

Software through Pictures[®] Core

Release 7.1

StP Administration Guide



Aonix

Software through Pictures Core StP Administration Guide

Release 7.1

Part No. 10-MN103/ST7100-01298/001

December 1998

Aonix reserves the right to make changes in the specifications and other information contained in this publication without prior notice. In case of doubt, the reader should consult Aonix to determine whether any such changes have been made. The software described in this document is furnished under a license and may be used or copied only in accordance with the terms of such license.

Copyright © 1998 by Aonix.™ All rights reserved.

This publication is protected by Federal Copyright Law, with all rights reserved. Unless you are a licensed user, no part of this publication may be reproduced, stored in a retrieval system, translated, transcribed, or transmitted, in any form, by any means, without prior written permission from Aonix. **Licensed users may make copies of this document as needed solely for their internal use—as long as this copyright notice is also reproduced.**

Trademarks

Aonix and the Aonix logo are trademarks of Aonix. ObjectAda is a trademark of Aonix. Software through Pictures is a registered trademark of Aonix. All rights reserved.

HP, HP-UX, and SoftBench are trademarks of Hewlett-Packard Inc. Sun and Solaris are registered trademarks of Sun Microsystems, Inc. SPARC is a registered trademark of SPARC International, Inc. Products bearing SPARC trademarks are based on an architecture developed by Sun Microsystems, Inc. Open Software Foundation, OSF, OSF/Motif, and Motif are trademarks of the Open Software Foundation, Inc. UNIX is a registered trademark in the United States and other countries, exclusively licensed through X/Open Company, Ltd. X Window System is a trademark of X Consortium, Inc. Windows NT is a trademark and Microsoft and Windows are registered trademarks of Microsoft Corporation in the United States and other countries. Adobe, Acrobat, the Acrobat logo, and PostScript are trademarks of Adobe Systems, Inc. Sybase and the Sybase logo are registered trademarks of Sybase, Inc. Adaptive Server, Backup Server, Client-Library, DB-Library, Open Client, PC Net Library, SQL Server, SQL Server Manager, SQL Server Monitor, Sybase Central, SyBooks, System 10, and System 11 are trademarks of Sybase, Inc.



© 1998 Aonix. All rights reserved.

World Headquarters

595 Market Street, 12th Floor
San Francisco, CA 94105
Phone: (800) 97-AONIX
Fax: (415) 543-0145
E-mail: info@aonix.com
<http://www.aonix.com>

Table of Contents

Preface

Intended Audience	ix
Typographical Conventions	x
Contacting Aonix.....	x
Reader Comments	x
Technical Support	xi
Websites.....	xi
Related Reading	xii

Chapter 1 Introduction

StP Overview	1-1
Definitions	1-1
Who Is the StP Administrator?	1-5
StP Administrator	1-5
System Administrator's Password.....	1-6
System Owner	1-6
Prerequisites for StP Administration	1-7
StP Is Installed.....	1-7
You Can Use the StP Desktop	1-7
You Have the Appropriate Privileges.....	1-7
Starting StP	1-8
Starting the Desktop.....	1-8

Setting the Current Project and System.....	1-8
Using StP Administration Commands	1-9
Commands to Administer the Repository	1-10
Commands to Administer Individual Systems.....	1-11

Chapter 2 StP Users

What Is an StP User?	2-1
StP User Access Security.....	2-2
StP User Names.....	2-2
Repository Manager Users	2-4
Adding Sybase Repository Manager Users	2-4
Deleting Sybase Repository Manager Users.....	2-5
Granting Sybase Users System Creation Privileges.....	2-5
Revoking Sybase System Creation Privileges.....	2-6
Listing Current Sybase Repository Manager Users.....	2-7
Listing All Repository Manager Users	2-7
Assigning User Passwords	2-7
User Environments	2-8
How ToolInfo Files Work.....	2-8
ToolInfo Variables Versus Environment Variables.....	2-9
Understanding ToolInfo Variable Syntax.....	2-11
Creating User ToolInfo Variables.....	2-13
Finding the Current ToolInfo Value	2-13
Modifying the Environment for Individual Users.....	2-14
System Owners.....	2-15
File Permissions	2-15
System Names.....	2-16
Creating a System	2-16
Changing System Ownership	2-19
Commands Performed by System Owners.....	2-20

System Users	2-21
Privileges.....	2-21
Adding Users to a System	2-22
Deleting Users from a System.....	2-22
Changing a User's Privileges.....	2-23
Commands Performed by System Users	2-23
Passwords	2-27
Passwords for Sybase.....	2-28
Repository Manager Security Levels	2-29
Password Syntax.....	2-31
Changing the System Administrator's Password.....	2-32
Changing User Passwords	2-33
Troubleshooting Password Changes.....	2-34

Chapter 3 **Administering StP**

Configuring StP Interprocess Communications.....	3-1
The <i>msgd_host</i> ToolInfo Variable.....	3-3
Message Routing in a Single-User Environment	3-3
Message Routing in a Multi-User Environment	3-4
Running Message Router on a Nominated Host	3-5
Connecting to the StP Message Router.....	3-6
Troubleshooting Message Router Problems	3-7
Managing Sybase Database Placement	3-11
Creating Sybase Systems on Default Devices.....	3-12
Creating Systems on Specific Devices	3-12
Maintaining the Repository Manager.....	3-14
Routine Maintenance	3-14
Listing Repositories.....	3-15
Showing Space Available on Devices	3-15
Managing a System	3-17
Cleaning Up.....	3-17

Expanding a Microsoft Jet Repository	3-18
Expanding a Sybase Repository	3-20
Adding Files to the Repository	3-21
Deleting Repository Data	3-22
Destroying a System Repository	3-23
Destroying a System.....	3-23
Backup and Recovery.....	3-24
Backing Up a Sybase Repository to Dump Files.....	3-25
Restoring a Repository from Dump Files.....	3-26
Rebuilding the Repository from System Files	3-27

Chapter 4 Using StP Utility

StP Utility Subsystems	4-1
Accessing the StP Utility	4-2
Setting StP Utility Modes.....	4-2
Getting Help on StP Utility Commands.....	4-3
Accessing Subsystem Command Groups	4-4
Using Character Strings with StP Utility.....	4-4
Full Names.....	4-5
Patterns.....	4-5
StP Utility Main Level Commands.....	4-5
clean.....	4-6
copy.....	4-7
delete.....	4-8
editor.....	4-9
environment	4-9
files.....	4-10
filetype.....	4-11
fix content	4-11
force	4-12

help	4-12
interactive	4-12
lock.....	4-13
ls	4-13
project	4-14
quiet	4-14
quit.....	4-14
rename.....	4-15
schema.....	4-15
status.....	4-16
system.....	4-16
undef.....	4-17
verbose	4-18
Environment Subsystem.....	4-18
check	4-18
list.....	4-19
update.....	4-19
Locking Subsystem.....	4-19

Chapter 5 StP/DOORS Integration

Overview of StP/DOORS.....	5-1
How DOORS Stores StP Objects.....	5-2
Integrating StP with DOORS	5-3
Setting the User's Environment	5-3
Moving StP Files to DOORS	5-3
After Integrating StP and DOORS	5-4
Starting StP/DOORS.....	5-4
Navigating from StP to DOORS	5-4
Navigating from DOORS to StP	5-5
Updating Renamed Objects Exported to DOORS.....	5-6

Appendix A ToolInfo Variables

ToolInfo Variable Descriptions..... A-1

Appendix B Command Reference

Index

Preface

Software through Pictures (StP) is a family of multi-user integrated environments that supports the software development process, as well as maintenance and re-engineering of existing systems.

This manual is part of a complete StP documentation set of “Core” manuals, which includes: *Fundamentals of StP*, *Customizing StP*, *Query and Reporting System*, *Object Management System*, and *StP Administration*. Basic information about the StP user interface is documented in *Fundamentals of StP*.

Intended Audience

The *StP Administration* manual is intended for system administrators or users who are responsible for managing the StP environment. The main task of the StP system administrator is to ensure that the StP software runs properly so that others can use StP.

This manual assumes you are familiar with your operating system. You should also be familiar with the *Fundamentals of StP* manual for a basic understanding of the StP editors, including editing diagrams and tables.

Typographical Conventions

This manual uses the following typographical conventions:

Table 1: Typographical Conventions

Convention	Meaning
Palatino bold	Identifies commands.
<code>Courier</code>	Indicates system output and programming code.
Courier bold	Indicates information that must be typed exactly as shown, such as command syntax.
<i>italics</i>	Indicates pathnames, filenames, and ToolInfo variable names.
<angle brackets>	Surround variable information whose exact value you must supply.
[square brackets]	Surround optional information.
{curly brackets}	Surround information that can be set to a default value.

Contacting Aonix

You can contact Aonix using any of the following methods.

Reader Comments

Aonix welcomes your comments about its documentation. If you have any suggestions for improving *StP Administration Guide*, you can send email to docs@aonix.com.

Technical Support

If you need to contact Aonix Technical Support, you can do so by using the following email aliases:

Table 2: Technical Support Email Aliases

Country	Email Alias
Canada	support@aonix.com
France	customer@aonix.fr
Germany	stp-support@aonix.de
United Kingdom	stp-support@aonix.co.uk
United States	support@aonix.com

Users in other countries should contact their StP distributor.

Websites

You can visit us at the following websites:

Table 3: Aonix Websites

Country	Website URL
Canada	http://www.aonix.com
France	http://www.aonix.fr
Germany	http://www.aonix.de
United Kingdom	http://www.aonix.co.uk
United States	http://www.aonix.com

Related Reading

In addition to *StP Administration Guide*, other core-related StP manuals are as follows:

Table 4: Further Reading

For Information About	Refer To
Installing StP	<i>Installing StP for Windows NT</i>
Using the StP Desktop	<i>Fundamentals of StP</i>
Using standard StP editors and tools	<i>Fundamentals of StP</i>
Printing diagrams, tables, and reports	<i>Fundamentals of StP, Query and Reporting System</i>
Internal components of StP and how to customize an StP installation	<i>Customizing StP</i>
Using StP QRL Scripts	<i>Query and Reporting System</i>
StP Object Management System (OMS)	<i>Object Management System</i>
Interacting directly with the StP storage manager Sybase Adaptive Server/SQL Server	<i>StP Guide to Sybase Repositories</i>
Specific StP products information	The documentation provided with your StP product
A summary of changes in the current software release and last minute information that was not included in the manuals	Release Notes
Windows NT commands	Windows NT documentation

1 Introduction

This manual describes the StP administration tasks required to support Software through Pictures (StP). It explains how to manage StP users, the StP repository manager, and StP systems. This manual also describes how to configure StP to run with DOORS, which allows you to manage system requirements.

