v5.2 ^v6.0	All valid items in the <i>Release</i> field for the sreport command. Default cannot be changed. Comments can be added.
__SABRPN	summary ^long	All valid items in the <i>Report Type</i> field for the sreport command. Default can be changed. Comments can be added.
__SABSEV	1 2 3 4	All valid items in the <i>Severity</i> field for the screate and sreport commands. Default and/or comments can be added.
	 CAUTION: <i>Severities are limited to a single alphanumeric character. The query command sorts only single-character severities.</i>	
__SABSYS	admin docmgmt inforet mrmgmt qrmgmt srcmgmt xmrmgmt usr_manuals Web_Sablime other	All valid items in the <i>Systems</i> field for the screate and sreport commands. Default and/or comments can be added.
__SABTYP	modification ^enhancement	All valid items in the <i>Type</i> field for the screate and sreport commands. Default can be changed. Comments can be added.

Table A-1. Sablime-Level Reserved Groups—Continued

Group	Members	Notes
__SCRATT	^0	All valid items in the <i>Attribute</i> field for the <i>ftd</i> command. Default cannot be changed. Comments can be added.
__SEND	^all group individual	All valid items in the <i>Message Selection</i> field for the <i>sendmsgs</i> command. Default can be changed. Comments can be added.
__SETREL	ADM CAS CP CRIT ES PR PRX	All valid items in the <i>Relation</i> field for the <i>setrel</i> command. Default can be specified. Comments are supplied and can be changed.
__SRCMGMT	SabMerge addsrc addisrc common edget edput getversion node_update node_validate primsdb reserve setnode sget smerge snapedit source srrepr uncommon unedget unedput unreserve	All valid items used in the <i>Command</i> field in the <i>screate</i> command if srcmgmt is entered in the <i>System</i> field. Default and comments can be specified.
__SRC_RNAME	^edgotten extract_file	All valid items in the <i>Name of Report</i> field for the <i>report</i> command if source is entered in the <i>Class of Report</i> field. Default can be changed. Comments can be added.

Table A-1. Sablime-Level Reserved Groups—Continued


Group	Members	Notes
__SRPSTAT	created killed accepted deferred understudy nochange assigned submitted preitpassed prestpassed stpassed approved closed	All valid items used in the <i>Status</i> field for the sreport command. Sablime uses these states for the MR life cycle.
__STSEV	1 2 ^3 4	All valid items in the <i>MRG Severity</i> field for the study command. Default can be changed. Comments can be added.
	 CAUTION: <i>Severities are limited to a single alphanumeric character. The query command sorts only single-character severities.</i>	
__TLEVEL	1 2 3 4 5	Test level 1 (includes <i>preitpassed</i> , <i>fpreitpassed</i> , <i>hpreitpassed</i> , <i>inspected</i>) Test level 2 (e.g., <i>itpassed</i>) Test level 3 (e.g., <i>prestpassed</i>) Test level 4 (e.g., <i>stpassed</i>) Test level 5 (e.g., <i>preapproved</i>)
__VCT	SCCS SBCS	All valid items in the <i>Control Tool</i> field for the addisrc command.
__XMRMGMT	listmsgs qmr rcv_msgs review sabmldm sabrepedm sendmsgs	All valid items used in the <i>Command</i> field for the screate command if xmrmgmt is entered in the <i>System</i> field. Default and comments can be specified.

Table A-1. Sablme-Level Reserved Groups—Continued

Group	Members	Notes
__WEB_SABLIME	home_page myMRs Browse_MRs Browse_Files settings HelpScreens MR_Details File_History File_View SabMerge create accept fcreate spawnmr unaccept study propose defer nochange activate killmr assign submit testassign testpass reject approve closemr mrnote depend addisrc addgsrc common edget edput smerge uncommon unedget unedput sget getversion setnode report query setgroup qmr	All valid items used in the Command field for the screate command if Web_Sablme is entered in the System field. Default and comments can be specified.

Table A-1. Sablime-Level Reserved Groups—Continued

Group	Members	Notes
__YN	y n	All valid responses for <i>MR Class</i> fields in the <i>addgen</i> command. Default cannot be specified. Comments can be added.
__YNALL	^y n all	All valid responses for the <i>Reassign Flag</i> field in the <i>setrel</i> command. Default cannot be changed. Comments can be added.
__YNN	y ^n	All valid responses for yes/no fields in various commands. Default cannot be changed. Comments can be added.
__YNO	^y n o	All valid responses for the <i>Licensed</i> field in the <i>pts</i> command.
__YNQN	y ^n q	All valid responses to the standard confirm field on all Curses Forms interface screens.
__YNY	^y n	All valid responses for yes/no fields in various commands. Default cannot be changed. Comments can be added.

Product-Level Reserved Groups

B

This appendix contains a listing of all the Product-Level Reserved Groups and their members. It also indicates the standard default value for each group. Notes for each group indicate which commands use the group and what changes can be made to the group

Table B-1. Product-Level Reserved Groups

Group	Members	Notes
_ATYPE	enhancement initialization ^modification	All valid items in the <i>MR Type</i> field for the accept, fcreate, mrgedit, and spawnmr commands to qualify an MR when it is accepted.
_CMRCS	lack_of_communication miscommunication misapplication	All valid items in the <i>RC Subcat</i> field for the submit command if communication is entered in the <i>Root Cause</i> field.
_DEFCODE	enhancement ^other	All valid items in the <i>Reason Code</i> field for the defer command.
_DOCUCLASS	figure index appendix text organization table_of_contents	All valid items in the <i>MR Subclass</i> field for the accept command if document is entered in the <i>MR Class</i> field.
_DOC_FLTYPE	document ascii data Frame	All valid items for the <i>File Type</i> prompt in primsdB for document MRs.
_ENHTYPE	conformity(sw) efficiency(sw) installability(sw) manufacturability(sw) operability(hw) portability(sw) producibility(hw) readability(sw) testability(sw) usability(sw)	All valid items in the <i>MR Subtype</i> field for the accept command if enhancement is entered in the <i>MR Type</i> field.
_ETRCS	nonexistent not_detailed_enough out_of_date misleading	All valid items in the <i>RC Subcat</i> field for the submit command if education_and_training is entered in the <i>Root Cause</i> field.
_FAVED	ed ^vi emacs	All valid items in the <i>Favorite Editor</i> field for the pts command.

Table B-1. Product-Level Reserved Groups—Continued

Group	Members	Notes
_FIRMCLASS	screen_handling data_validation database_manipulation mainline_processing error_handling algorithm	All valid items in the <i>MR Subclass</i> field for the accept command if firmware is entered in the <i>MR Class</i> field.
_FIRM_FLTYPE	c	All valid items for the <i>File Type</i> prompt in primsdB for firmware MRs.
_HARDCLASS	electrical physical	All valid items in the <i>MR Subclass</i> field for the accept command if hardware is entered in the <i>MR Class</i> field.
_HARD_FLTYPE	c	All valid items for the <i>File Type</i> prompt in primsdB for hardware MRs.
_ICAT	inspection review code_reading usage ^testing document_reading demo discussion field_enh field_mod maintenance new_feature	All valid items in the <i>Category</i> field for the create command.
_INITYPE	new_source reused_source	All valid items in the <i>MR Subtype</i> field for the accept command if initialization is entered in the <i>MR Type</i> field.
_KILLCODE	duplicate meaningless ^other	All valid items in the <i>Reason Code</i> field for the killmr command.

Table B-1. Product-Level Reserved Groups—Continued

Group	Members	Notes
_LOC	AK AL ALC AN CB CH FJ HL HO HOH HR IH IW IX LC LZ MH MO MT MV NK NP PK PY RD SF WB WH WI WV	All valid items in the <i>Loc Code</i> field for the <i>pts</i> command. The maximum length of this field is 25 characters.
_MODTYPE	conformity(sw) efficiency(sw) installability(sw) manufacturability(sw) operability(hw) portability(sw) producibility(hw) readability(sw) testability(sw) usability(sw)	All valid items in the <i>MR Subtype</i> field for the <i>accept</i> command if modification is entered in the <i>MR Type</i> field.

Table B-1. Product-Level Reserved Groups—Continued

Group	Members	Notes
_NOCHCODE	duplicate not_applicable ^other unnecessary wrong_generic	All valid items in the Reason Code field for the nochange command.
_ODPGRP	architecture design inspection development unit_test integration_test system_test	All valid items in the <i>Optimal Detection Phase</i> field for the submit and mrgedit commands.
_OSRCS	initialization_error editing_error forgot	All valid items in the <i>RC Subcat</i> field for the submit command if oversight is entered in the <i>Root Cause</i> field.
_PDGRP	requirements architecture design development ^unit_test integration_test system_test beta_trial controlled_intro general_availability	All valid items in the <i>Phase Detected</i> field for the create, fcreate, mredit, and review commands.
_PDRCS	lack_of_documentation insufficient_details	All valid items in the <i>RC Subcat</i> field for the submit command if project_documentation is entered in the <i>Root Cause</i> field.
_PIGRP	requirements architecture design code	All valid items in the <i>Phase Introduced</i> field for the submit and mrgedit commands.
_PMRCS	lack_of_standards lack_of_consistency lack_of_templates lack_of_documentation	All valid items in the <i>RC Subcat</i> field for the submit command if project_methodology is entered in the <i>Root Cause</i> field.

Table B-1. Product-Level Reserved Groups—Continued

Group	Members	Notes
_RCGRP	resources_and_planning education_and_training project_methodology project_documentation communication oversight	All valid items in the <i>Root Cause</i> field for the submit and mrgedit commands.
_REJCODE	non_compile no_documentation bad_problem_desc incomplete bad_resolution code_faults ^other	All valid items in the <i>Reject Code</i> field for the reject command.
_RELS	^not_applicable	All valid items in the <i>Release Detected</i> field for the create, fcreate, mredit, review, and spawnmr commands, and in the <i>Release Introduced</i> field for the submit command. This field should be customized for your product. .
_RESCODE	^as_proposed other	All valid items in the <i>Resolution Code</i> field for the submit command.
_RPRCS	not_assigned low_priority new_developer methodology_not\ enforced poor_documentation no_documentation	All valid items in the <i>RC Subcat</i> field for the submit command if resources_and_planning is entered in the <i>Root Cause</i> field.
_SABSTAT	accepted approved assigned closed fitpassed fstpassed itpassed preapproved prefitpassed prefstpassed preitpassed prestpassed ^stpassed submitted	All valid items in the <i>System Test State</i> and <i>Release State</i> fields for the sqin command.

Table B-1. Product-Level Reserved Groups—Continued

Group	Members	Notes
_SITES	^not_applicable	All valid items in the <i>Site</i> field for the create, fcreate, review, and mredit commands.
_SOFTCLASS	screen_handling data_validation database_manipulation mainline_processing error_handling algorithm	All valid items in the <i>MR Subclass</i> field for the accept command if software is entered in the <i>MR Class</i> field.
_SOFT_FLTYPE	LR bal c c++ cobol comp dbd dirjcl dtp fortran html jcl limbo lke makefile mll parms perl pli sal shell snowflake spitbol binary other vb	All valid items for the <i>File Type</i> prompt in primsdB for software MRs.
_SYS	^none other all	All valid items in the <i>System</i> field for the create, fcreate, mredit, review, and spawnmr commands. This field should be customized for your product.

Initialization Program Diagnostic Messages

C

This appendix contains all the diagnostic message numbers generated by the Sablime initialization program, the corresponding DBA warning messages, and suggestions for troubleshooting. The diagnostic messages are provided to help Sablime users fix problems on their own. However, if a call to the Sablime Hotline is necessary, the diagnostic information can also be used by a member of the Sablime team to pinpoint the problem.

Organization

There are three types of diagnostic messages described in this appendix. These include IProgram diagnostic messages, Uid diagnostic messages, and Other diagnostic messages.

Each section is organized by message number. For each error message number, both the main portion of its dbawarn message and troubleshooting advice are given.

IProgram Diagnostic Messages

The error messages are usually displayed on the user's screen in the following format:

```
*SYS_ERR*      Message Number [90xx] From: IProgram.c [Version 1.1.1.17]
                Program Initialization Error 90xx (See Sablime Administrator's Manual.
                Please Notify Your Sablime System Administrator Immediately.
```

Users can find more information about an error by looking at the dbawarn file, which is located in the Global Database under the tmp directory. The file name is formed by concatenating the product name and the string _dbawarn, e.g., \$sabGDB/tmp/nmake_dbawarn.

The format of the dbawarn message is:

```
xx/xx/xx xx:xx:xx [login_id] command @(#) Program name \
[Version x.x.x.x] - dba_msg: "Warning Message" has \
failed as request by @(#)IProgram.c [Version x.x.x.x]
```

The following conventions are used throughout this appendix:

[adb_dir] mentioned in message text is the full path of the Sablime Active Database.

[idb_dir] mentioned in message text is the full path of the Sablime Inactive Database.

[gdb_dir] mentioned in message text is the full path of the Sablime Global Database.

[sdb_dir] mentioned in message text is the full path of the Sablime Source Database.

The most frequently seen error messages are 9001–9014, which are user-type error messages, for example, a Sablime environment variable was not set. Usually, this type of error requires little effort to fix. Error messages 9050–9090 are system errors where in most cases, the user would have to obtain help from the Sablime Administrator, the system administrator, or the Sablime hotline in order to solve the problem.

Some of the messages in this section refer to the Sablime environment variables, which are of the form sabNAME, e.g., sabGEN, sabGDB.

9001

Call to UNIX library function [putenv(2)] has failed,
Call to UNIX library function [uname(2)] has failed,
Call to UNIX library function [getpwent(3C)] has failed, or
Call to UNIX library function [getpwuid(3C)] has failed

9002

Required Sablime Environment Variable [sabHOST] is not set in your environment
Required Sablime Environment Variable [sabNET] is not set in your environment,
Required Sablime Environment Variable [TERM] is not set in your environment,
Required Sablime Environment Variable [sabGDB] is not set in your environment,
Required Sablime Environment Variable [sabPROD] is not set in your environment,
Required Sablime Environment Variable [sabGEN] is not set in your environment or
Required Sablime Environment Variable [sabDIFF] is not set in your environment

Troubleshooting/corrective action:

Check the values of the Sablime variables, e.g., `grep sabHOST $sabVAR`, `echo $sabGEN`.

Set the environment variables to the correct values.

Update your `xsablime.sh` shell script if necessary.

9003

Required Sablime Environment Variable [TERM] is set in your environment, but it is null,
Required Sablime Environment Variable [sabGDB] is set in your environment, but it is null,
Required Sablime Environment Variable [sabGEN] is set in your environment, but it is null

Troubleshooting/corrective action:

Set the environment variables to the correct values.

Update your `xsablime.sh` shell script if necessary.

9004

ADM Relation [adb_dir/ADM] tuple record is not unique; [x] tuple records found

Troubleshooting/corrective action:

You may have a corrupted ADM or PR relation in the Sablime database. You can check your database by doing the following:

- n Change directory to your PR relation, i.e., `cd $sabGDB/PR`.

- n Check that the sixth field of the specified product's PR tuple file contains the correct active database path.
- n Check the adb_dir/ADM relation. There should be only one tuple file; that tuple file should have only one record.

9005

Required Sablime Variable [sabNET] contains an invalid network code [netcode].

90076

Required Sablime Variable [sabNET] contains a network code [netcode] indicating 'no network' - but you are on a satellite machine [name] with host [name].

9007

**PATH [xxxxx] as found in the [sabGDB] Variable is not to an accessible directory; error[x],
PATH [xxxxx] as found in the [sabMCB] Variable is not to an accessible directory; error[x]
PATH [xxxxx] as found in the [sabLCB] Variable is not to an accessible directory; error[x]
PATH [xxxxx] as found in the [sabVHELP] Variable is not to an accessible directory; error[x]**

Troubleshooting/corrective action:

Check that the Sablime environment variable is set to the proper value.

Check that the specified directory exists.

Update your xsablime.sh program or \$sabVAR file to reflect the changes.

9008

**Product [xxxxx] is an internal product but user's PTS ID is NOT internal
Path [xxxx] as found in the [sabSPOOL] Variable does not exist or does not have permission
777; error [x].
Path [xxxx] as found in the [sabTMP] Variable does not exist or does not have permission 777;
error [x].**

Troubleshooting/corrective action:

If you should have internal access to this product, the Sablime Administrator can modify your PTS record to internal access mode.

9009

Required Sablime Variable [HOST=xxx] does NOT match <CR> field #4 [HostName=xxx] in the PR Relation record <CR> of the [xxxxx] Product

Troubleshooting/corrective action:

Check that both the fourth field of the PR relation tuple file of the specified product and the value of the sabHOST variable in the sabVAR file are set correctly. These items should contain the same information.

If the PR relation must be updated, set up for a different product; then run setrel. If no other product is available, call the Sablime hotline.

Update the xsablime.sh shell script, if necessary.

9010

Required Sablime Variable [MCB=/xx/xx] does NOT match field #3 [MasterCommandBin=/xx/xx] in the PR Relation record of the [xxxxx] Product

Troubleshooting/corrective action:

Check that both the third field of the PR relation tuple file of the specified product and the value of the sabMCB variable in the \$sabVAR file are set correctly. They should contain the same information.

If the PR relation must be updated, set up for a different generic; then run setrel. If no other generic is available, call the Sablime hotline.

Update the xsablime.sh shell script.

9011

Field #3 [Multi-Machine Flag] in the GDB's [PR] Relation tuple record for the [xxxxx] product is empty,

Field #6 [Full Path to Active Data Base] in the GDB's [PR] Relation tuple record for the [xxxxx] product is empty,

Field #7 [Full Path to Inactive Data Base] in the GDB's [PR] Relation tuple record for the [xxxxx] product is empty, or

Field #8 [Full Path to Source Code Data Base] in the GDB's [PR] Relation tuple record for the [xxxxx] product is empty,

Troubleshooting/corrective action:

If the PR relation must be updated, set up for a different generic; then run setrel. If no other generic is available, call the Sablime hotline.

9012

Field #3 [Multi-Machine Flag] in the GDB's [PR] Relation tuple record for the [xxxxx] product is too long; maximum expected size is [1] characters,
Field #6 [Full Path to Active Data Base] in the GDB's [PR] Relation tuple record for the [xxxxx] product is too long; maximum expected size is [131] characters,
Field #7 [Full Path to Inactive Data Base] in the GDB's [PR] Relation tuple record for the [xxxxx] product is too long; maximum expected size is [131] characters, or
Field #8 [Full Path to Source Code Data Base] in the GDB's [PR] Relation tuple record for the [xxxxx] product is too long; maximum expected size is [131] characters,

Troubleshooting/corrective action::

Change the value to be within the length specified.

9013

Field #3 [Multi-Machine Flag] in the GDB's [PR] Relation tuple record for the [xxxxx] product must be '0' [Host Only] or '1' [Multi-Machine] only

Troubleshooting/corrective action:

Change the value to be '0' or '1', whichever is appropriate for your installation.

9014

You are on a Satellite Processor [machine_name] not set for Multi-Machine Sablime; Field [x] is set to '0'

Troubleshooting/corrective action:

Check that the values of \$sabNET and the sabHOST variable in the \$sabVAR file are set correctly.

Check the third field of your PR relation tuple file. Set this field to 1 if multi-machine environment; otherwise, set to 0.

9015

User is setup for product [xxxxx], but the allowed products for the user's PTSid are [xxx,xxx,xxx], or
You are NOT authorized to access [xxxxx] product

Troubleshooting/corrective action:

If you should be authorized to access this product, the DBA can run the pts command to authorize you to access this product.

9015

Path [adb_dir] as found in Field #6 [Full Path to Active Data Base] of the [xxxxx] Product PR Relation record in the GDB's [PR] Relation tuple record for the [failcode],
Path [idb_dir] as found in Field #7 [Full Path to Inactive Data Base] of the [xxxxx] Product PR Relation record in the GDB's [PR] Relation tuple record for the [failcode], or
Path [sdb_dir] as found in Field #8 [Full Path to Source Code Data Base] of the [xxxxx] Product PR Relation record in the GDB's [PR] Relation tuple record for the [failcode]

Troubleshooting/corrective action:

Check the PR relation tuple file of the specified product. The sixth, seventh, and eighth fields of the tuple file contain the full paths of the Active Database, Inactive Database, and Source Database.

Check that the specified paths are existing directories with permission 755.

9017

Path [PATH] as found in the [sabVAR] Environment Variable is not to an accessible file; error [fail code].

Troubleshooting/corrective action:

Make sure that the sabVAR variable in the sablime.sh script points to the correct file.

9018

Cannot open variables file [FILE] in mode [r].

Troubleshooting/corrective action:

Change permissions on the \$sabVAR file to 644.

9019

Required Sablime variable [sabMCB] is not defined.
Required Sablime variable [sabHOST] is not defined
Required Sablime variable [sabNET] is not defined
Required Sablime variable [sabSERV] is not defined
Required Sablime variable [sabVHELP] is not defined
Required Sablime variable [sabDIRF] is not defined
Required Sablime variable [sabLCB] is not defined

Troubleshooting/corrective action:

Make sure that the sabMCB variable in the \$sabVAR file is set to the machine name where the Sablime databases reside.

9021

Required Sablime variable [sabLCB] is too long; maximum size is [131] characters.
Required Sablime variable [sabMCB] is too long; maximum size is [131] characters.
Required Sablime variable [sabSERV] is too long; maximum size is [20] characters.

Troubleshooting/corrective action:

Make sure that the full path is less than the indicated number of characters long.

9022

Cannot close variables file [FILE].

Troubleshooting/corrective action:

Call the Sablime hotline.

9023

Path [xxx] as found in the sabNCSL variable is not an accessible file; error [x].
Path [xxx] as found in the sabDIRF variable is not an accessible file; error [x].
Path [xxx] as found in the sabDIFF variable is not an accessible file; error [x].
Path [xxx] as found in the sabMFR variable is not an accessible file; error [x].

Troubleshooting/corrective action:

One of the Sablime variables is set to a directory rather than a file. Reset the variable to the correct executable.

9024

Path [DIRECTORY] for [RELATION] relation is not to an accessible directory.

Troubleshooting/corrective action:

Run hotline.ck.

Check the sabGDB variable in xsablime.sh. Check the PR relation entries for the ADB, IDB, SDB, and MCB.

9025

[sabSERV=/dev/null] is not a valid setup when [sabNET=xx].

Troubleshooting/corrective action:

Set the environment variable to the correct service (such as net_recv).

Update the xsablime.sh shell script, if necessary.

9051

openfile (/usr/tmp/Sablimedbugxxxx) for write failed

Troubleshooting/corrective action:

Check the write permission of the /usr/tmp directory by using
ls -ld /usr/tmp.

Check the available space left in /usr/tmp by using
df /usr/tmp.

Check the write permission of the specified file if it already exists.

9052

openfile (/usr/tmp/Sablimestatxxxx) for write failed or
openfile (/usr/tmp/Sablimesqxxxx) for write failed

Troubleshooting/corrective action:

Check the write permission of the /usr/tmp directory by using
ls -ld /usr/tmp.

Check the available space left in /usr/tmp by using
df /usr/tmp.

Check the write permission of the specified file if it already exists.

9053

Can't get initial value for hangup signal as requested by [sigsinit.c]; errno [x],
Can't get initial value for interrupt signal as requested by [sigsinit.c]; errno [x],
Can't get initial value for quit signal as requested by [sigsinit.c]; errno [x], or
Can't get initial value for softkill signal as requested by [sigsinit.c]; errno [x]

Troubleshooting/corrective action:

Cannot ignore the SIGHUP, SIGINT, SIGQUIT, or SIGTERM signal. Call the Sablime hotline to resolve the problem.

9054

Can't turn on hangup [for InterruptExit()] as requested by [IProgram.c]; errno [x],
Can't turn on interrupt [for InterruptExit()] as requested by [IProgram.c]; errno [x],
Can't turn on quit [for InterruptExit()] as requested by [IProgram.c]; errno [x], or
Can't turn on softkill [for InterruptExit()] as requested by [IProgram.c]; errno [x]

Troubleshooting/corrective action:

Cannot catch SIGHUP, SIGINT, SIGQUIT, or SIGTERM signal. Call the Sablime hotline to resolve the problem.

9055

Call to 'new' has failed to obtain requested space

Troubleshooting/corrective action:

The processor is running out of memory.

9056

Call to UNIX library function [uname(2)] returns NULL as nodename

Troubleshooting/corrective action:

Call the Sablime hotline to resolve the problem.

9057

Cannot communicate with host [name] machine.

Troubleshooting/corrective action:

This message implies that your Sablime communication link was not set properly. First, check that the Sablime variables such as sabHOST and sabSERV in the \$sabVAR file and \$sabNET are set correctly. Secondly,

- n In case of TCP/IP:
 - Check if the Sablime daemon, sablimed, was running on the host machine for Sablime-to-Sablime communication; sablimed should run on both host machines.

9060

Missing a valid Sablime PTSid for [machine!id]; contact your Sablime Administrator,
You do not have a valid Sablime PTSid; contact your Sablime Administrator, or
Expected to break up tuple record [xx!xx;xxx;xxx;xx;xx;xx;y,n,y,0;] into [20] fields, found [8]

Troubleshooting/corrective action:

If you do not have a PTS ID, ask your Sablime administrator to add a PTS ID for you.

If you got the "Expected to break up tuple ..." dbawarn message, that implies that you have a corrupted PTS relation in the Sablime database.

9062

Call to 'new' has failed to obtain requested space

Troubleshooting/corrective action:

The processor is running out of memory.

9063

Tuple record with primary key(s) not in tuple [gdb_dir/PR/xx].
Expected to break up tuple record [xx;internal;x;xx;xx/bin;adb;idb;sdb] into [12] fields, found [8].

Troubleshooting/corrective action:

The PR relation tuple file is corrupted or the PR relation tuple file for your product is missing.

9066

Call to 'new' has failed to obtain requested space

Troubleshooting/corrective action:

The processor is running out of memory.

9067

***SYS_ERR* Message Number [9067] From: IProgram.c [Version 4.1.1.2].**
Expected to break up tuple record into [20] fields, found [1]. Please check the dbawarn file for more information.
Please Notify Your Sablime System Administrator Immediately.

Troubleshooting/corrective action:

Examine the dbawarn file to determine the relation/tuple file that has the corruption and try to fix it by deleting any blank lines or by fixing the records that contain wrong number of fields.

9069

Call to 'new' has failed to obtain requested space

Troubleshooting/corrective action:

The processor is running out of memory.

9071

The named file [adb_dir/product/FILES/SAB.stopfile] does not exist; 'stat' errno [2] on file [adb_dir/product/FILES/SAB.stopfile]

Troubleshooting/corrective action:

The SAB.stopfile does not exist. Create an empty file called SAB.stopfile and place it in the adb_dir/product/FILES directory with permissions set to 644.

9072

Can't fopen(3) file [adb_dir/product/FILES/SAB.stopfile] in mode [r]

Troubleshooting/corrective action:

The permission of adb_dir/product/FILES/SAB.stopfile file should be set to 644.

9073

Call to 'new' has failed to obtain requested space

Troubleshooting/corrective action:

Computer system is running out of memory space.

9074

pl() overflow; parsing [command;sub;y,n,n;xxx: ,xx,,999;x,x,;;x,x,_,x,xx,xx:x,x;] into an array of size [7] with separator [;],

Tuple record with primary key(s) not in tuple [/usr1/sablime/adb/tst+/FTD/oa], or Expected to break up tuple record [command;sub;y,n,n;] into [7] fields, found [4].

Troubleshooting/corrective action:

Corrupted FTD data. A database field separator has either been added or deleted from the FTD record. Compare against the original record found in the `ftd_data` file in the `sablime` bin

9075

Call to 'new' has failed to obtain requested space

Troubleshooting/corrective action:

The processor is running out of memory.

9076

Call to 'new' has failed to obtain requested space

Troubleshooting/corrective action:

The processor is running out of memory.

9077

Expected to break up string [string] into [3] fields, found [0].

Troubleshooting/corrective action::

Corrupted FTD relation tuple file. The third field (`ftd_flag`) should have three comma-separated fields.

9078

Expected to break up string [string] into [4] fields, found [0].

Troubleshooting/corrective action:

Corrupted FTD relation tuple file. The fourth field (`ftd_text_flag`) should have four comma-separated fields.

9089

Expected to break up string [xx,xx,xx,] into [3] fields, found [x].

Troubleshooting/corrective action:

Corrupted FTD relation tuple file. The fifth field (ftd_value_flag) should have three comma-separated fields.

9090

Expected to break up string [xx,xx,xx,xx,xx] into [8] fields, found [x].

Troubleshooting/corrective action:

Corrupted FTD relation tuple file. The seventh field (ftd_hmi_flag) should have eight comma-separated fields.

9500

Unable to 'mkdir' MailSpool directory [product] for mail processing.
Unable to create [directory]. No search permissions OR wrong directory path.

Troubleshooting/corrective action::

Permissions on the MailSpool and/or MailSpool/*product* directory are not 777;
change permissions. Also, verify that there is adequate disk space.

Check the value of \$sabSPOOL; if sabSPOOL is defined, make sure that the
sablime login can write to that directory. If Sablime cannot write to \$sabSPOOL, the
user must create the MailSpool/*product* directory.

9501

Error in mail directory structure, [directory] is not a directory.

Troubleshooting/corrective action:

This error occurs when the command is trying to create any of the following
directories: MailSpool or MailSpool/*product* or MailSpool/*product/server_name*. If a
file exists by one of these names, the command does not remove the file. The
Sablime Administrator must remove the file.

Uid Diagnostic Messages

This section describes seven Sablime diagnostic messages related to the license key file. The format of the messages is:

Uid error code x: Please call the Sablime Hotline

4

Uid error code 4: Please call the Sablime Hotline

Troubleshooting/corrective action:

The Sablime key file, `.usrld`, is missing from the MCB directory. Call the Sablime hotline to resolve this problem.

5

Uid error code 5: Please call the Sablime Hotline

Troubleshooting/corrective action:

You have wrong Sablime key file for your machine. Call the Sablime hotline to resolve this problem.

6

Uid error code 6: Please call the Sablime Hotline

Troubleshooting/corrective action:

Call the Sablime hotline to resolve this problem.

7

Uid error code 7: Please call the Sablime Hotline

Troubleshooting/corrective action:

Check that `$sabNET` set correctly. Set `sabNET=5` only if you use NFS/RFS networking.

14

Uid error code 14: Please call the Sablime Hotline

Troubleshooting/corrective action:

The Sablime key file, .usrld, is missing from the LCB directory. Make a copy of the .usrld file from the MCB directory and place it in the LCB directory.