Topics covered in this chapter are as follows:

- “StP Overview” on page 1-1
- “Who Is the StP Administrator?” on page 1-5
- “Prerequisites for StP Administration” on page 1-7
- “Starting StP” on page 1-8
- “Using StP Administration Commands” on page 1-9

StP Overview

StP products provide editors for creating and storing system models. These editors include graphical diagram editors, table editors, and an annotation editor. The models that a user creates with these editors are saved both in ASCII files and in a repository, which is supported by an underlying relational database.

Definitions

The following terms are important for understanding any StP product:

- Project
-

- System
- Repository
- System ASCII files
- Repository manager

Project

A project is a directory in your file system. StP system files are stored in a project. It may be useful to think of a project as a parent directory.

System

A system is the basic unit of organization for StP activity and has two main components:

- A directory of ASCII files, each representing an entire diagram, table, or set of annotations created with a product editor
- A database storage area (repository) containing consolidated information about individual objects (rather than whole diagrams or tables) that users create with the product's editors

When a user saves the contents of an StP editor as a diagram, table, or annotation, the contents are written both to an ASCII file and to that system's repository (see Figure 1 on page 1-4). Diagrams and tables must be syntactically correct for StP to store their contents in the repository. Otherwise, the data is stored only in the system's ASCII files.

A system belongs to and is managed by its system owner, although other users may be granted permission to access it.

Repository

A repository is a storage area within a shared, underlying relational database management system (RDBMS). Each repository is an individual, logical database that consolidates and stores information about the objects created in a particular system. Each unique object is stored only once in the system's repository, no matter how many times it appears in an editor's diagrams or tables.

The repository stores the data as Persistent Data Model (PDM) objects, described in *Object Management System*. A single system repository can store objects created by different editors in that system and keeps track of those that belong to each diagram or table. The repository also provides concurrent multi-user access to system data.

The StP modeling applications create and modify repository data. Depending on which StP product is used, a user can generate various outputs from a repository, including:

- Source code
- Structured Query Language statements (SQL)
- Reports

System ASCII Files

Each ASCII file contains:

- An ASCII representation of a viewable diagram or table
- Information about how a diagram or table is connected to other diagrams and/or tables

The ASCII files for a particular system are contained in a subdirectory of the system's project directory. The ASCII files are used by the editors and also enable you to reconstruct the repository, if necessary.

While StP will not update the repository with syntactically incorrect data, users can choose to save an ASCII representation of the incorrect or incomplete diagram or table. When the diagram or table is loaded into the editor, the ASCII-saved elements appear as well, allowing the user to edit and correct the errors.

Repository Manager

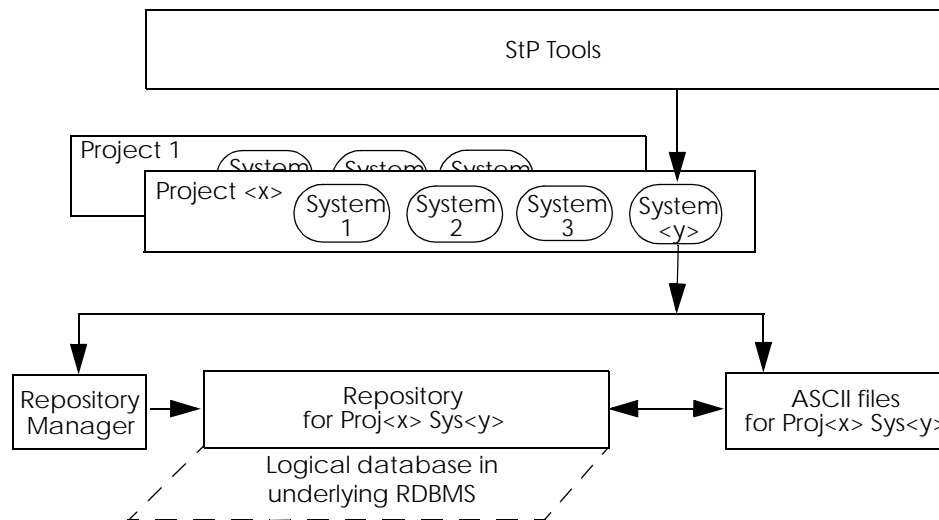
The repository manager is the StP storage manager, which stores and accesses the data for the various system repositories. The repository manager handles storage and access for multiple repositories and keeps track of the objects that belong to each one. Operations involving administration of StP's storage manager, as opposed to the administration of a single system, belong to the repository manager.

The underlying storage manager for StP-generated data can be Sybase Adaptive Server, Sybase SQL Server, or Microsoft Jet. Although the average StP user is not aware of the underlying storage manager, some StP administration tasks involve direct interaction with this storage manager.

For information on using Sybase Adaptive Server or SQL Server, refer to *StP Guide to Sybase Repositories* or to the Sybase documentation, available from their website at <http://www.sybase.com>. To learn more about using Microsoft Jet, refer to the Microsoft Access on-line help.

Figure 1 provides an overview of how StP stores the information for each system.

Figure 1: Overview of StP Structure



Who Is the StP Administrator?

StP administration tasks fall into two major categories that can be performed by various users.

Table 1: StP Administration Tasks

Administrative Tasks	Performed By
Repository manager tasks	StP administrator
System-level tasks	System owner or other authorized user, including the StP administrator

StP Administrator

The StP administrator is responsible for overall administration of the StP repository manager. If you are using Sybase Adaptive Server or SQL Server, the StP administrator is user “sa”, which is also the Sybase system administrator’s user name.

The StP administrator has all access privileges within the repository manager as well as all privileges within any system under the repository manager.

Some of the major repository manager-level StP administration tasks for Sybase may include:

- Adding users to and removing users from the repository manager
- Granting and revoking system creation privileges to repository manager users
- Changing user passwords

For both Sybase and Microsoft Jet, an StP administration task includes performing routine repository manager maintenance

System Administrator's Password

To execute most StP administration commands at the repository manager level, a user must know the system administrator password for user “sa” (Sybase) or “Admin” (Microsoft Jet). Any repository manager user who knows the system administrator's password can act as the StP administrator.

When you first install StP, the StP repository manager password for a Sybase repository is set to “welcome”. StP does not assign a password for Microsoft Jet.

If you perform a function that requires access to Sybase or Microsoft Jet, you may be prompted for the password Sybase or Microsoft Jet is using. Because the repository manager is non-secure at this stage, it is important to assign passwords at both the administrator and user level as soon as possible.

To set up a secure or partially secure repository manager, you set the system administrator's password so that only users who know that password can act as the StP administrator. See “Passwords” on page 2-27 for information about passwords and the differences between secure, partially secure, and non-secure repository managers.

System Owner

A system owner is responsible for administration of a particular system, which may be one of many managed by the repository manager. By default, the system's creator is the system owner.

While the administration tasks relating to the repository manager are the responsibility of the StP administrator, administration tasks that affect only a single system are usually the responsibility of the owner of the system. See “System Owners” on page 2-15 for more information about system ownership.

Because the StP administrator has all privileges within any system under the repository manager, the StP administrator can function as the system owner for any system under the repository manager.

Prerequisites for StP Administration

There are three prerequisites for performing the tasks described in this manual.

StP Is Installed

StP must be installed on your hardware. If necessary, refer to *Installing StP for NT* for instructions on installing StP.

You Can Use the StP Desktop

You should know how to use the StP Desktop. The StP Desktop is the top-level interface between a user and StP. For information on using the StP Desktop, refer to *Fundamentals of StP*. You also can automate many tasks by using QRL scripts. See *Query and Reporting System* and *Object Management System* for more information on QRL (Query Reporting Language) and OMS (Object Management System) functions.

You Have the Appropriate Privileges

To execute repository manager administration commands, you must have system administrator privileges in the repository manager. If StP has been installed with a non-secure repository manager and the system administrator password has not been changed, you, as well as every other user, already have these privileges. Otherwise, it is necessary to request the system administrator password from the StP administrator.

To execute administration commands for an individual StP system, you must be the owner of that system or have been granted the appropriate permissions by the system owner.

Starting StP

After starting the Desktop, you must set the current project and system before you can execute StP administration commands.

Starting the Desktop

To start the Desktop:

1. Click the **Start** menu button.
2. Choose the **Programs** menu.
3. Choose **Aonix Software through Pictures > StP <product>**.

Setting the Current Project and System

To execute most StP administration commands from the StP Desktop, the current project and system must be set to a system in which you have access privileges.

If the current project and system have already been set by default to an appropriate system, the Desktop appears after starting StP, with the system open.

If no project or system default has been set, the **Open System** dialog box displays the contents of the StP installation directory. To change the current project and system, select **Open System** from the **File** menu on the desktop, and select the system you want.

To specify a default project and system to be set automatically when StP is started, you can set the *projdir* and *system* ToolInfo variables in a shared ToolInfo file, or in individual ToolInfo files for each user (see “How ToolInfo Files Work” on page 2-8 and “Creating User-Specific Toolinfo Files” on page 2-14 for details). Additionally, there are corresponding environment variables (IDE_PROJDIR and IDE_SYSTEM, respectively) that set the current project and system (see “Modifying the Environment for Individual Users” on page 2-14). However, it is preferable to use ToolInfo variables, because it is easier to manage StP settings from a single location.

Using StP Administration Commands

You can access StP administration commands from the **Tools**, **File**, and **Repository** menus from the StP Desktop.

Additionally, you can write QRL scripts to run the StP administration commands. For more information on the functions to use, see the administrative functions described in *Query and Reporting System*. See also Chapter 4, “Using StP Utility,” and Appendix B, “Command Reference,” for additional commands to use in scripts.

Table 2 describes the **Tools** menu commands relevant for StP Administration.

Table 2: Tool Menu Commands for StP Administration

Command	Description	For Details, See
Locks > Manage Locks	Manages lock in current system.	<i>Fundamentals of StP</i>
Locks > Set Lock Administrators	Sets lock administrators.	
Locks > List Locks	Displays information on all locks in the current system.	
StP Utility	Launches the StP utility.	Chapter 4, “Using StP Utility”
Synchronize DOORS with StP	Synchronizes StP with DOORS requirements.	Chapter 5, “StP/DOORS Integration”
Show ToolInfo Variable	Shows the value of a particular toolinfo variable.	“Showing Toolinfo Variables” on page 2-26

Commands to Administer the Repository

The commands that set up and remove repository manager users are accessible from the Desktop's **Repository** menu and are executed by the StP Administrator.

Table 3 lists the commands.

Table 3: Repository Menu Commands for the Repository Manager

Command	Description	For Details, See
Manage Users > Grant User(s) Sybase System Creation Privileges	Gives Sybase users the ability to create systems.	"Granting Sybase Users System Creation Privileges" on page 2-5
Manage Users > Revoke User(s) Sybase System Creation Privileges	Revokes a Sybase user's system creation privileges.	"Revoking Sybase System Creation Privileges" on page 2-6
Manage Users > Repository Manager > List All Users	Lists users with access to the StP repository manager.	"Listing All Repository Manager Users" on page 2-7
Manage Users > Repository Manager > List Current Sybase Users	Lists current Sybase users in the repository manager.	"Listing Current Sybase Repository Manager Users" on page 2-7
Manage Users > Repository Manager > Add Sybase User(s)	Gives users access to the StP Sybase repository manager.	"Adding Sybase Repository Manager Users" on page 2-4
Manage Users > Repository Manager > Delete Sybase User(s)	Revokes users' access to the Sybase repository manager.	"Deleting Sybase Repository Manager Users" on page 2-5

Table 3: Repository Menu Commands for the Repository Manager (Continued)

Command	Description	For Details, See
Manage Users > Repository Manager > Change User(s) Sybase Password	Changes a Sybase user's StP password.	"Changing User Passwords" on page 2-33
List Repositories	Lists repositories.	"Listing Repositories" on page 3-15
Show Sybase Server Space	Shows device space available in the current Sybase repository manager.	"Showing Space Available on Devices" on page 3-15
Perform Manager Maintenance	Performs tasks to make the StP repository more efficient.	"Routine Maintenance" on page 3-14

Commands to Administer Individual Systems

The commands for managing individual systems are accessible from the Desktop's **File** and **Repository** menus.

Table 4 lists the **File** menu commands and who can execute them.

Table 4: File Menu Commands to Administer Individual Systems

Command	Description	Who Can Execute	For Details, See
New > System	Creates a database and a hierarchy of directories and files for a system.	User (who automatically becomes the System Owner with appropriate privileges)	<i>Fundamentals of StP</i> and "Creating a System" on page 2-16

Table 4: File Menu Commands to Administer Individual Systems (Continued)

Command	Description	Who Can Execute	For Details, See
Open System	Opens a system.	StP Administrator, System Owner, System User	<i>Fundamentals of StP</i>
Copy System	Makes a copy of a system.	StP Administrator, System Owner, System User	<i>Fundamentals of StP</i> and “Copying a System” on page 2-27
Destroy System	Destroys a system repository and system files.	StP Administrator, System Owner	“Destroying a System” on page 3-23

Table 5 lists the **Repository** menu commands and who can execute them.

Table 5: Repository Menu Commands to Administer Individual Systems

Command	Description	Who Can Execute	For Details, See
Maintain Systems > Show System Space	Shows the amount of space used and space available for the current system.	StP Administrator, System Owner, System User	“Showing Space for the Current System” on page 2-25
Maintain Systems > Expand Current System Repository	Increases the size of the StP repository storage space.	StP Administrator, System Owner	“Expanding a Sybase Repository” on page 3-20
Maintain Systems> Check if System Repository Exists	Determines if the current system has an existing repository.	StP Administrator, System Owner, System User	“Checking for a Repository” on page 2-24
Maintain Systems > Destroy System Repository	Destroys a repository without destroying the system files.	StP Administrator, System Owner	“Destroying a System Repository” on page 3-23

Table 5: Repository Menu Commands to Administer Individual Systems (Continued)

Command	Description	Who Can Execute	For Details, See
Maintain Systems > Recover System Repository	Restores a repository by rebuilding it from information in system files.	StP Administrator, System Owner	"Rebuilding the Repository from System Files" on page 3-27
Maintain Systems > Add New Files to Current System Repository	Add files from one system to another.	StP Administrator, System Owner, System User	"Adding Files to the Repository" on page 3-21
Maintain Systems > Delete Unreferenced Objects in Current System Repository	Deletes from the repository any objects that no longer appear in the current system.	StP Administrator, System Owner, System User	"Deleting Unreferenced Objects" on page 3-17
Maintain Systems > Dump Current System Repository to Files	Creates a backup of the repository by saving it to dump files.	StP Administrator, System Owner, System User	"Backing Up a Sybase Repository to Dump Files" on page 3-25
Maintain Systems > Load Current System Repository from Files	Restores a repository from a backup consisting of dump files.	StP Administrator, System Owner	"Restoring a Repository from Dump Files" on page 3-26
Maintain Systems > Delete Repository Data in Current System	Removes data about the current system from the repository, but leaves the repository intact.	StP Administrator, System Owner	"Deleting Repository Data" on page 3-22
Maintain Systems > Truncate History for Current System	Cleans up out-of-date history objects from the repository.	StP Administrator, System Owner	"Truncating History" on page 3-18
Manage Users > Current System > List All Users	Displays a list of all valid users of the current system.	StP Administrator, System Owner, System User	"Listing All Users for the Current System" on page 2-24

Table 5: Repository Menu Commands to Administer Individual Systems (Continued)

Command	Description	Who Can Execute	For Details, See
Manage Users > Current System > List Current Users	Displays a list of Sybase users currently using the system.	StP Administrator, System Owner, System User	"Listing Current Users for the Current System" on page 2-24
Manage Users > Current System > Add Sybase User(s)	Gives Sybase users read and/or write access to the current system.	StP Administrator, System Owner	"Adding Users to a System" on page 2-22
Manage Users > Current System > Delete Sybase User(s)	Revokes a Sybase user's access to the current system.	StP Administrator, System Owner	"Deleting Users from a System" on page 2-22
Manage Users > Current System > Change Sybase User(s) Group	Changes a Sybase user's read or write permissions.	StP Administrator, System Owner	"Changing a User's Privileges" on page 2-23
Manage Users > Current System > Change Sybase Owner of Current System	Changes the ownership of the current Sybase system to another Sybase user.	StP Administrator	"Changing System Ownership" on page 2-19

2 StP Users

This chapter describes the roles, identities, and privileges of different types of StP users.

Topics covered are as follows:

- “What Is an StP User?” on page 2-1
- “Repository Manager Users” on page 2-4
- “User Environments” on page 2-8
- “System Owners” on page 2-15
- “System Users” on page 2-21
- “Passwords” on page 2-27

What Is an StP User?

There are different types of StP users with different access privileges. A user who knows the system administrator’s password can perform all StP operations at the repository manager level or in any system that has been created under the repository manager. At the other extreme, there may be new users who can start the StP Desktop but cannot execute any commands or access any StP data. In general, most users probably have access to one or more systems in which they can read, and possibly write to, the system repository using one of StP’s modeling tools.

StP User Access Security

There are three levels of security involved in user access to StP data and StP facilities:

- **Operating system**
The person must be a valid user at the operating system level.
- **Repository manager**
The person must be a valid repository manager user to start the StP Desktop and to execute any StP commands. Only users who have been validated at the operating system level can become repository manager users. See “Repository Manager Users” on page 2-4.
- **System**
To read from or write to a repository, the person must be a valid system user in one or more systems. Only valid repository manager users can become system users. A repository manager user can become a system user either by becoming a system owner or by being granted access privileges to an existing system by the system’s owner or by the StP administrator. See “System Users” on page 2-21.

StP User Names

An StP user’s repository manager user name is usually the same as his or her operating system user login.

Sybase User Name Restrictions

The Sybase user name cannot contain any non-ASCII characters and must be a valid Sybase user account name, conforming to Sybase’s identifier rules, as follows:

- Name begins with an alphabetic character.
- After the first character, can include alphanumeric characters, or the symbols @, #, \$, _, or the symbols for yen or pound sterling.
- No embedded spaces are allowed.
- Maximum length is 30 bytes.
- Cannot be the same as a Sybase reserved word.
- Names are case sensitive.

StP user names are case sensitive; Windows NT user account names are not. This could cause an StP access problem for a user whose login does not match the case of the user's StP user name.

For more information on setting up Sybase user accounts, see *StP Guide to Sybase Repositories* or the *Sybase System Administration Guide*, which is available from the Sybase web site at <http://www.sybase.com>.

Microsoft Jet User Name Restrictions

User names can be 1 to 20 characters long and can include alphanumeric characters, accented characters, numbers, spaces, and symbols, with the following exceptions:

- The characters `" / \ [] : | < > + = ; , ? *`
- Leading spaces
- Control characters (ASCII 00 through ASCII 31)

For more information on setting up Microsoft Jet user accounts, see the Microsoft Access on-line help.

StP User Names that Differ from the User's Login

You can assign an StP repository manager user name that differs from the user's login. This could include any user:

- Whose operating system account name contains non-ASCII characters (for instance, diacritical marks such as accents and umlauts)
- Who is accustomed to logging into Windows NT by typing their user account name in a varying mix of upper or lower case letters

To assign an StP user name that differs from the user's login, set the *oms_dbuser* ToolInfo variable to the new user name in the user's environment and add this user name to the StP repository manager. StP then uses the *oms_dbuser* ToolInfo variable setting as the valid StP user name.

For information on ToolInfo variables, see "User Environments" on page 2-8.