15

Uid error code 15: Please call the Sablime Hotline

Troubleshooting/corrective action:

You have the wrong Sablime key file in the LCB directory. Make a copy of the .usrld file from the MCB directory and place it in the LCB directory.

16

Uid error code 16: Your Sablime license expired [xxx] days ago

Troubleshooting/corrective action:

Call the Sablime hotline to resolve this problem.

Other Diagnostic Messages

This section includes the most frequently seen error messages from the satellite machine. The format of the messages is:

SYS_ERR **Message Number [xxx] From: IProgram.c [Version 1.1.1.17]**
(xxxxxxxxxx diagnostic message here xxxxxxxxxxxx)
Please Notify Your Sablime System Administrator Immediately.

0

cEOF.

Troubleshooting/corrective action:

- n Set the permission of MCB/net_recv to 4755.
- n Check Sablime variables on satellite machine. If all variables are set properly, call the Sablime hotline to resolve the problem.
- n Call the Sablime hotline to resolve the problem.

999

Network read channel [-77] failed; errno [9].

Troubleshooting/corrective action:

- n Run dk host_name.authorize to authorize yourself to the host machine.
- n Make sure that the sabSERV variable in the \$sabVAR file is set to net_recv if sabNET=7 (for TCP/IP).

999

Network write to channel [3] has failed; errno [5].

Troubleshooting/corrective action:

Set the permission of MCB/net_recv to 4755.

Make sure that the dksrvtab entry for the MCB is spelled correctly.

999

Network read [x] bytes; expected [x] bytes

Troubleshooting/corrective action:

Check the Sablime environment variables on satellite machine. If all variables are set properly, call the Sablime hotline to resolve the problem.

6951

Can't [fclose(3)] file on fd [1090880]; errno [28].

Troubleshooting/corrective action:

Make sure that there is enough space on the file system where the databases reside.

6952

Can't fopen(3) file in mode {a}.

Troubleshooting/corrective action:

Set the permission of the \$sabGDB/tmp directory to 777. Make sure the command permissions on the host machine are set to 4755 and the command permissions on the satellite machine are set to 755. If an NFS/RFS network is being used, the command permissions should be set to 4755 on the satellite machine. Run hotline.ck.

The 'initftd' has failed Cleaning Up Product Specific Directories & Files Please Stand By!

Troubleshooting/corrective action:

A message is displayed prior to this message. Note the previous message and then call the Sablime hotline to resolve the problem.

Audit Program Messages

D

This appendix contains all the messages generated by the Sablime audit programs (dbcross, dbxcross, dbdelta, hotline.ck, and spacecheck) and tells the administrator how to respond to each of them.

Messages from dbcross

Table D-1 shows the messages generated by the audit program dbcross and the appropriate response to each. All reported errors are from the Active Database unless otherwise indicated.

Table D-1. dbcross Messages

dbcross Message	Response
\$_ not in environment and not found in \\${sabVAR} file.	Variable must be set and exported.
ERROR: Global database [\${sabGDB}] is not a directory.	Check that sabGDB is set to proper location.
ERROR: Sablime stop file [\${sabADB}/FILES/SAB.stopfile] is not a regular file.	Make sure SAB.stopfile is a valid and readable file.
WARNING -- Sablime is stopped for the following reason(s):	No action is necessary. just informing user of the fact.
The following file names are invalid within the relations (RELATION/file) of the [ADB/IDB/GDB]:	Remove all invalid tuple files from relations in that DB.
The following relation directories are missing:	Create the missing relations.
The following are not regular files in the relations (RELATION/file) of the [ADB/IDB/GDB]:	Files listed are directories or zero-length. Make them into regular text files.

Table D-1. dbcross Messages—Continued

dbcross Message	Response
The following file names are invalid within the relations (RELATION/file) of the [ADB/IDB/GDB]:	Remove all invalid tuple files from the relations.
The following lock files should be removed from DBLOCK (RELATION/file) in the [ADB/IDB/GDB] [filepath]:	Remove the named lock files.
The following lock files were found in DBLOCK (RELATION/file) in the [ADB/IDB/GDB] [filepath]:	You did not stop the database before running the audits. Other problems that are incorrect might be found because you are running the audits this way.
The following keys are duplicated in [INACTIVE.]\$rel: field[1]; .. ;field[n] (\$stup)	Verify relation tuple files and remove duplicate entries.
The following keys are duplicated between \$rel and INACTIVE.\$rel: field[1]; .. ;field[n] (\$stup)	Verify relation tuple files and remove duplicate entries.
File System and Inode Space Check	
Execution of the 'spacecheck' script has failed due to the following variables being unset: <Variable List>.	Edit spacecheck and set these variables according to the description in <i>Checking for Disk Space</i> .
G vs. G	
The following GENERIC GID combinations from [INACTIVE] G have invalid GID:	Review the GID allocations in the G relation to determine the correct GID setting for this G.
MR vs MR, ORG vs MR, MR vs ORG, FILES vs MR	
The following MR MRSTATUS combinations from [INACTIVE.]MR have invalid MRSTATUS:	Verify other relation tuple files and change any invalid MRSTATUS to one of the following: <i>created</i> , <i>active</i> , <i>mra_deferred</i> , or <i>mra_study</i> .
The following MR's for which spawned MR's exist are missing in [INACTIVE.]MR:	Verify the MRS relation record and either add record to MR and/or ORG relation or delete record from MRS relation.
The following MR names from [INACTIVE.]ORG are not in [INACTIVE.]MR:	Verify MR records and either add record to MR relation or remove from ORG relation.
The following MR names from [INACTIVE.]MR are not in [INACTIVE.]ORG:	Verify MR records and either add record to ORG relation or remove from MR relation.
The following MR names from [INACTIVE.]FILES/description are not in [INACTIVE.]MR	Save description file and recreate MR using saved description file.
The following MR names from [INACTIVE.]MR are not in [INACTIVE.]FILES/description:	Create a new description file with the MR number as the name and enter into the FILES/description directory.

Table D-1. dbcross Messages—Continued

dbcross Message	Response
The following MR names from [INACTIVE.]FILES/glob_solution are not in [INACTIVE.]MR	Save glob_solution file and reuse to propose MR.
MRX vs MRX, MRX vs MR, MRX vs ORG	
The following MR names from [INACTIVE.]MRX are not in [INACTIVE.]MR:	Verify MR record and either add record to MR relation or remove from MRX relation.
The following MR names from [INACTIVE.]MRX are not in [INACTIVE.]ORG:	Verify MR record and either add record to ORG relation or remove from MRX relation.
MG vs MG, MG vs G, MG vs MR, FILES vs MG	
The following MR GENERIC combinations in MG have no assigned developer:	Assign a developer or change the state so that it is neither under study nor assigned.
The following GENERIC names from MG are not in G:	Change invalid generic names to valid ones from list in G relation.
The following GENERIC names from INACTIVE.MG are not in G or INACTIVE.G:	Change invalid generic names to valid ones from list in G relation.
The following MR names from MG are not in [INACTIVE.]MR:	Verify MR records and either add missing or delete extra record.
The following MR names from MG should not be in MG, because the MRSTATUS from MR is “created”, “mra_deferred” or “mra_study”:	Verify MR records and either remove MG relation record or change MR relation MRSTATUS to <i>active</i> .
The following MR names from MR should be in MG, because the MRSTATUS from MR is “active”:	Verify MR records and either add record to MG relation or change MR relation MRSTATUS to <i>created</i> .
The following MR GENERIC MGSTATUS combinations from INACTIVE.MG have invalid MGSTATUS. It should be “closed” because the MRSTATUS from INACTIVE.MR is “completed:”	Change the MGSTATUS in the IDB for that MRG to be closed.
The following MR GENERIC combinations from [INACTIVE.]FILES/resolution are not in [INACTIVE.]MG	Save the resolution file and resubmit the MR.
The following MR GENERIC combinations from [INACTIVE.]FILES/solution are not in [INACTIVE.]MG	Save the solution file and repropose the MR.
The following MR GENERIC combinations from [INACTIVE.]FILES/rejection are not in [INACTIVE] MG	Save the rejection file and rereject the MR.
The following MR GENERIC STATE SYSCODE combinations from MG contain inconsistent STATUS and SYSCODE:	Correct the MG record so that Status and Syscode match as defined by table 5-27 in the Administrator’s Guide.

Table D-1. dbcross Messages—Continued

dbcross Message	Response
Active GS vs Active GS, Active GS vs Active G Inactive GS vs Inactive GS, Active GS vs Inactive GS	
The following SOURCEFILE DIRECTORY GENERIC GSSTATUS combinations from [INACTIVE] GS indicate that the “source” command was renaming, moving, or deleting the file during the audits. This could cause unpredictable audit results.	Run the audits with the database stopped to avoid such messages. This does not necessarily indicate corruption.
The following GENERIC SOURCEFILE DIRECTORY COMMON records from GS contain inconsistent common field declarations:	The displayed records show disagreements between 2 or more generics. Determine which is correct and add or remove generic entries to the common field of the others.
The following GENERIC SOURCEFILE SCCSDIR combinations are duplicated in [active db]/GS:	Verify relation tuple files and remove duplicate entries.
The following GENERIC SOURCEFILE DIRECTORY SCCSDIR GSSTATUS combinations from [active db]/GS have invalid GSSTATUS:	Change the GS relation GSSTATUS so that it is either <i>free</i> or <i>busy</i> .
The following GENERIC SOURCEFILE SCCSDIR combinations are duplicated in [INACTIVE.]GS:	Verify relation tuple files and remove duplicate entries.
The following GENERIC SOURCEFILE SCCSDIR combinations are duplicated between GS and INACTIVE.GS:	Verify relation tuple files, decide whether they should be active or inactive, and remove the appropriate entries.
The following SOURCEFILE DIRECTORY GENERIC GSSTATUS combinations from [INACTIVE.]GS have invalid GSSTATUS:	If error occurred in the ADB, change GSSTATUS to be free or busy. If error occurred in the IDB, change GSSTATUS to be free.
The following GENERIC names from [INACTIVE.]GS are not in [INACTIVE.]G:	Change generic names in GS relation to a valid choice from the list in the G relation.
The following SOURCEFILE DIRECTORY GENERIC GSSTATUS combinations from inactive GS show a GSSTATUS of other than “free”. All files from inactive generics should have GSSTATUS of “free”:	Verify whether the file should be in inactive DB, and either remove the record or set the GSSTATUS to free.
The following GENERIC SOURCEFILE DIRECTORY combinations from [INACTIVE] GS show reserved MR checkout, but zero total checkouts:	Verify MR and source file records* and either remove the “rmr” entry in the GS record, or change the “cocnt” record to reflect the correct number of checkouts.
Active MD vs Active MD, Inactive MD vs Inactive MD	
The following GENERIC SOURCEFILE DIRECTORY combinations have more than one “reserved” checkout MD records:	Verify MR and source file records* and change all but one of the MD record’s MDSTAT to “unreserved”.

Table D-1. dbcross Messages—Continued

dbcross Message	Response
The following MR GENERIC SOURCEFILE DIRECTORY SID MDSTATUS combinations from [active db]/MD have invalid MDSTATUS:	Change MD relation MDSTATUS to either <i>delta</i> , <i>reserved</i> , or <i>unreserved</i> .
The following MR GENERIC SOURCEFILE DIRECTORY SID MDSTATUS combinations from inactive MD show an MDSTATUS other than "delta". All MD records from closed MRs should have MDSTATUS of "delta":	Verify that the record should exist in the INACTIVE MD and if so, change the MDSTATUS to <i>delta</i> .
WARNING: The following groups are empty:	No action is necessary, just informing user of the fact.
GRP vs GRP, GRPM vs GRP, GRP vs GRPM, GRP vs MR, GRP/GRPM vs other	
The following GROUPNAMEs from GRPM do not exist in GRP:	Verify group records and either add record to GRP relation or remove from GRPM relation.
WARNING: The following groups are empty:	This is a just a warning.
The following MR names from [INACTIVE.]MR and GROUPNAMEs from GRP are the same. MR and group names must be mutually exclusive:	Change the name of the group to a non-MR name.
The following GROUPNAMEs from CAS do not exist in GRP	Remove the groupname from the Cascade relation, or add a group by that name.
MS vs MG, MG vs MS	
The following MR GENERIC combinations from [INACTIVE.]MS are not in [INACTIVE.]MG:	Verify MR records and either add record to MR relation or remove from MS relation.
The following MR GENERIC MGSTATUS MSSTATUS combinations from [INACTIVE.]MG and [INACTIVE.]MS have incompatible statuses:	Change either MG relation or MS relation status so they are equal. Refer to list of valid statuses for each relation.
UMS vs MG	
The following MR GENERIC combinations from UMS are not in MG:	Verify MR records and either add record to the MG relation or remove from UMS relation.
MS vs GS, GS vs MS, UMS vs GS	
The following GENERIC SOURCEFILE DIRECTORY combinations from MS are not in GS:	Verify MR and source file records and either add record to GS relation or remove from MS relation.
The following GENERIC SOURCEFILE DIRECTORY combinations from INACTIVE.MS are not in GS or INACTIVE.GS:	Verify MR and source file records in the IDB and either add record to the GS relation in the IDB or remove the record from the MS relation in the IDB.
The following GENERIC SOURCEFILE DIRECTORY combinations from GS are not in MS or INACTIVE.MS:	Verify MR and source file records and either add record to MS relation or remove from GS relation.

Table D-1. dbcross Messages—Continued

dbcross Message	Response
The following GENERIC SOURCEFILE DIRECTORY combinations from UMS are not in GS:	Verify MR and source file records and either add record to GS relation or remove from UMS relation.
MS vs MD, MD vs MS	
The following MR GENERIC SOURCEFILE DIRECTORY combinations from [INACTIVE.]MS are not in [INACTIVE.]MD:	Verify MR records and either add record to MD relation or remove from MS relation.
The following MR GENERIC SOURCEFILE DIRECTORY combinations from [INACTIVE.]MD are not in [INACTIVE.]MS:	Verify MR records and either add record to MS relation or remove from MD relation.
GS vs MD, MD vs GS	
The following GENERIC SOURCEFILE DIRECTORY combinations from MD have MDSTATUS of “reserved/unreserved”, but there is no corresponding “busy” record in GS:	Verify MR and source file records* and either change MD relation MDSTATUS to <i>delta</i> or change GS relation GSSTATUS to <i>busy</i> .
The following GENERIC SOURCEFILE DIRECTORY combinations from GS have GSSTATUS of “busy”, but there is no corresponding “reserved/unreserved” record in MD:	Verify MR and source file records* and either change an MD relation MDSTATUS to reserved or unreserved or change GS relation GSSTATUS to free.
The following GENERIC SOURCEFILE DIRECTORY RMR combinations from GS show the MR as a reserved checkout, with no corresponding MD reserved checkout record:	Verify MR and source file records* and either remove the reserved MR number from the GS record, or change the check-out type of an MD record to reserved.
The following MR GENERIC SOURCEFILE DIRECTORY combinations from MD are listed as reserved checkouts, but are not present in the “reserved MR (rnr)” for any GS record:	Verify MR and source file records* and either add the reserved MR number to the GS record, or change the check-out type in MD to unreserved.
The following GENERIC SOURCEFILE DIRECTORY COCNT #_of_MD have COCNT (check-out count) showing a different number of checkouts in GS than there are reserved or unreserved MD records:	Verify MR and source file records* and change COCNT to show the correct number of existing check-outs.
UMS vs MS, MS vs UMS	
The following MR GENERIC SOURCEFILE DIRECTORY combinations from UMS are not in MS:	Verify MR and source file records* and either add record to MS relation or remove from UMS relation.
The following MR GENERIC SOURCEFILE DIRECTORY combinations from UMS also exist in MS, but the MSSTATUS is [approved]:	Verify MR and source file records* and either change MS relation MSSTATUS to <i>unapproved</i> or remove record from UMS relation.