Repository Manager Users

You must be a repository manager user to execute any StP commands. However, the only privilege of being repository manager user is permission to start StP and connect to the repository manager. Since most meaningful work occurs within the context of a system, you must become a system owner or a system user to use the modeling tools to create, access, and modify repository data. See “System Owners” on page 2-15 for information about becoming a system owner and “System Users” on page 2-21 for information about becoming a system user.

The StP administrator maintains repository manager users. This section describes procedures for doing so. Also, system users can change their own passwords.

Adding Sybase Repository Manager Users

When you add a new Sybase user to the repository manager from the Desktop, the user is given access to the entire repository.

To add repository manager users:

1. From the **Repository** menu, choose **Manage Users > Repository Manager > Add Sybase User(s)**.
2. In the **Add Sybase Server User(s)** dialog box, type the StP user names in the **User Name(s)** field.
 - When adding multiple users, separate their names by a space.
 - With certain exceptions, the StP user name is usually the same as the user’s operating system login (for details, see “StP User Names” on page 2-2).
3. Click **OK**.

Alternatively, you can use the **repos_add_user** function in a QRL script to add users and specify their passwords. For details, see *Query and Reporting System*.

See “Passwords” on page 2-27 for information about user passwords, including how to change a password. For Sybase, the default password for new users (including administrators) is “welcome”. The StP administrator or the user should assign a password as soon as possible.

Deleting Sybase Repository Manager Users

The **Delete Sybase User(s)** Desktop command deletes a user from the current Sybase repository manager. This command deletes:

- The user's name from any system under the current repository manager to which the user formerly had access permission
- The user from the current repository manager

A user who has been deleted from the repository manager can no longer open StP systems.

You cannot delete a repository manager user who owns a system under that repository manager. Before deleting this type of user, you must either change system ownership for any systems owned by the user or remove those systems from the repository manager.

To delete repository manager users:

1. From the **Repository** menu, choose **Manage Users > Repository Manager > Delete Sybase User(s)**.
2. In the **Delete Sybase Server User(s)** dialog box, type the user names in the **User Name(s)** field.

When deleting multiple users, separate their names by a space.

3. Click **OK**.
4. If prompted, enter the system administrator's password.

Alternatively, you can use the **repos_delete_user** function in a QRL script to delete users. For details, see *Query and Reporting System*.

Granting Sybase Users System Creation Privileges

The StP administrator can grant permission to create a system to any Sybase user who has been validated at the operating system level. If the user has not yet been added as a repository manager user, the **Grant User(s) System Creation Privileges** Desktop command automatically adds the user as a repository manager user. It is not necessary to add the user to the repository manager as a separate operation.

To grant users system creation privileges:

1. From the **Repository** menu, choose **Manage Users > Grant User(s) Sybase System Creation Privileges**.
2. In the **Grant User(s) System Creation Privileges** dialog box, type the user names in the **User Name(s)** field.
When granting privileges to multiple users, separate their names by a space.
3. Click **OK**.

Alternatively, you can use the **repos_add_maker** function in a QRL script to grant users system creation privileges. For details, see *Query and Reporting System*.

Revoking Sybase System Creation Privileges

You can revoke system creation privileges from a repository manager user with the **Revoke User(s) Sybase System Creation Privileges** Desktop command.

This command revokes the specified user's permission to create a new system, but it does not remove the user as a repository manager user. To remove the user from repository manager, use the **Delete Sybase User(s)** Desktop command.

Revoking system creation privileges from a user does not affect any systems that the user has already created. If the user is currently the owner of a system, he or she remains the owner of that system.

To remove users' system creation privileges:

1. From the **Repository** menu, choose **Manage Users > Revoke User(s) Sybase System Creation Privileges**.
2. In the **Revoke User(s) Sybase System Creation Privileges** dialog box, type the user names in the **User Name(s)** field.
When revoking privileges from multiple users, separate their names by a space.
3. Click **OK**.
4. If prompted, enter the system administrator's password.

Alternatively, you can use the **repos_delete_maker** function in a QRL script to revoke users' system creation privileges. For details, see *Query and Reporting System*.

Listing Current Sybase Repository Manager Users

To list current Sybase repository manager users:

1. From the **Repository** menu, choose **Manage Users > Repository Manager > List Current Sybase Users**.
2. Click **OK**.
3. If prompted, enter the system administrator's password.

Alternatively, you can use the **repos_list_current_users** function in a QRL script. For details, see *Query and Reporting System*.

Listing All Repository Manager Users

To list repository manager users:

1. From the **Repository** menu, choose **Manage Users > Repository Manager > List All Users**.
2. In the **List All Repository Manager Users** dialog box, select either **Sybase** or **MS Jet** from the **Server** menu.
3. Click **OK**.
4. If prompted, enter the system administrator's password.

Alternatively, you can use the **repos_list_users** function in a QRL script to list all repository manager users. For details, see *Query and Reporting System*.

Assigning User Passwords

A user's password is originally set when the user is added to the repository manager by the **Add Sybase User(s)** or **Grant User(s) Sybase System Creation Privileges** Desktop commands, or by the **repos_add_user** or **repos_add_maker** functions. The StP administrator can change a user's password, or a user can change his or her own password. See "Passwords" on page 2-27 for more information.

It is not possible to change Microsoft Jet passwords through StP.

User Environments

You can customize each StP user's environment by using either of the following methods:

- **ToolInfo variables**—The ToolInfo variables are stored in the ToolInfo file, which is by default kept in the *StP* install directory. You can change these settings in a commonly accessed ToolInfo file for all users, as well as in copies of the ToolInfo file that you can create and customize for each user. Ideally, you should use the ToolInfo variables to customize the user's environment.
- **Environment variables**—Several StP environment variables affect StP and Sybase Adaptive Server or SQL Server. The StP installation and start-up procedures set default values for these variables. Users can change certain variable settings to customize their own environments.

How ToolInfo Files Work

A ToolInfo file contains StP ToolInfo variable settings that determine certain characteristics of the StP products, such as the default project directory and system name. An StP installation can contain a hierarchy of ToolInfo files, in which the first-read (default) ToolInfo file takes precedence over any subsequently read ToolInfo files. You can specify the first-read ToolInfo file by setting the ToolInfo environment variable. If the ToolInfo environment variable is not set, StP looks for the *ToolInfo.<platform>* file in the StP installation directory. If *ToolInfo.<platform>* does not exist, StP looks for any *ToolInfo* file in the installation directory.

You can daisy-chain a series of ToolInfo files by including a *ToolInfo* variable setting in each one, to point to the next ToolInfo file in the chain.

Variables not set in an earlier-read ToolInfo file may be set in a ToolInfo file that is lower in the hierarchy. In this way, an installation may have default ToolInfo settings for an entire StP installation, for development

groups, and for individual users. For example, a development group's primary project directory might be set in the first-read ToolInfo file. However, an individual user may prefer to open a specific system by setting an additional ToolInfo variable in the user's local environment.

In some cases, a setting in one ToolInfo file will conflict with another ToolInfo file variable setting. When that happens, StP chooses the setting in the first-read ToolInfo file, starting from the location of the ToolInfo environment variable setting. If a user has preferred ToolInfo variable settings that conflict with those in the main ToolInfo file, the user can set the ToolInfo environment variable to point to his or her own ToolInfo file, and have the main ToolInfo file lower in the ToolInfo file daisy chain.

To ensure that the default settings in the installation-wide ToolInfo file are set appropriately, the StP administrator should check and, if necessary, modify these settings in the default ToolInfo file provided with StP.

For information about setting ToolInfo variables for individual users, see "Creating User-Specific Toolinfo Files" on page 2-14.

For a list of ToolInfo variables associated with StP administration, see Table 1 on page 2-10.

For a list of all ToolInfo variables, see Appendix A, "ToolInfo Variables."

ToolInfo Variables Versus Environment Variables

Except for the DSQUERY environment variable, which is supplied by Sybase, StP environment variables have ToolInfo variable equivalents. It is better to use the ToolInfo variables rather than the environment variables. For compatibility with previous versions, StP supports the environment variable equivalent, but may not support these environment variables in the future. You can still set the environment variables, but it is easier to manage these settings from a single location, the ToolInfo file, than from the disparate locations where environment variables are often set.

Note: The StP environment variables take precedence over the settings in the ToolInfo file. Be careful to make sure that the environment variables and ToolInfo variable settings are consistent.

Table 1 describes ToolInfo variables for administering StP. Corresponding environment variables, if applicable, are also listed.

For a list of all StP ToolInfo variables, see Appendix A, “ToolInfo Variables.”

Table 1: StP ToolInfo and Environment Variables

ToolInfo Variable	Environment Variable	Description
msgd_host	IDE_MSGD_HOST	Specifies a host machine for the StP Message Router.
msgd_port	IDE_MSGD_PORT	Specifies the port number used by the Message Router.
msjet_password	None	For Microsoft Jet, specifies the password for any user who is not a system administrator.
msjet_admin_password	None	Specifies the password for the Microsoft Jet system administrator.
oms_dbuser	None	Assigns an StP user name that differs from the user's login.
product	IDE_PRODUCT	Specifies the default StP product. (The previous name for the <i>product</i> ToolInfo variable was <i>ide_product</i> .)
projdir	IDE_PROJDIR	Specifies location of the StP project directory.
<server>_admin_password	None	Specifies the password for the Sybase system administrator.
<server>_password	<server>_PASSWD	For Sybase, specifies the password for the default Adaptive Server or SQL Server for any user who is not a system administrator.
syscreate_device	DBDEVICE	Specifies the Sybase device to store the Adaptive Server or SQL Server repository data.
syscreate_log_device	LOGDEVICE	For Sybase, specifies the device on which to store the repository's transaction log.

Table 1: StP ToolInfo and Environment Variables (Continued)

ToolInfo Variable	Environment Variable	Description
syscreate_log_size	LOGSIZE	For Sybase, sets the size of the repository data, in megabytes. If not set, the default is 6MB.
syscreate_size	SIZE	For Sybase, sets the size of the repository, in megabytes.
system	IDE_SYSTEM	Specifies the default StP system.
ToolInfo	ToolInfo	ToolInfo variable—Points to another ToolInfo file in the user's environment. Environment variable—Specifies default ToolInfo file to use.
None	DSQUERY	Specifies the default Sybase server. This environment variable is supplied with the Sybase software.
None	USER	Specifies a user name.