Table D-1. dbcross Messages—Continued

dbcross Message	Response
The following MR GENERIC SOURCEFILE DIRECTORY combinations from MS have MSSTATUS of [unapproved], but do not exist in UMS	Verify MR and source file records* and either change MS relation MSSTATUS to <i>approved</i> or add a record to UMS relation.
FTD vs FTD	
WARNING: The following Command/Field Labels (right column) do not match the '\$cmd1' command Screen Label (left column). Format Below: (CMD INT-KEY LABEL CMD INT-KEY LABEL):	This is just a warning that an inconsistency exists. You can either ignore it or change the field label names to be consistent.
<*> WARNING: The following Command/Field External Keys (right column) do not match the '\$cmd1' command External Key (left column). Format Below: (CMD INT-KEY EXT-KEY CMD INT-KEY EXT-KEY):	This is just a warning that an inconsistency exists. You can either ignore it or change the field label names to be consistent.
The following COMMAND INT-KEY names have their Display Flag turned on, but the '\$cmd1' command Display Flag is turned off:	Either turn off the Display flag for the COMMAND FIELD listed or turn on the Display flag for the specified field in the command mentioned.
The following COMMAND EXT-KEY combinations in FTD are duplicated	Remove one of the records.
The following COMMAND INT-KEY names do not have their Display Flag turned on:	This is not an error; it is just an informative message.
ADM vs ADM, ADM vs. GS, ADM vs GRP, ADM vs /etc/passwd	
The ADM relation does not contain exactly 1 record.	Remove all but one tuple file in the ADM relation.
The ADM relation shows Temporary Branching as disabled, but the following GENERIC SOURCEFILE DIRECTORY combinations show multiple checkouts in COCNT	Verify MR and source file records*. Normally, turning off temporary branching (using setrel) is not permitted while any files are checked out more than once. This condition suggests that the "brflags" flag (for "Temporary Branching") in the ADM relation was changed manually to "n". Either use setrel to set the flag back to "y", or use edput or unedget to eliminate the extra check-outs of the indicated file(s).
The following database administrator from ADM is not a valid PTS group.	Change the first field of the ADM relation so that it contains the name of a valid group.
The following MR administrator from ADM is not a valid PTS group	Change the second field of the ADM relation so that it contains the name of a valid group.

Table D-1. dbcross Messages—Continued

dbcross Message	Response
PTS vs GRP, PTS vs GRPM, GS vs PTS/GRP, MG vs PTS/GRP, CRIT vs PTS/GRP, PTS vs. Key File	
The following database administrator from ADM is not a valid PTS group:	Change the first field of the ADM relation so that it is the name of a valid PTS group.
The following MR administrator from ADM is not a valid PTS group.	Change the second field of the ADM relation so that it is the name of a valid PTS group.
The following GROUP OWNERS from GRP are not valid PTS IDs:	Change the invalid group owners to valid PTS IDs.
The following GROUP MEMBERS from GRPM are not valid PTS IDs:	Delete the invalid group members from the group.
The following FILE OWNERS from GS are not valid PTS IDs/groups:	Change the invalid file owner to valid groups/PTS IDs.
The following ASSIGNEE's from MG are not valid PTS IDs/groups:	Either reassign the MR to a valid PTS ID/group or create a PTS ID/group for the MR assignee.
The following CRITERIA OWNERS from CRIT are not valid PTS IDs/groups:	Either delete the Criteria Owner from MG or create a PTS ID/group for the Criteria Owner.
WARNING: The number of PTS records listed as licensed in the PTS relation exceeds the maximum of [xxx] as defined in the license file.	Either contact the Sablime helpdesk to increase the number of users allowed for your current contract, or change the excess PTS records to unlicensed.
WARNING: The number of PTS records listed as licensed for Sablime merge tool in the PTS relation exceeds the maximum of [xxx] as defined in the license file	Either contact the Sablime helpdesk to increase the number of users allowed to use the Sablime merge tool, or change the excess PTS records to unlicensed.
The PTS license file is not readable.	Change permissions on the PTS license file so that it is readable.
The following PTS records are listed as licensed in the PTS relation, but are not in the license file:	Change the PTS records for those users to show them as not licensed. Then use the pts command to license them, if need be.
The following PTS records are listed as licensed for Sablime merge tool in the PTS relation, but are not in the license file:	Change the PTS records for those users to show them as not licensed. Then use the pts command to license them, if need be.
DEP vs DEP, DEP vs MG, DEP vs. MD	
The following MR GENERIC DEPMR combinations from [INACTIVE.]DEP and [INACTIVE.]PDEP have Logical and Physical dependency flags set to "n":	Set either the Logical or the Physical dependency flag in the DEP relation to "y."
The following MR GENERIC DEPMR combinations from [INACTIVE.]DEP were found in PDEP but Physical dependency flag was not set to "y":	Set the Physical dependency flag in the DEP relation to "y."

Table D-1. dbcross Messages—Continued

dbcross Message	Response
The following MR GENERIC DEPMR combinations from [INACTIVE.]DEP were not found in PDEP but Physical dependency flag was not set to “n”:	Set the Physical dependency flag in the DEP relation to “n.”
The following MR GENERIC combinations from [INACTIVE.]DEP are not in [INACTIVE.]MG:	Verify the MR records and either add record to the MG relation or remove from the DEP relation.
The following DEPMR GENERIC combinations from DEP are not in MG or INACTIVE.MG:	Verify the MR records and either add record to the MG relation or remove from the DEP relation.
The following DEPMR GENERIC combinations from INACTIVE.DEP are not in INACTIVE.MG:	Verify the MR records and either add record to the MG relation or remove from the DEP relation.
The following MR GENERIC FILENAME DIR SID combinations from [INACTIVE.]PDEP are not in [INACTIVE.]MD:	Remove the PDEP record or add an MD record with corresponding values.
GT vs G	
The following Generic names from [INACTIVE.]GT are not in [INACTIVE.]G:	Verify generic records and either add record to the G relation or remove from the GT relation.
The following Generic names from [INACTIVE.]G are not in [INACTIVE.]GT:	Verify generic records and either add record to the GT relation or remove from the G relation.
MRS vs MG, MG vs MRS, MRS vs MR	
The following spawned MR GENERIC combinations from [INACTIVE.]MRS are not in [INACTIVE.]MG:	Verify MR records and either add record to the MG relation or remove from the MRS relation.
The following spawned MR names from [INACTIVE.]MRS are not in [INACTIVE.]MR:	Verify MR records and either add record to the MR relation or remove from the MRS relation.
The following MR GENERIC combinations from FILES/spawnotes are not in MRS:	Verify that the MR was spawned and then either create the MRS record or remove the spawnotes file.
The following MR GENERIC combinations from [INACTIVE.]MG are in the spawned state but are not in [INACTIVE.]MRS:	Create a new spawnotes file with the MR number as the name in the FILES/spawnotes directory.
The following spawned MR GENERIC combinations from [INACTIVE.]MG are not in [INACTIVE.]MRS:	For each case, either take the record out of the MG relation or add an MRS record for it.
The following MR GENERIC combinations from [INACTIVE.]MG are in the spawned state but are not in [INACTIVE.]MRS:	Add new records to the MRS relation for the MG records mentioned.

Table D-1. dbcross Messages—Continued

dbcross Message	Response
The following MR GENERIC combinations from MRS are not in MG with a STATUS of “spawned”:	Change the status in the MG relation to “spawned” for those MG records.
The following MR GENERIC combinations from MRS are not in MG:	For each case, either take the record out of the MRS relation or add an MG record for it.
MR Initialization Check	
The following MR GENERIC MGSTATUS TYPE combinations from MG are [initialization] type MRs, but have not yet been approved:	Have the approval team approve the MR as soon as possible to avoid future corruption in the SDB.
Inactive MR vs Inactive ORG, Inactive ORG vs Inactive MR	
The following MR names from INACTIVE MR are not in INACTIVE ORG:	Verify MR records and either add record to ORG relation or remove from MR relation.
The following MR names from INACTIVE ORG are not in INACTIVE MR:	Verify MR records and either add record to MR relation or remove from ORG relation.
Inactive MR vs Inactive MR	
The following INACTIVE MR's are missing for which spawned MR's exist:	Verify the MRS relation record and either add record to MR and/or ORG relation or delete record from MRS relation.
Inactive MG vs Inactive MR	
The following MR names from INACTIVE MG are not in INACTIVE MR:	Verify MR records and either add missing or delete extra record.
Inactive MS vs Inactive MD, Inactive MD vs Inactive MS	
The following MR GENERIC SOURCEFILE DIRECTORY combinations from INACTIVE MS are not in INACTIVE MD:	Verify MR records and either add record to MD relation or remove from MS relation.
The following MR GENERIC SOURCEFILE DIRECTORY combinations from INACTIVE MD are not in INACTIVE MS:	Verify MR records and either add record to MS relation or remove from MD relation.
Inactive MS vs Inactive MG, Inactive MD vs Inactive MG	
The following MR GENERIC names from INACTIVE MS are not in INACTIVE MG:	Verify MR records and either add record to MD relation or remove from MS relation.
The following MR GENERIC names from [INACTIVE.]MD are not in [INACTIVE.]MG:	Verify MR records and either add record to MS relation or remove from MD relation.
CP vs. PTS, CP vs. GS, CP vs. CP	
The following field from CP should NOT be comma-separated:	Only one entry can be in this field. Create more CP records for the extra entries.
The following CMD GENERIC FUNCTYPE combinations from CP should NOT be empty in the fourth field:	Fill in an executor list for this CP record.

Table D-1. dbcross Messages—Continued

dbcross Message	Response
The following CMD GENERIC FUNCTYPE combinations from CP should NOT be empty in the fifth field:	Fill in an email list for this CP record.
The following CMD GENERIC combinations from CP have an invalid generic:	Replace the invalid generic with a valid one, or eliminate the record.
The following CMD FUNCTYPE combinations from CP are incorrect. FUNCTYPE should be blank for that CMD:	Remove third field for CP record starting with command mentioned.
The following CP CMDs have _AD in fields 4 or 5, but are not allowed to:	Take out the _AD entry for the CP record starting with the command mentioned.
_NONE is not allowed as an EMAIL RECIPIENT for the following CP CMDs:	Enter “_DEFAULT,” “_AD,” “_ORIG,” a PTS ID, a PTS ID group, an email group or an email address in the last field for this CP record.
The following CMD is invalid for CP:	The only commands allowed are those listed in table 4-7 of the Administrator’s Guide.
The following CMD FUNCTYPE combination is invalid for CP:	Change the FUNCTYPE according to table 4-7 of the Administrator’s Guide.
The following EXECUTOR(s) from CP are invalid:	Remove the executor from the 4th field.
The following EMAIL RECIPIENT(s) from CP are invalid:	Remove the email recipient from the 5th field.
The following CMD GENERIC FUNCTYPE combinations from CP are duplicated:	Verify relation tuple files and remove duplicate entries.
The following CMD GENERIC FUNCTYPE combinations from CP contain empty fields while CMD indicates that this is incorrect:	These keys should not be empty. Fill in the appropriate fields.
The following CMD GENERIC combinations from CP are incorrect. GENERIC should be blank for that CMD:	Blank out the Generic field for that record.
The following CMDs from CP have _ALL or _NONE mixed with other values in field 4 or 5:	In fields 4 and 5 of CP, _ALL and _NONE can only appear by themselves. Either remove the _ALL or _NONE or the other value(s) in that field.
Miscellaneous	
The following SNAPID GENERIC combinations from [INACTIVE.]SNAP have empty CREATOR:	Enter a PTS ID in the third field of that SNAP record.
The following SNAPID GENERIC combinations from SNAP have invalid GENERIC:	Put a correct generic into the second field of that SNAP record.
The following CREATOR from [INACTIVE.]SNAP is invalid:	Replace invalid creator with a valid PTS ID.

Table D-1. dbcross Messages—Continued

dbcross Message	Response
The following FILE DIR GENERIC combinations from FZ are not in GS:	Fix the FZ record so that it matches a GS record.
The following FILE DIR GENERIC SID combinations from FZ are not in MD or INACTIVE.MD:	Fix the FZ record so that it matches an MD record.
The following GENERIC SNAPID combinations from FZ are not in SNAP:	Fix the FZ record so that it matches a SNAP record.
Relation \$rel expected \$N fields but got \$nf	Fix the record so that it has the correct number of fields.
The following keys have empty fields in \$rel:	Keys should not be empty. Fill in the appropriate fields.
The following TESTERs from MG are not valid PTS IDs/groups:	Change the entry in the tester field to be valid PTS IDs or groups.
The following COMMAND EXT-KEY combinations in FTD are duplicated:	Verify relation tuple files and remove duplicate entries.
The following MR GENERIC combinations from MRS are not in FILES/spawnotes:	Create a spawnotes file for that MRS record.
The following ASSIGNEE's from [INACTIVE.]MRX are not valid PTS IDs/groups:	Change the entry in the assignee field to be valid PTS IDs or groups.
dbcross internal error: unknown CPcmd group [\$cgrp] for [\$cmd]	There is a bug in the program. Report this to the Sablime HOTLINE.
The following MR GENERIC spawned MR combinations from [INACTIVE.]MRS do not belong together:	The spawned MR must contain the parent MR number.
Missing directory: \$adb/FILES/spawnotes/\$gen	There is a bug in the program. Report this to the Sablime HOTLINE.
No \$adb/FILES/spawnotes directory. Could not run tests!	Create the spawnotes directory in the ADB.

* In the above table “Verify MR and sourcefile records” means to verify the following:

- n The file, if it has any MD records, has a GS record.
- n The GS record has a GSSTATUS of busy if it is checked out in the current generic.
- n The GS COCNT has the total number of existing checkout for the file in the current generic.
- n There is an MD record with an MDSTATUS of “reserved“ or “unreserved“ for each check-out of the file.

- n There is an MD record with an MDSTATUS of “delta” for each completed check-in of the file.
- n There is no more than one “reserved” check-out.
- n The “reserved” check-out (if there is one) has its MR number listed in the GS record as “rnr”.
- n The “reserved” check-out has a lock file (p.) in the Source Database (SDB).
- n Each completed check-in has a corresponding record in the SDB history (s.) file.
- n There is a UMS record for any unapproved MR’s file changes, and an MS record for changes approved or not.

The audits report when any of the above are not correct, but it is up to the administrator to determine which of the conflicting records is wrong.

Messages from dbxcross

Table D-2 shows the messages generated by the audit program dbxcross and the appropriate response to each. All reported errors are from the Active Database unless otherwise indicated.

Table D-2. dbxcross Messages

dbxcross Message	Response
\$_ not in environment and not found in \\${sab}VAR file.	Variable must be set and exported.
ERROR: Global database [\${sabGDB}] is not a directory.	Check that sabGDB is set to proper location.
File System and Inode Space Check	
Execution of the 'spacecheck' script has failed due to the following variables being unset: <Variable List>	Edit spacecheck and set these variables.
COM vs G	
The following GENERIC names from COM are missing in G: (list of generics)	Verify generic records and either add record to the G relation or remove from the COM relation.
EMR vs MR	
The following MR names from EMR are not in MR: (list of mrs)	Verify the MR records and either add record to MR relation or remove from EMR relation
EMR vs ORG	
The following MR names from EMR are not in ORG: (list of mrs)	Verify the MR records and either add record to ORG relation or remove from EMR relation.