Understanding ToolInfo Variable Syntax

The syntax of a ToolInfo assignment is:

```
<ToolInfo_variable>=<value>
```

where <ToolInfo_variable> is the variable name and <value> is the setting.

A ToolInfo variable can have a value that is:

- A number or a character string
- The pathname for a file or directory
- A command with optional parameters

If <value> is blank (for example, `projdir=`), this in effect “clears” the variable.

Number or String Variables

The simplest variables are those whose values are integers, decimal numbers, or character strings. These variables often provide a simple on/off switch for some action that StP can take, or they give the dimensions of a display feature, such as window height or page length. Typical values are 0 (or False) and 1 (or True), or a mnemonic string, such as internal or external.

For example, to set the variable controlling the location of the system to the *elevator* system, use:

```
system=elevator
```

Pathname Variables

Some variables have values that are complete pathnames. The pathname may end in the name of an executable file. In that case, when the variable is accessed, StP executes the file. Other pathnames end in the name of a directory. In those cases, when the variable is accessed, StP has access to any of the files in that directory.

For example, the variable that specifies the path used by StP to find information in UML templates directories may appear in the file as:

```
uml_stp_file_path=C:\StP\templates\uml
```

Command Variables

Some variables have a command as their value. This can be a simple, single-line operating system command, such as **more** <filename>, or the command may name a command script file followed by a list of parameters or flags to pass to the file.

For example, the variable that designates the editor used to edit scripts in the Script Manager has a value that is a command:

```
scriptman_script_editor=notepad${scriptpath}&
```

Creating User ToolInfo Variables

You can create your own ToolInfo variables. For example, if you have created or modified a QRL (Query and Reporting Language) script that uses variables, you can create ToolInfo variables to set values to those variables. Different users can run the script using different values, simply by setting the ToolInfo variables differently in their local ToolInfo files.

Finding the Current ToolInfo Value

You can find the current value of a ToolInfo variable by using the **Show ToolInfo Variable** command from the StP Desktop or by running **toolloc** at a command line.

Using the StP Desktop

To find a value of a ToolInfo variable from the StP Desktop:

1. From the **Tools** menu, choose **Show ToolInfo Variable**.
2. In the **Show ToolInfo Variable** dialog box, enter the name of the ToolInfo variable.
3. If you want see the file path of where the ToolInfo variable is located, check **Show ToolInfo Chain**.
4. Click **OK**.

The variable information is displayed. For examples, see the following section.

Using toolloc from a Command Line

The **toolloc** command provides a means of obtaining the current value for a given ToolInfo variable.

The syntax of **toolloc** is:

```
toolloc [-w] <ti_variable> [<value>]
```

where <ti_variable> is a given ToolInfo variable.

For example, to print the current value of the *projdir* ToolInfo variable to standard output, type:

```
toolloc projdir
```

The output could look like this:

```
C:\development\StP\projects
```

If the value of the variable is undefined, **toolloc** prints <ti_variable>, unless you specify <value>. This is useful when you want to pipe a specific value for the variable to another command.

The **-w** option displays the default value, as well as the chain of ToolInfo files searched, marking the file that gives the current value. For example, to see the chain of ToolInfo files searched to find the value of the *projdir* variable, type:

```
toolloc -w projdir
```

The output displayed would resemble this:

```
Variable: projdir
```

File	Value
=====	
*C:\dlarkin\stp\ToolInfo	C:\development\StP\projects

The asterisk (*) indicates the file with the current setting.

Modifying the Environment for Individual Users

Users can customize their individual environments by:

- Creating and pointing to their own ToolInfo file
- Setting environment variables in their own environment

Creating User-Specific Toolinfo Files

To customize certain aspects of an StP product for an individual user, you can create a user-level ToolInfo file with the desired ToolInfo variable settings in the user's environment—for example, in the user's home directory. Next, point to this customized ToolInfo file instead of the installation-wide one by including its filename and directory

specification in the ToolInfo environment variable setting in the user's environment, as explained in the following section.

To invoke a hierarchical chain of ToolInfo files, include in each ToolInfo file a setting for the *ToolInfo* ToolInfo variable that points to the next ToolInfo file in the hierarchy. The last-referenced ToolInfo file in the chain should be the installation-wide ToolInfo file that was installed with StP.

Setting Environment Variables

Two types of environment variables affect a Windows NT user's environment: system variables and user variables. Any user can change his or her own user variables. Only the Windows NT system administrator can change the system variables.

For information on how to set environment variables, refer to your Windows NT documentation.

System Owners

A repository manager user becomes a system owner by creating a system. A system owner is the owner of that system's repository and has special privileges that allow him or her to:

- Maintain or modify that system's repository
- Perform other administration tasks related to that system, such as granting system access to other users

Before a user can create a system, the user must have been granted system creation privileges by the StP administrator (see "Granting Sybase Users System Creation Privileges" on page 2-5).

File Permissions

Make sure that users have change permissions in the project directory under which the system directory will be created. To modify permissions, use the Windows NT Explorer program.

System Names

For Sybase users, system names must conform to Sybase's identifier rules, as follows:

- Adaptive Server and SQL Server identifiers can be a maximum of 30 characters in length, whether or not multibyte characters are used.
The first character of an identifier must be any alphabetic character as defined in the current character set in use on Adaptive Server or SQL Server or the _ (underscore) symbol.
- You cannot begin an identifier with a number.
- You cannot include any non-ASCII characters in the system name.
- After the first byte, you can include characters declared as alphabetic, numeric, and these symbols: #, @, and money symbols such as \$ (dollars) or ¥ (yen).
- You cannot embed spaces in identifiers.
- You cannot use the names of Sybase reserved words.

For more information on Sybase naming conventions, identifiers, and reserved words, consult the Sybase documentation, which is available on their website at <http://www.sybase.com>.

For Microsoft Jet users, system names must follow the regular file conventions used by their file systems. For more information on names in Microsoft Jet, see the Microsoft Access on-line help.

Creating a System

The **New System** Desktop command creates a new system as a subdirectory of the specified project directory.

To create a system:

1. From the **File** menu, choose **New > System**.
2. In the **Create System** dialog box, do the following:
 - In the **Create In** field, type the path to the desired project. The default is offered, specified by the *projdir* ToolInfo variable.
 - In the **System Name** field, specify the new system's name.
 - In the **Repository Server** menu, select either Sybase or MS Jet.

Though there are no set guidelines for choosing a database type, if there will be more than five users, or you anticipate having a large repository, it is recommended that you select Sybase. (Sybase must be installed and running.)

- If you are using Sybase, enter a value in the **Repository Size (MB)** field. The default is 6 MB.

3. Click **OK**.

Alternatively, you can use the **sys_create** function in a QRL script to create a system. For details, see *Query and Reporting System*.

System Defaults for Sybase Repositories

When StP creates a new system in a Sybase repository, it uses the ToolInfo settings in the user's environment to determine the following values:

- **syscreate_size** is the size of the system repository's data area for a Sybase server. If not set, the default is 6 MB.
- **syscreate_log_size** is the size of the system repository's transaction log. The transaction log is a record of modifications to the repository data. It enables automatic recovery of the repository in the event of a system failure on the server. If **syscreate_log_size** is not set, the default is 2 MB.
- **syscreate_device** is the Sybase device on which StP will store the repository data. If not set, it defaults to the device designated as a default device in your Sybase Adaptive Server or SQL Server installation. Generally, it is best to leave this variable unset, unless you have specific reasons for creating a system whose repository resides on its own device (see the discussion that follows for more information).
- **syscreate_log_device** is the Sybase device on which to store the repository's transaction log. If not set, it defaults to the device designated as a default device in your Sybase Adaptive Server or SQL Server installation. It's generally a good idea to store the transaction log on a different device from the repository data to enable data recovery if the repository's device fails.

You can override any of the default settings for these variables by setting the user environment variables before starting the StP Desktop or before creating a system with the **sys_create** function, as described in "User

Environments” on page 2-8. Make sure the specified device has been configured for Adaptive Server or SQL Server before you use it with StP (see *StP Guide to Sybase Repositories* for details).

When deciding whether to set the *syscreate_device* ToolInfo variable, keep in mind the following restrictions regarding repository expansion:

- If your system grows too large for the originally allocated space, StP allows you to expand the size of the repository.
- If you create your repository on an Adaptive Server or SQL Server default device (*syscreate_device* is unset) and this device later becomes full, StP can expand the repository to additionally designated default devices. However, if you create your repository on a device specified by *syscreate_device*, StP performs all repository tasks on that device, including any future expansion.
- If the device becomes full, no expansion to other devices can occur.

For more information, see “Expanding a Sybase Repository” on page 3-20.

To determine how much space is available on the devices in the repository manager, use the **Show Sybase Server Space** command. See “Showing Space Available on Devices” on page 3-15.

System Defaults for Microsoft Jet Repositories

Except for the *msjet_password* and *msjet_sysadmin_password* ToolInfo variables, there are no special ToolInfo or environment variables that set defaults for Microsoft Jet repositories.

The size of the Microsoft Jet repository depends on the size of your StP systems. As the StP repository grows, Microsoft Jet automatically allocates it sufficient space. If the StP repository grows too large, you may experience:

- Insufficient disk space
- Performance degradation or data corruption problems

For more information, see “Routine Maintenance” on page 3-14 and “Expanding a Microsoft Jet Repository” on page 3-18.

Locking

When a new system is created on either a Sybase or Microsoft Jet repository, locking is automatically enabled. The system owner is the lock administrator. To change the lock administrator or to add additional lock administrators, use the **Tools > Locks > Set Lock Administrators** command.

For information about locking, see *Fundamentals of StP*.

Server/Repository Identification

When you create a system, a file named *.repinfo* is created in the system directory. This file contains the type of repository server, the name of the server (if the repository is on Adaptive Server or SQL Server), and the name of the StP repository for the system. You can view the contents of *.repinfo*, as you would any other ASCII text file. Do not remove or change the contents of this file.

Changing System Ownership

The user who creates a system is the original system owner. Changing ownership of a system from one user to another requires the system administrator's password.