Table D-2. dbxcross Messages—Continued

dbxcross Message	Response
EMG vs EMR	
The following External ID Names in EMG have a broken link, the EMR record is missing: (list of IDs) NOTE: An investigation into the Sender's Databases must occur.	The ID name must be found from the Database of the product that sent the MR. Once known, it must be entered into the fourth field of the EMR relation.
The following External Product Names found in EMG are not in EMR (list of products)	Verify product records and change product names so they are the same.
MG vs COM, MG vs FILES	
The following Commitment ID Names found in MG are not in COM (list of IDs)	Verify MR records and add ID to second field of COM relation.
The following GENERIC/COMMID names found in MG do not have corresponding files in [\$sabADB/FILES/mrcommit]	Save the mrcommit file and recommit the MR.
The following MR GENERIC COMMID records found in MG do not have corresponding [\$sabADB/FILES/mrcommit] records	Recommit the MR.
DOC vs G, DOL vs G, DS vs G	
The following GENERIC Names in DOC are not in G (list of generics)	Verify generic records and change generic names in DOC relation to match those from valid list of generics in G relation.
The following GENERIC Names in DOL are not in G (list of generics)	Verify generic records and change generic names in DOL relation to match those from valid list of generics in G relation.
The following GENERIC Names in DS are not in G (list of generics)	Verify generic records and change generic names in DOL relation to match those from valid list of generics in G relation.
DS vs DOL	
The following GENERIC DOC DIR combinations in DS are not in DOL (list of generics, documents, and directories)	Verify generic records and either add record to the DOL relation or remove from DS relation.
DS vs GS, DOL vs GS	
The following GENERIC DOC DIR combinations in DS are not in GS (list of generics, documents, and directories)	Verify generic and source file records and either add record to GS relation or remove from DS relation.
The following GENERIC DOC DIR combinations in DOL are not in GS list of generics, documents, and directories)	Verify generic and source file records and either add record to GS relation or remove from DOL relation.

Table D-2. dbxcross Messages—Continued

dbxcross Message	Response
HC vs G, HC vs HC, MG vs HC, HC vs FILES	
The following GENERIC names from HC are missing in G:	Verify the valid generic names in G and modify the record in the HC relation with a valid generic name.
The following HCODE GENERIC names are duplicated in HC:	Verify relation tuple files and remove duplicate entries.
The following HCODE names from MG are not in HC:	Verify HC hcode names and modify the record in the MG relation with a valid hcode name.
The following GENERIC/HCODE names from FILES/hcode are not in HC:	Either add the missing record to the HC relation or delete the hcode description from FILES/hcode/ <i>generic</i> directory.
PDI vs G, PDI vs PDI, MG vs PDI	
The following GENERIC names from PDI are missing in G:	Verify the valid generic names in G and modify the record in the PDI relation with a valid generic name.
The following PDI GENERIC names are duplicated in PDI:	Verify relation tuple files and remove duplicate entries.
The following PDI names from MG are not in PDI:	Verify PDI relation pdi names and modify the record in the MG relation with a valid pdi name.

Messages from dbdelta

Table D-3 shows the messages generated by the audit program dbdelta and the appropriate response to each. All reported errors are from the Active Database unless stated otherwise.

Table D-3. dbdelta Messages

dbdelta Message	Response
\$_ not in environment and not found in \\${sabVAR} file\n	Variable must be set and exported.
ERROR: Global database [\${sabGDB}] is not a directory.	Check that sabGDB is set to proper location.
File System and Inode Space Check	
Execution of the 'spacecheck' script has failed due to the following variables being unset: <Variable List>	Edit spacecheck and set these variables according to the description in <i>Checking for Disk Space</i> .

Table D-3. dbdelta Messages—Continued

dbdelta Message	Response
Starting GS retrieve / SDB PRS	
GS record exists for [[\${sabSDB}]/\${SDBPATH}/s.\${Source}], but the SDB file doesn't exist or has 0 size.	Remove records from the following ADB relations: GS, MS, MD, UMS and (if it exists) remove the SDB file; then return the file to the Database using addisrc or addgsrc.
GS record exists for [[\${sabSDB}]/\${SDBPATH}/s.\${Source}], but the SDB file is unreadable.	Remove records from the following ADB relations: GS, MS, MD, UMS and (if it exists) remove the SDB; file then return the file to the Database using addisrc or addgsrc.
GS record exists for [[\${sabSDB}]/\${SDBPATH}/s.\${Source}], but the status of the SDB file is unknown.	Remove records from the following ADB relations: GS, MS, MD, UMS and (if it exists) remove the SDB file; then return the file to the Database using addisrc or addgsrc.
The following FILE DIR GEN combinations do not have a valid 'Binary Flag' set:	Check the SDB file for binary characteristics; use the source command to set the Binary flag to y or n accordingly.
The following FILE DIR GEN combinations do not use a valid 'Version Control Tool' of either SBCS or SCCS:	Check the Binary flag; if the flag is set to y , use the source command to change the Version Control Tool to SBCS. If the flag is set to n , choose a valid Version Control Tool.
The following FILE DIR GEN combinations have a 'Binary Flag' set to 'y' but the 'Quality Assurance Flag' is not set to 'n':	Use the source command to set the Quality Assurance flag to n .
The following FILE DIR GEN combinations have a 'Binary Flag' set to 'y' but the 'Version Control Tool' is not set to 'SBCS':	Use the source command to set the Version Control Tool to SBCS .
Starting MD / GS join file build Starting JOIN Starting SDB MD tuple compression	
The following errors were detected while processing the SDB files: (list of errors)	The errors that appear here are from SCCS or SBCS directly. Execute the SCCS help or SBCS nhelp command on the help number given to find the actual problem and solution.
The following errors were generated by the SDB prs command: (list of errors)	The errors that appear here are from SCCS or SBCS directly. Execute the SCCS help or SBCS nhelp command on the help number given to find the actual problem and solution.
The following SDB files do not have a corresponding GS record: (list of files)	Verify the SDB file and MR records in the MS, MD, and UMS relations and either add record to the GS relation or remove the SDB file.

Table D-3. dbdelta Messages—Continued

dbdelta Message	Response
The following generic SIDs found in the SDB files reflect non-existent G records: (list of SIDs)	Compare SID numbers with those in the third field of the G relation. Remove any SDB file SID with the SCCS rmdel or SBCS nrmdel command.
The following FILE GENERIC sabSDB combinations reflect nonexistent GS records: (list of file names, generics, and Databases)	Verify the SDB file and MR records in the MS, MD, and UMS relations and either add record to the GS relation or remove the SDB file.
The following SDB deltas [FILE sabSDB SID PGMR MDSTATUS] reflect non-existent MD combinations: (list of deltas)	Verify the SDB file delta numbers and add missing delta records to the MD relation.
The following MD combinations [FILE (GS_SDBDIR) SID PGMR MDSTATUS] show an expected check-in (SID) that is beyond the next available delta in the SDB:	This may mean that a delta was manually removed (rmdel) from the SDB. Unedget and re-edget the existing check-outs of the file.
The following MD combinations [FILE (GS_SDBDIR) PSID PGMR MDSTATUS] show a parent delta ID (PSID) that does not exist in the SDB:	This may mean that a delta was manually removed (rmdel) from the SDB. Unedget and re-edget the existing check-outs of the file.
The following MD combinations [FILE (GS_SDBDIR) SID PGMR MDSTATUS] show a reserved checkout where the expected delta is not in the next available delta in the SDB:	This could mean that a delta was manually removed (rmdel) from the SDB, or some other corrupting event occurred. Unedget and re-edget the existing check-outs of the file.
The following MD combinations [FILE (GS_SDBDIR) SID PGMR MDSTATUS] reflect non-existent SDB deltas:	Verify the MR records in the MD relations in the IDB and ADB, and check with the MS and UMS relations, and with the source file. If these MR records are useless, remove them from the MD relations; otherwise, remove them and redo the changes for the MRs to the source file.
The following file, directory, SID and MDSTATUS were duplicated in the MD relation	Check the SDB file to see which of these deltas (if any) ended up in the source file. Remove all but one of the MD records for this SID.
The following SID were duplicate in SDB delta files	Check the SDB file and any related "p." file for duplicated deltas. If the duplicated delta is the most recent, use unedput to remove it. Otherwise, this is fairly severe corruption of the SCCS file. Contact the help desk.
The following SDB files do not have a corresponding GS record:	Add a GS record for each generic where this file has any SDB file deltas.

Table D-3. dbdelta Messages—Continued

dbdelta Message	Response
The following FILE DIR DELTA combinations from the GS relation have corrupt lastofcsid	Verify the status of the MRs against this SBCS file, and insert the highest numbered approved MD SID into lastofcsid.
WARNING: The following LOGICAL DIRECTORY Names found in the GS Relation do not have a corresponding Directory Structure File entry [generic]:	These directories should probably be added to your Directory Structure File (\$sabDIRF) for the indicated generic (so that setnode operations create them).
WARNING: The following GENERIC Names do not have a corresponding Directory Structure File in the Global Database DIR Relation:	If the indicated generics have files associated with them, then a Directory Structure File (\$sabDIRF) should be created.
The following FILE DIR GENERIC MR combinations are missing from MD relation for official branch:	Add MD records for these approved deltas from the SDB file.
The following FILE DIR DELTA combinations from the GS relation are missing from the MD relation:	Verify the status of the deltas for this file and add records to the MD relation as necessary.

Messages from hotline.ck

Table D-4 shows the messages generated by the audit program hotline.ck. The Sablime variables may appear in either the xsablime.sh script or the Sablime variable file sabVAR, depending on how the system has been configured by the Sablime Administrator.

Table D-4. hotline.ck Messages

hotline.ck Message	Response
ERROR: Cannot read variable [sabDIRF].	Variable must be set and exported.
ERROR: Cannot read variable [sabGDB].	Variable must be set and exported.
ERROR: Cannot read variable [sabGEN].	Variable must be set and exported.
ERROR: Cannot read variable [sabHOST].	Variable must be set and exported.
ERROR: Cannot read variable [sabLCB].	Variable must be set and exported.
ERROR: Cannot read variable [sabMCB].	Variable must be set and exported.
ERROR: Cannot read variable [sabNET].	Variable must be set and exported.
ERROR: Cannot read variable [sabPROD].	Variable must be set and exported.
ERROR: Cannot read variable [sabPTS].	Variable must be set and exported.
ERROR: Cannot read variable [sabSERV].	Variable must be set and exported.

Table D-4. hotline.ck Messages—Continued

hotline.ck Message	Response
ERROR: Cannot obtain the current machine name.	Find out the command for getting the current machine name and change <code>hotline.ck</code> . Then let the hotline people know.
ERROR: Global database [<code>\$sabGDB</code>] is not a directory.	Check the path of <code>\$sabGDB</code> .
Sablime BIN Permissions	
The Host Bin Directory does not have the correct permissions of 755.	Change the mode of this directory to 755.
The following Host Sablime Commands do not exist:	Contact your local system administrator to recover the missing commands from backup or call the Sablime hotline to receive the missing commands.
The following Sablime Commands do not have the correct permissions of 4755:	Change the mode of these commands to 4755.
The following Sablime Commands do not have the correct permissions of 755:	Change the mode of these commands to 755.
The following Sablime Commands do not have the correct permissions of 700:	Change the mode of these commands to 700.
The <code>[SPOOL]</code> variable is set to an invalid directory.	Check the value of <code>\$sabSPOOL</code> .
The MailSpool Directory does not have the correct permissions of 777	Change the mode of this directory to 777.
The MailSpool Directory does not exist.	Create the MailSpool Directory under <code>\$sabLCB</code> or <code>\$sabMCB</code> .
Warning: The following MailSpool Lock Files may cause problems with Mail delivery.	If the MailDemon is running and the process number matches the contents of the lock file, do nothing. Otherwise, remove the lock file.
Sablime xbin Permissions	
The following Host Sablime Commands do not exist:	Contact your local system administrator to recover the missing commands from backup or call the Sablime hotline to receive the missing commands.
The following Sablime Commands do not have the correct permissions of 4755:	Change the mode of these commands to 4755.
The following Sablime Commands do not have the correct permissions of 755:	Change the mode of these commands to 755.

Table D-4. hotline.ck Messages—Continued

hotline.ck Message	Response
The following Sablime Commands do not have the correct permissions of 444:	Change the mode of these commands to 444.
The setperm file does not exist.	Get the setperm file from the ftp site. This file is included in one of the tar files.
The setperm file does not have the correct permissions of 755.	Change the mode of this file to 755.
Sablime Satellite bin Permissions	
You are executing this script from the satellite machine [\$CUR_MACH] and your environment variable sabNET is set to '0'; this is an error.	Set sabNET to 5, if NFS/RFS networking is being used. Set sabNET to 7, if TCP/IP networking is being used.
The following Satellite Sablime Commands do not exist:	Contact your local system administrator to recover the missing commands from backup or call the Sablime hotline to receive the missing commands.
The following Sablime Commands do not have the correct permissions of 4755:	Change the mode of these commands to 4755.
The following Sablime Commands do not have the correct permissions of 755:	Change the mode of these commands to 755.
The following Sablime Commands do not have the correct permissions of 444:	Change the mode of these commands to 700.
The following Sablime Bin Executables are not owned by [\$LOGNAME] logid:	Change the owner of the Bin Executables to \$LOGNAME.
The following Sablime Bin Directories are not owned by [\$LOGNAME] logid:	Change the owner of the Bin Directories to \$LOGNAME.
Global Database Permissions	
The Global Database does not have the correct permissions of 755	Change the mode of this directory to 755.
The Multi-Machine Flag for this product is set to '1', but the Global Database [tmp] Directory does not have the correct permissions of 777	Change the mode of this directory to 777.
The following Global Database Relations do not exist:	Execute the mkdir command to make the missing directories, then refer to Sablime to replace all missing records. If assistance is required, call the Sablime hotline.
The following Global Database Relations do not have the correct permissions of 755:	Change the mode of these directories to 755.

Table D-4. hotline.ck Messages—Continued

hotline.ck Message	Response
The following Global Database RELATION/TUPLES do not have the correct permissions of 644:	Change the mode of these tuple files to 644.
Active Database Permissions	
The Active Database does not have the correct permissions of 755	Change the mode of this directory to 755.
The SAB.stopfile does not exist	Recreate the SAB.stopfile by redirecting stdout to the file (i.e., >SAB.stopfile).
The SAB.stopfile does not have the correct permissions of 644	Change the mode of this file to 644.
The Multi-Machine Flag for this product is set to '1', but the Active Database [tmp] Directory does not have the correct permissions of 777	Change the mode of this directory to 777.
The following Active Database Relations do not exist:	Execute the mkdir command to make the missing directories; then refer to the section on Sablime to replace all missing records. If assistance is required, call the Sablime hotline.
The following Active Database Relations do not have the correct permissions of 755:	Change the mode of these directories to 755.
The following Active Database RELATION/TUPLES do not have the correct permissions of 644:	Change the mode of these tuple files to 644.
Inactive Database Permissions	
The Inactive Database does not have the correct permissions of 755	Change the mode of this directory to 755.
The Multi-Machine Flag for this product is set to '1', but the Inactive Database [tmp] Directory does not have the correct permissions of 777	Change the mode of this directory to 777.
The following Inactive Database Relations do not exist:	Execute the mkdir command to make the missing directories; then refer to the section on Sablime to replace all missing records. If assistance is required, call the Sablime hotline.
The following Inactive Database Relations do not have the correct permissions of 755:	Change the mode of these directories to 755.
The following Inactive Database RELATION/TUPLES do not have the correct permissions of 644:	Change the mode of these tuple files to 644.

Table D-4. hotline.ck Messages—Continued

hotline.ck Message	Response
Source Database Permissions	
The Source Database does not have the correct permissions of 755	Change the mode of this directory to 755.
The following Source Database Directories are listed in the 'DIRF' file, but do not exist in the Source Database:	Execute the mkdir command to make the missing directories; then execute the dbdelta command to receive a list of the missing files. If assistance is required, call the Sablime hotline.
The following Source Database Directories do not have the correct permissions of 755:	Change the mode of these directories to 755.
The following Source Database DIRECTORY/SDBFILES do not have the correct permissions of 444:	Change the mode of these files to 644.
Miscellaneous Permissions	
The '/bin/sh' executable should not have the permissions of 4755	Contact your local system administrator to have the permissions changed on /bin/sh to 755.
The [NET] variable must be set to '0' on the Host Machine	Change the sabNET variable.
The [LCB] and [MCB] variables are not set to the same value	Change the sabLCB and/or sabMCB variable so they are equal.
WARNING: The following PTSID's are not valid login ID's:	Either contact your local system administrator to set up logins for these PTS IDs or remove the PTS IDs from the Global Database.
The following PTSID's have Userids and/or Groupids that are below 100:	Contact your local system administrator to change the UIDs and/or GIDs to a number greater than 100.
The [SERV] variable is either unset or set to '/dev/null'	On the satellite machine, change the sabSERV variable to net_recv.
WARNING: The [SERV] variable is not set to 'net_recv'	On the satellite machine, change the sabSERV variable to net_recv.

Table D-4. hotline.ck Messages—Continued

hotline.ck Message	Response
Required TCP/IP network file '/etc/services' does not exist	Contact your local system administrator to have this file installed.
File '/etc/services' does not have an entry for 'sablime'	Contact your local system administrator to have an entry created.
WARNING: File '/etc/services' has a value out of the recommended range for 'sablime' Recommended range is 1524-6000	Contact your local system administrator to have an entry created.

Messages from spacecheck

Table D-5 shows the messages generated by the audit program spacecheck and the appropriate response to each.

Table D-5. spacecheck Messages

spacecheck Message	Response
Cannot read variable [sabGDB].	Variable must be set and exported.
Cannot read variable [sabHOST].	Variable must be set and exported.
Cannot read variable [sabPROD].	Variable must be set and exported.
Spacecheck can only be executed from the host machine.	Run spacecheck on the host machine.
Execution of the 'spacecheck' script has failed due to the following variables being unset:	Variables must be set and exported.
Global database [\${sabGDB}] is not a directory.	Check the path of \$sabGDB.
File System [\$fs] is not a valid directory.	Check that the file system name stored in FSDB or supplied as the first argument is a valid directory.

Sablime Installation Worksheets

E

This appendix contains all the worksheets described in this book. Make copies of the worksheets you need to do your Sablime installation.

Worksheet 1: Pre-Installation Checklist

Before loading the Sablime executables, do the following:

- Establish a *sablime* login on the system.
- Make the Korn shell (ksh) the default shell for the *sablime* login.
- Create a directory for the Sablime commands.
- Create a file that contains the product's official directory structure.
- Make sure that the Source Code Control System (SCCS) is available on your machine.
- Establish networking between machines, when necessary, for multi-machine Sablime or the External Product Communications feature.
- Make sure you have a minimum process limit size of two megabytes (2 MB) available for Sablime and for all other logins using Sablime.
- Make sure a file system exists with enough free space for the Sablime Master Control Bin, VHELP messages, and databases.
- Make sure that all Sablime users' directories have permissions of at least 755 and that files that are stored in Sablime have permissions of at least 444.

Worksheet 2: Installation Information (initsab)

Multi-Machine Product? yes no

Global Database Path: _____

Active Database Path: _____

Inactive Database Path: _____

Source Database Path: _____

Sablime Master Control Bin: _____

Product Name: _____ (maximum 5 characters)

Product
Title: _____

Generic
Name: _____

Major Programming Language (select one):

- | | | | | |
|--------------------------------|-----------------------------------|--------------------------------|--------------------------------|------------------------------------|
| <input type="checkbox"/> bal | <input type="checkbox"/> dbd | <input type="checkbox"/> html | <input type="checkbox"/> mll | <input type="checkbox"/> shell |
| <input type="checkbox"/> c | <input type="checkbox"/> dirjcl | <input type="checkbox"/> jcl | <input type="checkbox"/> parms | <input type="checkbox"/> snowflake |
| <input type="checkbox"/> c++ | <input type="checkbox"/> document | <input type="checkbox"/> limbo | <input type="checkbox"/> perl | <input type="checkbox"/> spitbol |
| <input type="checkbox"/> cobol | <input type="checkbox"/> dtp | <input type="checkbox"/> lke | <input type="checkbox"/> pl1 | <input type="checkbox"/> vb |
| <input type="checkbox"/> comp | <input type="checkbox"/> fortran | <input type="checkbox"/> mark4 | <input type="checkbox"/> sal | <input type="checkbox"/> other |

Product Organization Number: _____

Contact Phone Number: _____

Project Site: _____

Path to Directory Structure File: _____

Default Version Control Tool for Non-Binary Files SCCS SBCS Either

Administrator Group Members

Database Administration

MR Administration

Source Administration

Generic Administration

MR Classes for Generic:

- | | | |
|----------------------------|------------------------------|-----------------------------|
| Document Test Teams/States | <input type="checkbox"/> yes | <input type="checkbox"/> no |
| Firmware Test Teams/States | <input type="checkbox"/> yes | <input type="checkbox"/> no |
| Hardware Test Teams/States | <input type="checkbox"/> yes | <input type="checkbox"/> no |
| Software Test Teams/States | <input type="checkbox"/> yes | <input type="checkbox"/> no |

Document Test Teams

Pre-Inspection Team

Inspection Team

Pre-Publishing Team

Publishing Team

Pre-Approval Team

Approval Team

Firmware Test Teams

Pre-Integration Team

Integration Team

Pre-System Test Team

System Test Team

Pre-Approval Team

Approval Team

Hardware Test Teams

Pre-Integration Team

Integration Team

Pre-System Test Team

System Test Team

Pre-Approval Team

Approval Team

Software Test Teams

Pre-Integration Team

Integration Team

Pre-System Test Team

System Test Team

Pre-Approval Team

Approval Team

Setting System Options

- | | | | |
|---|--|--|-------------------------------|
| Trace File | <input type="checkbox"/> yes | <input type="checkbox"/> no | |
| MR History File | <input type="checkbox"/> yes | <input type="checkbox"/> no | |
| Electronic Mail | <input type="checkbox"/> yes | <input type="checkbox"/> no | |
| In-Process Metrics Data | <input type="checkbox"/> yes | <input type="checkbox"/> no | |
| Automatic MR Routing | <input type="checkbox"/> yes | <input type="checkbox"/> no | |
| Automatic MR Assignment | <input type="checkbox"/> yes | <input type="checkbox"/> no | |
| MR Reassignment | <input type="checkbox"/> yes | <input type="checkbox"/> no | <input type="checkbox"/> all |
| Automatic Dependency | <input type="checkbox"/> Line
Level | <input type="checkbox"/> File
Level | <input type="checkbox"/> none |
| Developer Override of
Automatic Dependency | <input type="checkbox"/> yes | <input type="checkbox"/> no | |

Worksheet 4: primsdb Checklist

Before you run `primsdb`, do the following:

- Run the `dot sablime` program to set up the generic to which you will be adding files.
- If desired, categorize the source files by language and owner.
- Make sure that each file owner has a valid PTS ID.
- If the names of the files you are adding are longer than 12 characters, use the `ftd` command to increase the *L/R Scroll Size* field of the `addisrc` command.
- Create a file containing a list of the files you want to add.
Full path to this file: _____
- Use the `fcreate` command to create, accept, and assign an initialization MR.
- Make a note of the MR number used to initialize files:
MR Number: _____

Worksheet 5: Installation Information (primsdb)

Adding files for the first time? yes no

PTS ID or name of a group containing PTS IDs of the owner(s) of the files being added (for files copied from another generic, use **same** to keep the same ownership): _____



CAUTION:

Specified file owners are the only users allowed to edget the files they own.

Major Programming Language _____

Directory node where the files to be added are stored (applicable to files being added initially only): _____

If you are adding files from a previous generic, the name of that generic: _____

Are files common between generics? yes no

If files are not common, add source files from which branch?
 ofc (official) mr (unofficial)

Worksheet 6: Products List

List the products that you intend to put under a Sablime instance (generic names must be unique in an instance).

Product Name (Title) Versions (Generic)

Worksheet 7: Product Profile

Fill out a Product Profile for each product that you place under Sablime control.

Product Name: _____ (maximum 5 characters)

Product Title:

MR Classes:

Hardware Firmware Software Documentation

If Software is an MR Class, list the languages used in source code::

- | | | | | |
|--------------------------------|-----------------------------------|--------------------------------|--------------------------------|-----------------------------------|
| <input type="checkbox"/> bal | <input type="checkbox"/> dbd | <input type="checkbox"/> html | <input type="checkbox"/> mll | <input type="checkbox"/> shell |
| <input type="checkbox"/> c | <input type="checkbox"/> dirjcl | <input type="checkbox"/> jcl | <input type="checkbox"/> parms | <input type="checkbox"/> snoflake |
| <input type="checkbox"/> c++ | <input type="checkbox"/> document | <input type="checkbox"/> limbo | <input type="checkbox"/> perl | <input type="checkbox"/> spitbol |
| <input type="checkbox"/> cobol | <input type="checkbox"/> dtp | <input type="checkbox"/> lke | <input type="checkbox"/> pl1 | <input type="checkbox"/> vb |
| <input type="checkbox"/> comp | <input type="checkbox"/> fortran | <input type="checkbox"/> mark4 | <input type="checkbox"/> sal | <input type="checkbox"/> other |

Date Placed Under Sablime Control:

Generics

Generic	Release Date	Manager	H	F	S	D
_____	_____	_____	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
_____	_____	_____	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
_____	_____	_____	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
_____	_____	_____	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
_____	_____	_____	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
_____	_____	_____	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
_____	_____	_____	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
_____	_____	_____	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
_____	_____	_____	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

H=Hardware F=Firmware S=Software D=Documentation

Worksheet 8: **Host Assignments**

Machine Name:

UNIX Administrator Name:

List of Products Assigned to this Host:

Product Generics

Worksheet 9: Directory Structure

Product Name:

Host:

Master Control Bin:

Global Database:

Active Database:

Inactive Database:

Source Database:

Glossary

A

Active Database (ADB)

Contains all the active information about a Sablime *product*. It includes MR descriptions and information relating MRs to *generics*, developers, and file changes.

Approval Team

The staff responsible for approving resultant changes for an MR.

Assigned Developer (AD)

The user or group responsible for work related to an MR.

B

Bin

The directory where Sablime commands are installed on a machine. The recommended bin is the home directory of the *sablime* login.

Branch

Every file for each generic has two branches representing an approval level associated with it: the modification request (*mr*) branch and the official (*ofc*) branch. The *mr* branch contains all unapproved changes; the *ofc* branch contains all approved changes. An official source file corresponds to the version of the file that contains all *deltas* on the official branch that were made to the current date.

C

Command

An executable program that usually handles a user transaction.

D

Database Administrator (DBA)

Owner of the *sablime* login, the Sablime databases and the Sablime *commands*. Certain commands are restricted to use by the DBA.

Delta

The set of changes made to a Sablime *file* by each sequence of **edget/edit/edput**.

Each delta consists of administrative data added to the beginning of the file and the changes, if any, to the body or text of the file. Deltas are also made to a file when MRs are approved or new generics defined; however, these deltas generally add administrative data to the file without changing the body or text portion of the file.

dot sablime Command

A shell script that sets up the Sablime environment on a UNIX server. See the Sablime *Administrator's Manual* for details.

E

External Communications Network

The network used by the Sablime system to communicate with outside *projects* or other machines.

F

File

Binary and non-binary files, test scripts, and document input files are all considered files. Changes to files are generated outside of the Sablime system and are documented and controlled by Sablime in the Source Database.

G

Generic

A version of the product that has been or may be released and must be maintained. Each generic is maintained separately. The setup generic is the generic specified when the *dot sablime* command was issued.

Generic Administrator (GA)

Administrator with the authority to accept an MR for a generic and to assign a developer to study the MR or make changes in response to it. Certain commands are restricted to the GA.

Global Database (GDB)

Contains data that is used across the entire Sablime instance (personnel information, product information, etc.).

Group Name

The name assigned to a group in Sablime using the *setgroup* command.

H

Hideable Field

A field that can be removed from Sablime windows. If the field is not displayed, no data will be gathered in the field and the field in the database will always be blank.

Host Machine

A machine linked to zero or more satellite machines through a network that allows users of other machines to execute Sablime commands sharing common Sablime databases located on the host.

I

Inactive Database (IDB)

Contains all the information that is no longer required for the current work being done on a *product*. When an MR is killed or completed for all generics or all work on a generic is completed, all the information about it in the ADB can be moved to the product's IDB for historic purposes.

Instance

A set of Sablime commands and programs and its databases that supports development and maintenance of various *products* and that is owned by a single *sablime* login.

M

Modification Request Administrator (MRA)

The person or persons responsible for administering newly created MRs and completed MRs. The MRA has responsibility for the total MR including the determination of the *generics* to which it applies but not for the activity of the MR within a generic. Certain commands are restricted to use by the MRA.

MR

Modification Request—The description of an enhancement or of a problem in the existing *product*. In the Sablime system, an MR is required to request or make changes to the controlled product.

MR History File

A file created and maintained by Sablime if the History File flag in the ADM relation is set to **y**. Every time a Sablime command that affects an MR is executed, a record is written to this file.

N

Node

A set of files arranged in a UNIX system directory structure. The base of the node is the topmost directory in the structure. Any file that can be reached as a descendant from the base is contained in the node. A file in a node is identified by a relative directory path describing the path from the base of the node to the file.

P

Product

Any combination of software, firmware, hardware, or documentation that is eventually generated for use by customers. A Sablime *instance* supports the development and maintenance of one or more products.

Product Directory Structure

The organization of the directories and the files associated with a generic in a Sablime *product*. The highest-level directory associated with the structure is referred to as the base or the product *node*. The remaining directories must be at a lower level and reachable from the base. References to any directory within the product directory is considered to be the relative directory path from the base of the node.

Project

The MR trouble-reporting system with which a Sablime product communicates through the External MR Communications feature; it can be another Sablime project or a different type of trouble-reporting system.

PTS ID

Personnel Tracking System ID—the Sablime system identifier that allows a user access to a *product*.

R

Relation

A directory in the GDB, ADB, or IDB used to store tuple files containing lines of data (records) to be accessed by Sablime commands for information about MRs, generics, and source files.

Record

A line of data in a tuple file in a directory (relation) in the GDB, ADB, or IDB to be accessed by Sablime for information about MRs, *generics*, and *files*.

Request Severity

The impact of a fault on product operation as judged by the MR creator. Severity ratings are defined below.