A system can have only one owner, but the StP administrator has all the privileges of system ownership on all systems.

To change the ownership of a Sybase system from the StP Desktop:

1. Set the current project/system to the system for which you want to change ownership.
See "Setting the Current Project and System" on page 1-8 for instructions, if necessary.
2. From the **Repository** menu, choose **Manage Users > Current System > Change Sybase Owner of Current System**.
3. In the **Change Owner of Current System** dialog box, type the new user name in the **New Owner** field.
4. Click **OK**.

5. If prompted, enter the system administrator's password.

Alternatively, you can use the **sys_change_repowner** function in a QRL script to change system ownership. For details, see *Query and Reporting System*.

Commands Performed by System Owners

The owner of a system is responsible for overall administration of the individual system. Major system-level administration tasks involve managing system users and system data.

The primary tasks involving management of Sybase system users are:

- Listing System Users
- Adding System Users
- Deleting System Users
- Changing A System User's Access Privileges

The commands for performing these tasks are described in "System Users" on page 2-21.

The primary tasks involving management of system data are:

- Expanding a system repository (Sybase only)
- Deleting unreferenced objects in a system repository
- Adding files to a system repository
- Showing space for a system repository (Sybase only)
- Regenerating a system repository
- Destroying a system repository
- Destroying a system
- Dumping a system repository (Sybase only)
- Loading a system repository (Sybase only)
- Managing the system and project directories with StP Utility

Chapter 3, "Administering StP," describes the facilities for managing system data.

System Users

A person becomes a Sybase system user when the system owner adds the user to the system with the **Add Sybase User(s)** Desktop command or with the **sys_add_user** QRL function equivalent. The person being added as a system user must already be a valid repository manager user. See “Adding Sybase Repository Manager Users” on page 2-4 for information about becoming a repository manager user.

A repository manager user can be a system user in several systems.

The system owner or StP administrator maintains system users. This section describes procedures for doing so.

Note: Management of Microsoft Jet users is not provided with StP. In a default installation, there is no security for the Microsoft Jet systems. Refer to your Microsoft Jet or Microsoft Access documentation for more information on Microsoft Jet security.

Privileges

The privileges of a Sybase system user depend on the group in which the user is a member. There are two groups of system users: readers and writers. A user with read group privileges can only read repository objects, such as diagrams, tables, and annotations. A user with write group privileges can create and edit repository objects as well as read them.

System users can also execute certain system-level StP administration commands in the systems to which they have access. See “Commands Performed by System Users” on page 2-23 for a list of the commands that can be executed by readers and writers.

A user’s group is set when the user is added to the system. If no group is specified, the default is to add the user as a system writer. The system owner can later change a user’s group with the **Change Sybase User(s) Group** command.

The StP administrator and the system owner have all the privileges of system users within the system, as well as the additional privileges attached to their specific roles.

Adding Users to a System

You can add an existing repository manager user as a new user of the current system with the **Add Sybase User(s)** Desktop command.

To add users to the current system:

1. From the **Repository** menu, choose **Manage Users > Current System > Add Sybase User(s)**.
2. In the **Add User(s) to Current System** dialog box, type the user names in the **Writer(s)** and **Reader(s)** fields. When adding multiple users, separate their names by a space.

Writers are users who have write (change) privileges for the current systems. Readers can access the system, but cannot make any changes to it.

3. Click **OK**.

Alternatively, you can use the **sys_add_user** function in a QRL script to add users to a system. You can add some users as readers and others as writers in a single invocation of **sys_add_user**. For details, see *Query and Reporting System*.

Deleting Users from a System

You can revoke a system user's access to a system with the **Delete Sybase User(s)** Desktop command. After this command has successfully executed, the specified user no longer has access to the system for which the command was executed. However, the user remains a valid repository manager user, and may retain access to other systems under the repository manager.

To revoke user access to the current system:

1. From the **Repository** menu, choose **Manage Users > Current System > Delete Sybase User(s)**.
2. In the **Delete User(s) from Current System** dialog box, type the user names in the **User Name(s)** field. When deleting multiple users, separate their names by a space.
3. Click **OK**.

Alternatively, you can use the **sys_delete_user** function in a QRL script to revoke users' access to a system. For details, see *Query and Reporting System*.

Changing a User's Privileges

You can set the access privileges for one or more users with the **Change Sybase User(s) Group** command. If a user is currently a member of a different group from the group that you specify in the command (for example, the user is a reader and the command specifies writers) the command changes that user's privileges to those of the specified group. If a user is currently a member of the group that you specify in the command, the user's access privileges are not changed.

To change user access privileges in the current system:

1. From the **Repository** menu, choose **Manage Users > Current System > Change Sybase User(s) Group**.
2. In the **Change to Writers Group** and **Change to Readers Group** fields, enter the user names.
When changing the group for multiple users, separate their names by a space.
3. Click **OK**.

Alternatively, you can use the **sys_change_user** function in a QRL script to set users' system access privileges. You can make some users readers and other users writers in a single invocation of **sys_change_user**. For details, see *Query and Reporting System*.

Commands Performed by System Users

All system users can perform the following tasks within a system:

- List all system users
- List current system users (Sybase only)
- Check for existence of a repository
- Show space for the current system (Sybase only)
- Show ToolInfo variable values
- Copy a system

- Dump a system repository to a file on disk (Sybase only)

In addition, all system writers can delete unreferenced objects from the system.

Listing All Users for the Current System

To list all users in the current system:

1. From the **Repository** menu, choose **Manage Users > Current System > List All Users**.
2. Click **OK**.

Alternatively, you can use the **sys_list_users** function in a QRL script to list all system users. For details, see *Query and Reporting System*.

Listing Current Users for the Current System

To list all current Sybase system users:

1. From the **Repository** menu, choose **Manage Users > Current System > List Current Users**.
2. Click **OK**.

Alternatively, you can use the **sys_list_current_users** function in a QRL script to list all system users. For details, see *Query and Reporting System*.

Checking for a Repository

You can determine whether a system has a repository with the **Check if System Repository Exists** Desktop command.

To check for a repository:

1. From the **Repository** menu, choose **Maintain Systems > Check if System Repository Exists**.
2. In the **Check if System Repository Exists** dialog box, enter the project directory (the default is offered) and the system you want to check.
3. Click **OK**.

The status appears in the lower left corner of the Desktop; it displays whether the system has a repository and the name of the repository if it exists. The repository name is usually the same as the system name.

If the system has no repository, the system owner or the StP administrator can restore it by rebuilding a repository from the system files, using the **Recover System Repository** command. See “Rebuilding the Repository from System Files” on page 3-27 for information about this command.

Alternatively, you can use the **sys_has_rep** function in a QRL script to check for a system repository. For details, see *Query and Reporting System*.

Showing Space for the Current System

If you are using a Sybase repository, you can show the amount of space used and space available for the current system with the **Show System Space** Desktop command. Any system user can display the space usage data for the current system. If the repository is Sybase Adaptive Server or SQL Server, and if the system has grown too large for the allocated space, the system owner can allocate more space with the **Expand Current System Repository** command (see “Expanding a Sybase Repository” on page 3-20).

The **Show System Space** command does not apply to Microsoft Jet repositories. A Microsoft Jet repository is automatically allocated extra space as it grows.

To show space for the current system:

1. From the **Repository** menu, choose **Maintain Systems > Show System Space**.

The **Show System Space** window displays space statistics, for example:

Figure 2: Show System Space

Database Name: purchase_sys			
Database Segment	Size	Available	Capacity
data only	6.00Mb	5.20Mb	13.28%
log only	2.00Mb	2.00Mb	0.00%
Total:	8.00Mb	7.20Mb	9.96%

2. Click **Close**.

The results appear as individual totals of all space allocated (Size column), unused (Available column), and used (Capacity column) on all devices combined, for each of the database segment types that exist in this system. Possible segment types that may appear on separate lines in the display are:

- data only—Segments limited to repository data
- log only—Segments limited to transaction log
- data and log—Segments that provide space for both repository data and transaction log (appears only if repository has been expanded)

Alternatively, you can use the **sys_show_space** function in a QRL script to check for a system repository. For details, see *Query and Reporting System*.

Showing Toolinfo Variables

From the Desktop, you can display the current value of a ToolInfo variable. You can also display the file path and assigned value of a ToolInfo variable for the entire chain of ToolInfo files searched.

To display the value of a ToolInfo variable:

1. From the **Tools** menu, choose **Show ToolInfo Variable**.
2. In the **Show ToolInfo Variable** dialog box, type the variable name.
3. To display the value for the specified variable in all ToolInfo files searched, select the **Show ToolInfo Chain** option.

This option lists the file path and value for each instance of the specified ToolInfo variable in the entire search chain.

4. Click **OK**.

The **Show ToolInfo Variable** output window displays the ToolInfo variable's current value. If the output includes the entire ToolInfo chain, an asterisk (*) indicates the current setting.

5. Click **Close**.

Another way to show the value of a ToolInfo variable is to use the **toolloc** command at the command prompt, described under "toolloc" on page B-42.

Copying a System

To make a copy of a system:

1. From the **File** menu, choose **Copy System**.
2. In the **Copy System** dialog box, type the old and new project system directories (the defaults are offered) and project system names.
3. Click **OK**.

Dumping a System Repository

Any Sybase system user can dump a system repository to ASCII files. You might want to do this to create a set of files as a backup or to copy the repository to a different site or different project. However, only the system owner or the StP administrator can load a repository from the dump files.

Use the **Dump Current System Repository to Files** command to dump a repository. Because dumping a repository is most often done as a routine data maintenance task, it is described along with other data management tasks in Chapter 3, “Administering StP.”

Deleting Unreferenced Objects from a System

All system users with write privileges can delete unreferenced objects from the current system. This task is described under “Cleaning Up” on page 3-17.

Passwords

Repository manager security depends on the extent to which StP requires a user or the StP administrator to enter a password. As StP administrator, you can choose the level of security that best fits your needs.

The level of security is controlled by:

- Whether all user passwords are the same and whether that password is defined in the ToolInfo file or through an environment variable

- Whether the StP administrator's password is the default password

When you first install StP, the default Sybase password is "welcome". StP does not assign a default password for Microsoft Jet.