- n Severity 1—The basic service provided by the product is interrupted.
- n Severity 2—The basic service provided by the product is degraded; some functions may not be available or may be inadequate.
- n Severity 3—Functional problems cause inconvenience to users, administrators, or maintenance personnel; work-arounds exist or the software recovers on its own but the problem will be fixed.
- n Severity 4—A minor deficiency exists that is of little consequence.

S

Satellite Machine

A machine linked to a host machine through a network that allows users of the machine to access Sablime commands sharing common Sablime databases located on the host.

Source Administrator (SA)

The person or persons responsible for maintenance of the *Product Directory Structure* and for the administration of the *files* associated with a product.

Source Database (SDB)

The collection of version-controlled files placed under Sablime for your product.

SBCS

Source and Binary Control System, used by Sablime to control versions of binary and non-binary files in response to MRs.

SCCS

Source Code Control System, used by Sablime to control versions of non-binary files in response to MRs.

T

Template

An ASCII text file describing a format or guide designed to promote project consistency in documentation and programming structures.

Trace File

A file created and maintained by Sablime if the Trace Flag in the ADM relation is set to y. Every time a user executes a Sablime *command*, a trace record is written in the trace file with the same name as that user's PTS ID. A second trace file exists on the Windows client; it tracks the command execution from the Windows interface (see Options>Environment).

Tuple File

A file in a directory (relation) in the GDB, ADB, or IDB used to containing lines of data (records) to be accessed by Sablime for information about MRs, *generics*, and *files*.

V

Value

Data entered in a field.

Symbols

- .BIN File, 4-3
 - IPC, 2-46, 2-49, 2-52, 2-55, 2-58
- .EMR File, 4-3
 - IPC, 2-45, 2-49, 2-52, 2-55, 2-58
- .usrld File, 2-5, 2-9
 - permissions, 2-9
- __ACCT SABLIME-Level Reserved Group, A-2
- __ADB Run-time Expansion Keyword, 3-26
- __ADBREL SABLIME-Level Reserved Group, A-3
- __ADEP SABLIME-Level Reserved Group, A-3
- __ADMIN SABLIME-Level Reserved Group, A-3, A-4
- __AMODE SABLIME-Level Reserved Group, A-4
- __BINARY SABLIME-Level Reserved Group, A-4
- __BROFC SABLIME-Level Reserved Group, A-5
- __CASTYPE SABLIME-Level Reserved Group, A-5
- __CLASSPDI SABLIME-Level Reserved Group, A-5
- __CSEV SABLIME-Level Reserved Group, A-5
- __DB SABLIME-Level Reserved Group, A-5
- __DBAI SABLIME-Level Reserved Group, A-5
- __DELAY SABLIME-Level Reserved Group, A-6
- __DEVSEV SABLIME-Level Reserved Group, A-6
- __DOCMGMT SABLIME-Level Reserved Group, A-6
- __DOCUMENT SABLIME-Level Reserved Group, A-6
- __DVCT SABLIME-Level Reserved Group, A-6
- __EFFID Run-time Expansion Keyword, 3-26
- __EMR_RNAME SABLIME-Level Reserved Group, A-6
- __ESNET SABLIME-Level Reserved Group, A-6
- __FCBTMPS SABLIME-Level Reserved Group, A-7
- __FCNDBA SABLIME-Level Reserved Group, A-7
- __FCNSNP, A-7
- __FCNTMPL SABLIME-Level Reserved Group, A-7
- __FMTS SABLIME-Level Reserved Group, A-7
- __FTYPE SABLIME-Level Reserved Group, A-7
- __GDBREL SABLIME-Level Reserved Group, A-7
- __GENERIC Run-time Expansion Keyword, 3-26
- __GETVSTAT SABLIME-Level Reserved Group, A-8
- __GRP_RNAME SABLIME-Level Reserved Group, A-8
- __GRPTYPE SABLIME-Level Reserved Group, A-8
- __HMI SABLIME-Level Reserved Group, A-8
- __IDB Run-time Expansion Keyword, 3-26
- __IDBREL SABLIME-Level Reserved Group, A-9
- __INCLDEP SABLIME-Level Reserved Group, A-9
- __INFORET SABLIME-Level Reserved Group, A-9
- __MCB Run-time Expansion Keyword, 3-26
- __MM SABLIME-Level Reserved Group, A-9
- __MR_RNAME SABLIME-Level Reserved Group, A-10
- __MRMGMT SABLIME-Level Reserved Group, A-11
- __MvF_RNAME SABLIME-Level Reserved Group, A-11
- __PROD Run-time Expansion Keyword, 3-26
- __PROMPT SABLIME-Level Reserved Group, A-12
- __PTSID Run-time Expansion Keyword, 3-26
- __PTYPE SABLIME-Level Reserved Group, A-12
- __QMMGMT SABLIME-Level Reserved Group, A-12
- __QREL SABLIME-Level Reserved Group, A-13
- __QUEUES SABLIME-Level Reserved Group, A-13
- __RA SABLIME-Level Reserved Group, A-13
- __REALID Run-time Expansion Keyword, 3-26
- __RELDIR Run-time Expansion Keyword, 3-27
- __REPORTCLASS SABLIME-Level Reserved Group, A-14
- __REVACT SABLIME-Level Reserved Group, A-14
- __RTE SABLIME-Level Reserved Group, A-14
- __SABREL SABLIME-Level Reserved Group, A-14
- __SABRPN SABLIME-Level Reserved Group, A-14
- __SABSEV SABLIME-Level Reserved Group, A-14
- __SABSYS SABLIME-Level Reserved Group, A-14
- __SABTYP SABLIME-Level Reserved Group, A-14
- __SCRATT SABLIME-Level Reserved Group, A-15
- __SDB Run-time Expansion Keyword, 3-27
- __SEND SABLIME-Level Reserved Group, A-15
- __SETREL SABLIME-Level Reserved Group, A-15
- __SRC_RNAME SABLIME-Level Reserved Group, A-15
- __SRCMGMT SABLIME-Level Reserved Group, A-15
- __SRPSTAT SABLIME-Level Reserved Group, A-16
- __STSEV SABLIME-Level Reserved Group, A-16
- __TODAY Run-time Expansion Keyword, 3-27
- __VCT SABLIME-Level Reserved Group, A-16
- __VPDEV Run-time Expansion Keyword, 3-27
- __XMRMGMT SABLIME-Level Reserved Group, A-16
- __YN SABLIME-Level Reserved Group, A-18
- __YNALL SABLIME-Level Reserved Group, A-18
- __YNN SABLIME-Level Reserved Group, A-18
- __YNQN, A-18
- __YNY SABLIME-Level Reserved Group, A-18
- __ATYPE Product-Level Reserved Group, B-2
- __CMRCS Product-Level Reserved Group, B-2
- __DEFCODE Product-Level Reserved Group, B-2
- __DOC_FLTYPE Product-Level Reserved Group, B-2
- __DOCUCLASS Product-Level Reserved Group, B-2
- __ENHTYPE Product-Level Reserved Group, B-2
- __ETRCS Product-Level Reserved Group, B-2
- __FAVED Product-Level Reserved Group, B-2
- __FIRM_FLTYPE Product-Level Reserved Group, B-3
- __FIRMCLASS Product-Level Reserved Group, B-3
- __HARD_FLTYPE Product-Level Reserved Group, B-3
- __HARDCLASS Product-Level Reserved Group, B-3
- __ICAT Product-Level Reserved Group, B-3
- __INITYPE Product-Level Reserved Group, B-3
- __KILLCODE Product-Level Reserved Group, B-3
- __LOC Product-Level Reserved Group, B-4
- __MODTYPE Product-Level Reserved Group, B-4
- __NOCHCODE Product-Level Reserved Group, B-5
- __ODPGRP Product-Level Reserved Group, B-5
- __OSRCS Product-Level Reserved Group, B-5
- __PDGRP Product-Level Reserved Group, B-5
- __PDRCS Product-Level Reserved Group, B-5
- __PIGRP Product-Level Reserved Group, B-5
- __PMRCS Product-Level Reserved Group, B-5
- __RCGRP Product-Level Reserved Group, B-6

_REJCODE Product-Level Reserved Group, B-6
 _RELS Product-Level Reserved Group, B-6
 _RESCODE Product-Level Reserved Group, B-6
 _RPRCS Product-Level Reserved Group, B-6
 _SABSTAT Product-Level Reserved Group, B-6
 _SITES Product-Level Reserved Group, B-7
 _SOFT_FLTYPE Product-Level Reserved Group, B-7
 _SOFTCLASS Product-Level Reserved Group, B-7
 _SYS Product-Level Reserved Group, B-7

A

accept Command, 3-11, 3-27, 3-28, 3-36, 5-52, A-6, B-2, B-3, B-4, B-7
 Active Data Base, 4-25, 4-26, 5-1
 ADM relation, 5-29
 audits, D-1, D-13, D-15
 CAS relation, 5-30
 COM relation, 5-31, 5-32
 CRIT relation, 5-33
 DBLOCK directory, 5-15
 DEP relation, 5-34
 description, 5-14
 directory structure, 5-14
 EMG relation, 5-35
 EMR relation, 5-36
 FTD relation, 5-38
 G relation, 5-41
 GRP relation, 5-42
 GRPM relation, 5-43
 GS relation, 5-44
 GT relation, 5-45
 HC relation, 5-46
 MD relation, 5-47
 MG relation, 5-48
 MR relation, 5-52
 MRS relation, 5-53
 MRX relation, 5-54
 MS relation, 5-55
 ORG relation, 5-56
 PDI relation, 5-58
 relations, 5-23
 UMS relation, 5-61
 ADB
 changing, 6-69
 see Active Data Base
 addgen Command, 3-11, 4-4, 4-9, 5-41, 5-45, 6-1, 6-67, A-2, A-18
 description, 6-5
 addgsr Command, 4-14, 4-21, 5-44, 5-47, 5-55, A-5
 Adding a Generic, 4-4, 6-48
 Adding a Product, 4-1
 addisrc Command, 5-44, 5-47, 5-55, 6-59, A-4, A-16
 ADM Relation, 2-23, 5-23, 6-56, 6-57

 description, 5-29
 ADM Subcommand, 4-36, 6-29, 6-31, 6-45, A-3, A-6
 description, 6-56
 Approval Team, 2-20, 6-6
 assign Command, A-6
 Audit Programs, 4-28
 Automatic Assignment Criteria, 6-62
 Automatic MR Assignment, 2-22
 Automatic MR Dependencies, 2-23
 Automatic MR Routing, 2-22
 Automatic Routing Criteria, 6-62

B

Binary Files, 4-19

C

CAS Relation, 5-23, 6-59
 description, 5-30
 CAS Subcommand, 3-11, 3-15, A-5
 description, 6-59
 Cascade Effect
 customizing, 3-10
 changing, 2-6
 closegen Command, 6-1
 description, 6-10
 closemr Command, 5-52
 CMS, 4-30, 4-31
 COM Relation, 5-23, 6-10
 description, 5-31, 5-32
 Commands
 host only, 6-4
 commit Command, 5-31, A-7
 Corruption
 data base, 4-28
 create Command, 3-8, 3-14, 3-32, 3-34, 3-35, 3-42, 3-45, 3-46, 3-47, 3-48, 3-49, 3-50, 3-51, 3-52, 4-27, 4-35, 5-52, 5-56, 6-32, A-5, B-3, B-5, B-6, B-7
 UDFs, 3-40
 CRIT Relation, 5-23, 6-10, 6-62
 description, 5-33
 CRIT Subcommand, A-13
 description, 6-62

D

dbawarn File, C-2
 messages, C-1

dbcross Command, 4-29, 6-18, 6-26, D-1, D-13
description, 6-12, 16
dbdelta Command, 4-30, 6-18, 6-26
description, 6-13
dbedit Command, 4-28, 6-1, 6-14, 6-18, A-5
description, 4-28, 6-14
dbhash Command, 5-1, 6-2, 6-18
DBLOCK Directory, 5-5, 5-15, 5-16
dbstart Command, 6-2, 6-18
description, 6-17
dbstop Command, 4-23, 4-28, 6-2, 6-14, 6-17, 6-18
description, 6-18
dbxcross Command, 4-30, 6-18, 6-26
description, 6-20
defer Command, 5-54, B-2
Deleting a File, 4-23
DEP Relation, 5-23
description, 5-34
depend Command, 5-34
Dependencies
see MR Dependencies
description Node, 5-20
Diagnostic Messages, C-1
DIR Directory, 5-6
Directories
recover, 4-25
Directory Structure
adding a directory, 4-13
Display Flag, 3-19
dot sublime Command, 2-26, 2-30

E

edget Command, 5-47, 5-55
edput Command, 5-34, 5-47
dependencies, 6-58
EMG Relation, 5-23
description, 5-35
EMR Relation, 5-24
description, 5-36
Error Messages, 4-35
satellite, C-17
see also Diagnostic Messages
ES Relation, 6-65
description, 5-8
ES Subcommand, 2-46, 2-47, 2-49, 2-51, 2-53, 2-54, 2-56,
2-57, 2-59, 2-60, 4-3, A-6
description, 6-65
execdm.sh Script, 2-61
External Keyword, 3-25
External MR Communications, 2-44
IPC, 2-45
TCP/IP, 2-54
uucp, 2-58

External MR Communications Feature, 6-20
external project communication, 6-65
network types, 6-66

F

fcreate Command, 3-8, 3-11, 3-14, 3-45, 3-46, 3-58, 4-16,
4-27, 5-52, 5-56, 6-32, A-5, B-2, B-5, B-6, B-7
UDFs, 3-40
File Permissions
.usrld file, 2-9
Files
binary, 4-19
non-binary, 4-19
restoring, 4-24
FILES Directory, 6-17, 6-45
description, 5-20
glob_solution, 5-20
hcode, 5-20
mrcommit, 5-20
mrhistory, 5-20
rejection, 5-21
resolution, 5-21
SAB.stopfile, 5-20
solution, 5-21
spawnnotes, 5-21
templates, 5-22
trace, 5-22
find Command, 4-15
FSDB Variable, 4-34
ftd Command, 3-1, 3-17, 3-22, 3-25, 3-32, 3-33, 3-35, 3-36,
3-42, 3-43, 3-44, 3-45, 3-50, 3-51, 5-38, 6-2, 6-58,
A-7, A-12, A-15
description, 6-21
FTD Relation, 2-17, 3-29, 5-24
description, 5-38
FTD Tuple Records, 3-18
functions, 2-6
FZ, 5-24

G

G Relation, 2-23, 5-24, 6-10
description, 5-41
GDB
see Global Data Base
Generic
adding, 4-4, 6-48
naming, 4-5
Generic Identifier
description, 5-63

- numbering system, 6-5
- Generic Names, 6-7
- getversion Command, A-5, A-8, A-9
- GID
 - see Generic Identifier
- glob_solution Node, 5-20
- Global Data Base, 4-2, 4-35, 5-1
 - DBLOCK directory, 5-5
 - description, 5-5
 - DIR directory, 5-6
 - ES relation, 5-8
 - PR relation, 5-9
 - PRX relation, 5-10
 - PTS relation, 5-11
 - rd directory, 5-6
 - rq directory, 5-6
 - sd directory, 5-6
 - sq directory, 5-6
 - tmp directory, 5-6
 - TR relation, 5-13
- Group Names
 - test teams, 2-18
- GRP Relation, 2-23, 3-29, 5-24
 - description, 5-42
- GRPM Relation, 2-23, 3-29, 5-24
 - description, 5-43
- GS Relation, 5-24, 6-10
 - description, 5-44
- GT Relation, 2-23, 5-25, 6-10
 - description, 5-45

H

- Hash Field, 5-23
- Hashing, 5-1, 5-2
- HC Relation, 5-25, 6-10
 - description, 5-46
 - UDFs, 3-40
- hcode Command, 3-46, 5-46
 - UDFs, 3-40
- hcode Node, 5-20
- HCODE Relation
 - UDFs, 3-46
- Hideable Flag, 3-19
- HMI, 3-23
- Host Machine, 2-37
- hotline.ck Script, 2-39, 4-30, D-18
 - description, 6-26

I

- IDB
 - see Inactive Data Base
- Inactive Data Base, 4-2, 4-26, 5-1, 6-10
 - CAS relation, 5-30
 - changing, 6-69
 - COM relation, 5-31, 5-32
 - CRIT relation, 5-33
 - DBLOCK directory, 5-16
 - DEP relation, 5-34
 - description, 5-14
 - EMG relation, 5-35
 - EMR relation, 5-36
 - G relation, 5-41
 - GS relation, 5-44
 - GT relation, 5-45
 - HC relation, 5-46
 - MD relation, 5-47
 - MG relation, 5-48
 - MR relation, 5-52
 - MRS relation, 5-53
 - MRX relation, 5-54
 - MS relation, 5-55
 - ORG relation, 5-56
 - PDI relation, 5-58
 - Relations, 5-23
- initsab Script, 2-6, 4-36, 5-14, 6-28
 - information needed for, 2-8
 - limits, 2-18
- In-Process Metrics Data, 2-22
- Instance
 - machine requirements for, 2-3
 - space needed, 2-2
- Integration Test Team, 2-20
- Internal Product, 6-68
- IP Address
 - TCP/IP, 2-43
- IPC, 2-44, 6-67
 - external MR communications, 2-45
- itpass Command, 6-67

K

- killmr Command, 5-52, B-3

L

- License

- .usrld file, 2-5, 2-9
- .usrld file permissions, 2-9
- error messages, C-15
- listmsgs Command, A-7, A-13
- Local Control Bin, 4-3

M

- Mail, 2-22, 4-36, 4-39
 - consolidation, 4-37
 - flag, 6-31
- Major Programming Language
 - how to change, 2-14
- Mandatory Flag, 3-19
- Master Control Bin, 4-3
- MCB
 - changing, 6-68
- MD Relation, 4-30, 5-25
 - description, 5-47
- MG Relation, 5-26
 - description, 5-48
 - UDFs, 3-40, 3-46
- mhist Command, 6-2, 6-56
 - description, 6-29
- mmail Command, 4-37, 6-2, 6-56
 - description, 6-31
- mode
 - multi-machine, 6-68
 - single-machine, 6-68
- Moving a File, 4-23
- MR Assignment, 2-22
- MR Dependencies, 2-23, 6-58
 - SBCS files, 2-23
- MR History, 6-29
- MR History File
 - QA data and, 2-22
 - recommendation, 2-22
- MR Life Cycle, 6-29
- MR Number
 - setting, 6-58
- MR Prefix
 - initsab and, 2-13
- MR Reassignment, 2-22
- MR Relation, 4-27, 5-26, 6-32
 - description, 5-52
 - UDFs, 3-40, 3-45
- MR Routing, 2-22
- MR States, 4-5, 6-7
- mrcommit Node, 5-20
- mredit Command, 3-14, 3-45, 3-46, 3-58, 5-52, 6-2, A-5,
B-5, B-6, B-7
 - description, 4-27
 - UDFs, 3-40
- mrghedit Command, 3-46, 6-2, A-5, B-2, B-5, B-6

- description, 4-27, 6-38
- UDFs, 3-40
- mrhistory Node, 5-20
- MRS Relation, 5-27
 - description, 5-53
- MRX Relation, 5-27
 - description, 5-54
- MS Relation, 5-27
 - description, 5-55
- mtrace Command, 6-2, 6-56
 - description, 6-45
- Multi-Machine Feature, 2-37
 - initsab and, 2-11
 - networks, 2-38
- mvgen Script, 6-3, 6-6
 - description, 6-47

N

- Network Codes, 2-29, 2-34
- Networks, 2-38
 - communication with external projects, 6-66
- newgen Script, 4-4, 6-3
 - description, 4-4, 6-48
- NFS/RFS, 2-34, 2-38, 6-67
 - setup, 2-40
- nochange Command, B-5
- nohist Command, 6-2, 6-56
 - description, 6-29
- nomail Command, 4-37, 6-2, 6-56
 - description, 6-31
- Non-Binary Files, 4-19
- notrace Command, 6-2, 6-56
 - description, 6-45

O

- ORG Relation, 4-27, 5-27, 6-32
 - description, 5-56

P

- PDEP, 5-27
- pdi Command, 3-46, 5-58, A-5
 - UDFs, 3-40
- PDI Relation, 5-28, 6-10
 - description, 5-58
 - UDFs, 3-40, 3-46
- Permissions, 6-26

pickmsg.sh Script, 2-61
 Popup Selection Windows, 3-21, 3-29, 6-22
 PR Relation, 4-2, 6-68
 description, 5-9
 PR Subcommand, 2-39, A-9
 description, 6-68
 Pre-Approval Team, 2-20
 preapprove Command, 6-67
 Pre-Integration Test Team, 2-20
 preitpass Command, 6-67
 prestpass Command, 6-67
 Pre-System Test Team, 2-20
 primsdb Command, 6-3, 6-59, B-2, B-3, B-7
 description, 6-49
 Product Directory Structure, 2-7
 Product-Level Reserved Groups, ??-B-7
 Products
 adding, 4-1
 internal, 6-68
 Project Customization
 category, 3-8
 PRX Relation, 4-2, 6-50, 6-60, 6-69
 description, 5-10
 PRX Subcommand
 description, 6-60, 6-69
 pts Command, 5-11, A-4, A-6, A-8, B-2, B-4
 PTS ID
 format, 2-7
 PTS Relation, 6-50
 description, 5-11

Q

QA Team, 2-21
 qmr Command, 5-36
 Quality Assurance Feature, 6-69
 query Command, 3-4, 3-24, 3-25, 3-36, 3-42, 3-45, 3-46,
 3-52, 3-53, 3-54, 3-55, 3-56, 3-57, 4-2, 5-2, 6-18,
 6-22, A-3, A-5, A-6, A-7, A-9, A-13, A-14, A-16
 UDFs, 3-40

R

rcv_msgs Command, 6-66
 rd Directory, 5-6
 Records
 description, 5-1, 5-2
 recover Directory, 4-25
 reject Command, B-6
 rejection Node, 5-21
 Relations

ADM, 5-23, 5-29, 6-56, 6-57
 CAS, 5-23, 5-30, 6-59
 COM, 5-23, 5-31, 5-32, 6-10
 commands affecting, 5-2, 5-7, 5-23
 CRIT, 5-23, 5-33, 6-10, 6-62
 DEP, 5-23, 5-34
 description, 5-1
 EMG, 5-23, 5-35
 EMR, 5-24, 5-36
 ES, 5-8, 6-65
 FTD, 5-24, 5-38
 G, 5-24, 5-41, 6-10
 Global Data Base, 5-7
 GRP, 5-24, 5-42
 GRPM, 5-24, 5-43
 GS, 5-24, 5-44, 6-10
 GT, 5-25, 5-45, 6-10
 HC, 5-25, 5-46, 6-10
 MD, 4-30, 5-25, 5-47
 MG, 5-26, 5-48
 MR, 4-27, 5-26, 5-52, 6-32
 MRS, 5-27, 5-53
 MRX, 5-27, 5-54
 MS, 5-27, 5-55
 ORG, 4-27, 5-27, 5-56, 6-32
 PDI, 5-28, 5-58, 6-10
 PR, 4-2, 5-9, 6-68
 PRX, 4-2, 5-10, 6-50, 6-60, 6-69
 PTS, 5-11, 6-50
 TR, 5-13
 UMS, 5-28, 5-61
 Renaming a File, 4-23
 report Command, 3-4, 3-22, 3-24, 3-25, 3-31, 3-42, 3-45,
 3-46, 3-58, 5-2, 6-18, 6-22, A-5, A-6, A-8, A-10,
 A-11, A-14, A-15
 UDFs, 3-40, 3-41
 Reserved Groups
 comments, 3-31
 default, 3-6, 3-7, 3-8, 3-10, 3-31
 resolution Node, 5-21
 Restoring a File, 4-24
 Restoring Files, 4-24
 review Command, 3-8, 3-11, 3-14, 3-41, 3-45, 3-46, 3-58,
 4-27, 5-35, 5-36, 5-52, 5-56, 6-32, A-5, A-14, B-5,
 B-6, B-7
 UDFs, 3-40
 rq Directory, 5-6
 Run-time Expansion Keywords, 3-25-3-27

S

SAB.stopfile, 5-20, 6-17, 6-18
 sabDIRF Variable, 2-15
 product directory structure, 2-7

- sabDKFB Variable, 2-27, 2-43, 2-44
 - sabGDB Variable, 2-33
 - IPC, 2-45, 2-49, 2-52
 - TCP/IP, 2-55
 - uucp, 2-58
 - sabHOST Variable, 2-28
 - IPC, 2-45, 2-49, 2-52
 - TCP/IP, 2-55
 - uucp, 2-58
 - sabLCB Variable, 2-28
 - SABLIME Daemon, 2-41
 - SBCS path, 2-41
 - sablime Login
 - group memberships, 2-18
 - on NFS/RFS, 2-40
 - sablime Script, 2-31
 - sablime.sh Script, 2-31, 4-3
 - SABLIME-Level Reserved Groups, 3-29–??
 - see individual group names
 - sabMCB Variable, 2-28
 - IPC, 2-45, 2-49, 2-52
 - TCP/IP, 2-55
 - uucp, 2-58
 - sabMFR Variable, 2-28
 - sabNET Variable, 2-29, 2-34, 2-40
 - TCP/IP, 2-40
 - sabPROD Variable, 2-34
 - IPC, 2-45, 2-49, 2-52
 - TCP/IP, 2-55
 - uucp, 2-58
 - sabPTS Variable, 2-29
 - sabSERV Variable, 2-29
 - TCP/IP, 2-40
 - sabSPOOL Variable, 2-30, 2-34
 - sabTMP Variable, 2-30
 - sabVAR Environment Variable, 2-40
 - sabVAR Variable, 2-26, 2-31
 - sabVER Variable, 2-26, 2-31, 2-32
 - sabVHELP Variable, 2-30
 - SADMIN Variable, 4-34
 - Satellite Machine, 2-37
 - SBCS, 4-19, 6-59
 - SBCS Path
 - SABLIME daemon, 2-41
 - SCCS, 4-19, 6-59
 - screate Command, 6-3, A-3, A-4, A-6, A-9, A-11, A-12, A-14, A-15
 - description, 6-50
 - Screen Label
 - UDFs, 3-45
 - sd Directory, 5-6
 - SDB
 - changing, 6-69
 - see Source Data Base
 - sendmsgs Command, 3-33, A-15
 - setgroup Command, 3-1, 3-4, 3-6, 3-7, 3-9, 3-12, 3-13, 3-15, 3-31, 3-32, 3-33, 3-35, 3-36, 3-43, 3-44, 3-49, 4-4, 4-9, 5-42, 5-43, 6-60, A-8
 - setnode Command, 4-9, 4-14, 4-15
 - setperm Command
 - description, 6-54
 - setperm Script, 2-39
 - setrel Command, 2-39, 2-46, 2-47, 2-49, 2-51, 2-53, 2-54, 2-56, 2-57, 2-59, 2-60, 3-11, 3-14, 3-15, 4-3, 4-36, 5-8, 5-9, 5-10, 5-30, 5-33, 6-3, 6-29, 6-31, 6-45, A-3, A-5, A-6, A-7, A-9, A-12, A-13, A-15, A-18
 - description, 6-55
 - sget Command, A-5
 - sh2x Script, 2-37, 4-3, 4-11
 - SNAP, 5-28
 - Software Quality Assurance, 6-69
 - solution Node, 5-21
 - source Command, 4-24, 6-3
 - description, 4-23, 6-71
 - Source Data Base, 4-2, 4-4, 5-1, 6-48
 - description, 5-62
 - estimating size of, 2-2
 - Space Requirements, 2-2
 - spacecheck Script, 4-30, 4-33, 4-34
 - spawnmr Command, 4-27, 5-53, 6-32, A-5, B-2, B-6, B-7
 - spawnnotes Node, 5-21
 - Special Characters
 - generic name, 2-14
 - sq Directory, 5-6
 - sqin Command, B-6
 - sreport Command, 6-3, A-14, A-16
 - description, 6-75
 - ssql Command, 5-2
 - STOPINODE Variable, 4-34
 - STOPSPACE Variable, 4-34
 - stpass Command, 6-67
 - study Command, 5-54, A-16
 - su Command, 2-9
 - submit Command, 3-46, B-2, B-5, B-6
 - UDFs, 3-40
 - suucpdm Command, 2-61
 - System Test Team, 2-20
-
- T**
- TCP/IP, 2-38, 2-45, 6-67
 - external MR communications, 2-54
 - setup, 2-40
 - tcplisten Daemon, 2-41, 2-42
 - template Command, A-7
 - Templates, 2-6
 - changing, 3-38
 - files, 3-38
 - templates Node, 5-22
 - tempset Command, A-7

- Test Teams, 2-19, 2-20, 2-21
 - approval, 6-6
- tmp Directory, 5-6
- TMPDIR Variable, 4-29
- TR Relation
 - description, 5-13
- trace Directory, 6-45
- Trace File, 2-22
- Trace Flag, 6-45
- trace Node, 5-22
- Tuples
 - corruption, 4-28
 - description, 5-1
- TZ Environment Variable, 2-43

U

- UDFs
 - see User-Definable Fields
- UMS Relation, 5-28
 - description, 5-61
- unedget Command, 5-47
- User-Definable Fields, 3-40, 6-22, 6-43
 - by relation, 3-41
 - default data, 3-43
 - EMR, 3-41
 - maximum values, 3-43
 - VHELP, 3-45
- UUCP, 6-67
- uucp, 2-45
 - Daemon, 2-61
 - External MR Communications, 2-58

V

- Variable File, D-18
- Variables, D-18
 - definition format, 2-26
 - FSDB, 4-34
 - sabDKFB, 2-27
 - sabGDB, 2-33, 2-45, 2-49, 2-52, 2-55, 2-58
 - sabHOST, 2-28, 2-45, 2-49, 2-52, 2-55, 2-58
 - sabLCB, 2-28
 - sabMCB, 2-28, 2-45, 2-49, 2-52, 2-55, 2-58
 - sabMFR, 2-28
 - sabNET, 2-29, 2-34, 2-40
 - sabPROD, 2-34, 2-45, 2-49, 2-52, 2-55, 2-58
 - sabPTS, 2-29
 - sabSERV, 2-29, 2-40
 - sabSPOOL, 2-30, 2-34
 - sabTMP, 2-30

- sabVAR, 2-26, 2-31
- sabVER, 2-26, 2-31, 2-32
- sabVHELP, 2-30
- SADMIN, 4-34
- STOPINODE, 4-34
- STOPSPACE, 4-34
- TMPDIR, 4-29
- TZ, 2-43

W

- Worksheets, E-1

X

- xsablime Script, 2-31
- xsablime.sh Script, 2-31, 4-3, 4-4, 4-11, 6-48, D-18