Passwords for Sybase

When you first install Sybase, the default password is "welcome". If you change the Sybase password, however, you will need to set Sybase passwords using the following ToolInfo variables, which initially are not set:

- `<server>_password`—Sets the user password, where `<server>` is the name of the repository's Sybase server. This server is specified by the DSQUERY environment variable. Set this variable in the user's ToolInfo file.
- `<server>_admin_password`—Sets the Sybase administrator's password. Set this variable in the main ToolInfo file.

You can change these settings for a default Sybase server, as well as for each Sybase server to which a user may want to connect.

The `<server>_password` and `<server>_admin_password` ToolInfo variables affect security levels within the repository manager. Depending on their current setting, these ToolInfo variables determine whether StP prompts the user for a password, as follows:

- If `<server>_password` is not set and the password is not "welcome", StP prompts the user for a password for the corresponding server before starting StP or executing StP administration commands.
- If `<server>_password` is set to a password common to all users other than sa, users can start StP and execute StP system-level administration commands for their own systems without entering a password for that server.

You should set passwords as soon as possible. If the password is not set or is commonly known, as "welcome" is, the repository manager is non-secure. Any user who knows this information can start StP and execute all StP administration commands. For this reason, it is important to assign passwords for both users and administrators as soon as possible.

Note: Environment variables that are to be set the same for all users are defined in the ToolInfo file, which StP accesses automatically on start-up. See “User Environments” on page 2-8 for more information.

The StP administrator can change repository manager security levels by changing the system administrator and user passwords and changing the `<server>_password` and `<server>_admin_password` ToolInfo variables. A user can also change his or her own user password and can change the `<server>_password` ToolInfo variable in his or her own environment. For detailed instructions on changing passwords, see “Changing the System Administrator’s Password” on page 2-32 and “Changing User Passwords” on page 2-33.

To unset `<server>_password` by default for all users or `<server>_admin_password` for StP administrators, remove or comment out the lines in which it appears in the ToolInfo file in the StP installation directory. Keep in mind that users can set `<server>_password` variables in their own environments independently of the ToolInfo file in the StP installation directory.

To unset `<server>_password` or `<server>_admin_password` in the user’s environment, make sure that the `<server>_PASSWD` environment variable is not set in each individual user’s environment (see “Modifying the Environment for Individual Users” on page 2-14). For more information on setting the ToolInfo variables, see “User Environments” on page 2-8.

Repository Manager Security Levels

An StP repository manager can be non-secure, partially secure, or secure.

Non-Secure

Immediately after StP is installed, the repository manager is non-secure. A non-secure repository manager is completely accessible to all users who have been added to the repository manager.

All users can start StP and execute all StP commands, including all StP administration commands, without a password. The users must be valid repository manager users and valid system users in the system in which they are working.

Partially Secure

A partially secure repository manager requires a user to provide the system administrator's password for certain operations.

Users do not have to enter a password to start StP, but they do need to enter the StP administrator's password to execute repository manager administration commands.

Secure

In a secure repository manager, all users are prompted for a password to start StP and to execute StP administration commands.

Summary of Security Level Determinants

The following table explains the various determinants of repository manager security levels and the resulting user access capabilities.

Table 2: Repository Manager Security

StP Required Passwords	Non-Secure	Partially Secure	Secure
User passwords	Same password for all users (initially set to the Sybase or Microsoft Jet default password)	Same password for all users (initially set to the Sybase or Microsoft Jet default password)	Unique password for every user or user group
Sybase sa or Microsoft Jet Admin password	Same password for all users, including StP administrator (initially set to the Sybase or Microsoft Jet default password)	StP administrator's password differs from user passwords and is not publicly accessible	StP administrator's password differs from user passwords and is not publicly accessible

Table 2: Repository Manager Security (Continued)

StP Required Passwords	Non-Secure	Partially Secure	Secure
<code><server>_password</code> ToolInfo variable setting	Set to a common user password in the ToolInfo file (StP does not prompt for password)	Set to a common user password in the ToolInfo file (StP prompts only for the Sybase sa or Microsoft Jet Admin password, when appropriate)	Unset in both the ToolInfo and in all local user environments (StP will prompt for password)
Password required to start StP?	None	None	User or system administrator (sa or Admin) password required
Password required to execute StP administration commands?	None	Sybase sa or Microsoft Jet Admin password required for most repository manager-level administration commands No password required for system-level administration commands applied to user's own systems	Sybase sa or Microsoft Jet Admin password required for most repository manager-level administration commands No password required for system-level administration commands applied to user's own systems

Password Syntax

A password consists of six or more characters (no more than 14) and can contain any character except ASCII character 0 (null). Passwords are case sensitive.

If the password contains any non-alphanumeric characters, it must begin with an alphabetic character and the entire password must be quoted when it is set, though not when it is used.

If you are assigning passwords for Sybase, avoid using Sybase reserved words, such as “read,” “grant,” “use,” “alter,” and so on. For more

information on password restrictions for Sybase, see the Sybase documentation, which you can access on the Sybase web site, at <http://www.sybase.com>. For information on setting passwords for Microsoft Jet, see the Microsoft Access on-line help.

Changing the System Administrator's Password

The StP administrator has his or her own user name and user password. Additionally, the StP administrator is responsible for the system administrator's password.

When you first install StP, the password for all Sybase users is "welcome", including the StP administrator, which makes the StP environment unsecure.

StP does not assign a default password for Microsoft Jet users.

To change the Sybase system administrator's password:

1. From the **Repository** menu, choose **Manage Users > Repository Manager > Change User(s) Sybase Password**.
2. In the **Change User(s) Sybase Password** dialog box, type **sa** in the **User Name(s)** field.
3. Click **OK**.
4. If prompted, enter the system administrator's password.
5. In the **User Password** dialog box, enter the new value at the initial prompt, and again when prompted for verification.
6. Click **OK**.

Note: If you are using Sybase Adaptive Server or SQL Server, do not change the `<server>_password` ToolInfo variable setting to the system administrator's password, because doing so creates a non-secure repository manager.

Alternatively, you can use the **repos_change_password** function in a QRL script to change the password for user **sa**. For details, see *Query and Reporting System*.

Changing User Passwords

A user's password is originally set when the user is added to the repository manager. If you do not supply a password when you add a user, the user's password defaults to "welcome". See "Passwords" on page 2-27 for more information.

The StP administrator can change any user's password. A user without access to the system administrator's password can change only his or her own password.

You can set Sybase user passwords using the **Change User(s) Sybase Password** command. For a Sybase repository, the current value of the `<server>_password` ToolInfo variable determines whether or not the user must enter a password to access StP. For more information, see "Passwords for Sybase" on page 2-28.

To set Sybase user passwords:

1. From the **Repository** menu, choose **Manage Users > Repository Manager > Change User(s) Sybase Password**.
2. In the **Change User(s) Password** dialog box, type the user names in the **User Name(s)** field.
To assign one password to multiple users, separate names by a space.
To assign different passwords to multiple users, invoke this command separately for each user name and password combination.
3. Click **OK**.
4. If prompted, enter the system administrator's or your own password.
5. In the **User Password** dialog box, enter the new password, click **OK**, and enter the new password again to confirm.
6. Click **OK**.

Alternatively, you can use the **repos_change_password** function in a QRL script to change the password for any user. For details, see *Query and Reporting System*.

For Sybase repositories, if you are changing the common password shared by all users in a partially secure repository manager, reset the `<server>_password` ToolInfo variable to the new password. For a secure repository manager, in which each user or user group has a unique

password, unset `<server>_password` in the main ToolInfo file (see “User Environments” on page 2-8).

Note: Some users may choose to set `<server>_password` to their own password in the local environment to avoid being prompted for a password on system-level administration tasks in their own systems. However, doing so creates a partially secure rather than a secure repository manager, unless the setting in the ToolInfo file unconditionally overrides the users’ individual settings.

After a password change, the user must exit and restart StP in order to use the editors.

Troubleshooting Password Changes

If you receive error or warning messages after changing a password, consult the descriptions of possible errors and the corrective procedure discussed in this section.

Failing to Change the User Password

If a user or StP administrator fails to change or unset the `<server>_password` or `msjet_password` ToolInfo variable after changing a user’s password, the following warning messages appear:

For Sybase Adaptive Server or SQL Server:

```
Warning: ToolInfo Variable <server>_password does not match
the new password, setting should be changed
```

For Microsoft Jet:

```
Warning: ToolInfo Variable msjet_admin_password does not
match the new password, setting should be changed
```

Failing to Exit StP After a Password Change

If the user does not exit and restart StP after a password change, there will be a mismatch between the current value of `<server>_password` and

the user's new password. The next time the user attempts to execute an StP command, the following error messages appear:

```
10008: Unable to open current repository
10006: Not a valid login/password
1009: Unable to initialize repository manager
```

Correcting Problems Due to Password Changes

To correct either of the preceding problems stemming from a password change:

1. Exit StP.
2. Unset or reset the `<server>_password` or `msjet_password` ToolInfo variable, as appropriate (see “Summary of Security Level Determinants” on page 2-30).
3. Restart StP.
4. Type the new password, if prompted for it.

3 **Administering StP**

This chapter describes some basic StP administration tasks required to:

- Manage an StP multi-user environment
- Maintain the data in the repository manager and in an individual system

Topics covered are as follows:

- “Configuring StP Interprocess Communications” on page 3-1
- “Managing Sybase Database Placement” on page 3-11
- “Maintaining the Repository Manager” on page 3-14
- “Managing a System” on page 3-17
- “Backup and Recovery” on page 3-24

See Chapter 2, “StP Users,” for information about creating a system, changing system ownership, and managing repository manager users and system users.

For information on interacting directly with Sybase Adaptive Server or SQL Server, see *StP Guide to Sybase Repositories*. You can also refer to the Sybase documentation, which is available on their website at <http://www.sybase.com>.

Configuring StP Interprocess Communications

All StP tools (clients) connect to a central communication process, which routes messages between StP tools and provides addresses in response to connection requests. This communication process is called the StP

Message Router, or **msgd**. (See also “msgd” on page B-15.) The message router allows StP clients, such as the StP editors, to exchange messages in order to synchronize information about the StP system currently in use. A single message router can handle message passing between StP tools invoked on one or more StP systems by one or more users. To operate correctly, however, StP requires that no more than one message router run at any given time for any given StP system.

The existence of multiple message routers for the same StP system or an unexpected break in communications can result in data corruption and inconsistent behavior of StP (see “Troubleshooting Message Router Problems” on page 3-7).

Table 1 describes the various ways you can configure StP to handle message routing.

Table 1: StP Message Router Start-up and Connection Options

Message Router Start-up Options	Client Connection Options	Recommended For
StP autostart—Started when needed by an StP client on the local machine if it can’t find the message router already running elsewhere in the network (default start-up method).	Broadcasting—An StP client broadcasts a msgd connection request as needed, and if the message router is found, connects to it. This is the default behavior if the <i>msgd_host</i> ToolInfo variable is not set.	Single-user environments and most multi-user environments without subnets
Nominated host—Started manually or by Windows NT on a nominated host and runs on the nominated host continuously.	Direct connection—As needed, StP clients on all user machines connect directly to the message router running on the nominated host, using the <i>msgd_host</i> ToolInfo variable to determine the host name.	Large multi-user environments with a designated server; Multi-user environments in which StP users are dispersed over more than one subnet

The *msgd_host* ToolInfo Variable

It is recommended that you use the *msgd_host* ToolInfo variable to set the message router connection, not the IDE_MSGD_HOST environment variable.

For compatibility with previous versions, StP supports the IDE_MSGD_HOST environment variable. You can set the message router connection in IDE_MSGD_HOST, but it is easier to manage StP settings from a single location, the ToolInfo file, than from the different locations where environment variables are often set.

Note: The IDE_MSGD_HOST environment variable takes precedence over *msgd_host*. If you have questionable results regarding the message router host, check to see if there are any IDE_MSGD_HOST settings in your environment that may conflict with the *msgd_host* setting.

Message Routing in a Single-User Environment

For single-user StP environments, the default StP autostart and broadcast connection methods are recommended to conserve system resources that are only needed when StP is running. No special configuration is required to use the default StP autostart and broadcast connection methods for the StP Message Router. These are configured automatically when installing StP from the **Typical** or **Compact** installation option.

When installing StP for a single user from the **Custom** installation option:

- Enter “no” when asked if you want this machine to be the nominated host for the StP Message Router.
- Do not enter a host name when prompted for the host on which the message router will run. This leaves the *msgd_host* ToolInfo variable undefined.

If *msgd_host* or IDE_MSGD_HOST is not defined, when the user invokes StP it uses the broadcast method for connecting to the message router, by default.

Message Routing in a Multi-User Environment

For multi-user environments, running the StP Message Router on a nominated host provides the best results. If you already have a machine designated as a server that provides other services, you may want to run the message router on that machine.

Multi-User Environments without Subnets

If the environment does not include subnets, or if StP is used exclusively on one subnet, you can use the StP autostart method to run the message router. However, message router problems can occur if the user who first started StP logs off his or her machine, which causes the message router to terminate (see “Losing a Connection to the Message Router” on page 3-9). Running the message router on a nominated host is the preferred method.

When running the message router on a nominated host, StP clients can either broadcast a connection request or use the *msgd_host* ToolInfo variable to connect to the message router.

Note: You should configure all user environments to use the same connection method.

Multi-User Environments with Subnets

In multi-user environments where StP users are dispersed over more than one subnet, the StP Message Router should run on a nominated host, and users should connect to it directly, using the *msgd_host* ToolInfo variable.

Certain problems can arise if users on different subnets working on the same StP system use the broadcast method to connect to the message router. Gateways between subnets may be configured to allow or disallow broadcasts to cross subnet boundaries (the default disallows them). If the gateway disallows broadcasts across subnet boundaries, an StP client broadcasting for a message router connection cannot detect a message router in another subnet, and may start a second message router. Using the broadcast connection method, StP tools on these separate subnets cannot communicate or synchronize information with each other, which can result in data corruption and other StP errors.

For more information, see “Message Routing Problems in a Multi-Subnet Environment” on page 3-9.

Note: If you set a nominated host to run the StP Message Router, all users must connect to it using the same *msgd_host* ToolInfo value.

Multiple StP Releases on the Same Network

If StP 7.x coexists on the same network with StP 2.x or StP 1.x, each requires its own StP Message Router running on a separate nominated host. If the *msgd_host* ToolInfo variable is set in the user environment, make sure it points to the host running the message router for the correct StP release. If *msgd_host* is not set, StP tools broadcast connection requests to the message router for the appropriate StP release.

Note: Do not use these different releases of StP to access the same StP system concurrently; doing so can corrupt data.

Running Message Router on a Nominated Host

When you run the StP Message Router on a nominated host, it potentially serves all users working on all StP systems on that network. It knows which messages belong to which StP system and routes messages appropriately.

You can set up the message router to restart automatically when the host machine is rebooted. If you prefer, you can start the message router manually.

Note: If you change the host on which the message router is running, you must also change any existing *msgd_host* settings to point to the new host.

Running Message Router as a Windows NT Service

On Windows NT, the StP Message Router is always installed as a Windows NT service. However, you can run it as a regular program as well.

Configuration options for the StP Message Router are available when installing StP from the **Custom** installation option. When asked during

StP installation if you want this machine to be the nominated host for the StP Message Router:

- Answer “yes” when installing StP on the selected host for the message router. This installs the message router in automatic start-up mode, so that it starts every time Windows NT starts.
- Answer “no” when installing StP on all other machines. This installs the message router as a Windows NT service in manual start-up mode.

To change the start-up mode for the message router after StP has already been installed:

1. From the Windows NT **Start** menu, choose **Settings > Control Panel**.
2. Double click the **Services** icon.
3. In the **Services** dialog box, select **StP Message Router** and click **Startup**.
4. In the **Service** dialog box, select **automatic** or **manual**, as desired.
5. Click **OK**.

Connecting to the StP Message Router

As described in Table 1 on page 3-2, depending on your particular StP environment, StP tools can connect to the StP Message Router either:

- By broadcasting a connection request
- Directly, using the *msgd_host* setting for the message router’s host name

Broadcasting for a Message Router Connection

Broadcasting a connection request is the default StP Message Router connection method if *msgd_host* is undefined. StP clients are able to locate and connect to the message router wherever it happens to be running on the network by broadcasting a connection request over the network. When found, the client opens a socket connection to the message router for its communication needs.

If no existing message router is located, the StP client broadcasting the connection request starts one on the local host.

Connecting Directly Using the msgd_host Setting

When *msgd_host* is set in a user's environment, StP processes invoked by that user do not broadcast for the StP Message Router location, but rather contact it directly on the specified host. Connections to the message router across subnet boundaries occur without a problem.

The StP installation program prompts for the name of the host on which the message router is to run and assigns that host name to the *msgd_host* variable in that installation's ToolInfo file. By default, if you do not enter a host name for the message router during StP installation, *msgd_host* remains undefined in the ToolInfo file.

To change the *msgd_host* setting for StP after it has been installed, you can edit the StP installation's ToolInfo file. For example, to set *msgd_host* to host rhapsody, the ToolInfo file should contain the following lines:

```
msgd_host=rhapsody
```

Alternatively, you can set *msgd_host* directly in each user's personal ToolInfo file. The *msgd_host* value must be the same for all users.

Troubleshooting Message Router Problems

The StP Message Router is designed so that if more than one message router starts up, they negotiate with each other to decide which one will continue running. Any other message routers then terminate automatically. This mechanism automatically resolves most problems with multiple message routers. However, this mechanism does not work across subnet boundaries that disallow cross-subnet broadcasting.

Table 2 summarizes some of the message router problems that may occur within a multi-user environment.

Table 2: Summary of Message Router Problems

Symptom	Configuration	Possible Cause	For Help, See
The message router host connection is different from the <i>msgd_host</i> setting	Multi-user or single-user configuration.	There may be IDE_MSGD_HOST environment settings that conflict with <i>msgd_host</i> .	"The msgd_host ToolInfo Variable" on page 3-3
An error message indicates that the msgd connection has been lost.	Multi-user network. Message router autostarted by StP. Users broadcast connection requests.	The user who first started StP logged off the machine, thereby terminating the message router for all users connected to it.	"Losing a Connection to the Message Router" on page 3-9
The message router terminates immediately upon startup on a nominated host.	Multi-user network. Message router runs on nominated host. All users connect either by broadcasting or by using <i>msgd_host</i> .	Another message router is already running on the same network or subnet.	"Message Router Terminates on Startup" on page 3-9
Double-clicking on a list on the Desktop (such as a list of diagrams) starts the editor but does not load the diagram.	Multi-user network with subnets that disallow cross-subnet broadcasting. Message router is autostarted by StP. Users broadcast connection requests. Users on different subnets are accessing the same StP system.	Multiple message routers are serving StP users who are accessing the same StP system from different subnets.	"Message Routing Problems in a Multi-Subnet Environment" on page 3-9
Running Edit Object Annotation for an object starts the OAE, but cannot load the annotation.			
Running Rename Object Systemwide results in incomplete renaming, creating new objects without deleting the old, and so forth.			

Losing a Connection to the Message Router

Using the broadcast connection method, multiple users within a single network or subnet connect to the StP Message Router running on the machine on which StP was first started. If the initial user logs off his or her machine, the message router started by that user terminates, and the other StP users receive an error message telling them their connection to **msgd** has been lost. Users may also receive this error message if the message router on a nominated host terminates for any reason.

If a connection to the message router is lost, the user must exit and restart StP to start or connect to a new message router.

Message Router Terminates on Startup

StP Message Router may terminate on startup if it detects an existing message router on the network (unless one of them is for an StP release prior to StP 7.x, in which case both routers are required). This mechanism ensures that only one message router is running at a given time in an StP multi-user network environment without subnets. It works even when the message router is started explicitly by a system administrator.

If the message router that terminated is the one that should be running on the nominated host, you should determine if and where any other message routers are running on this network (see “Determining Where the Message Router is Running” on page 3-10).

Check to make sure which message router(s) should be running and follow the instructions in “Correcting a Multiple Message Router Problem” on page 3-11.

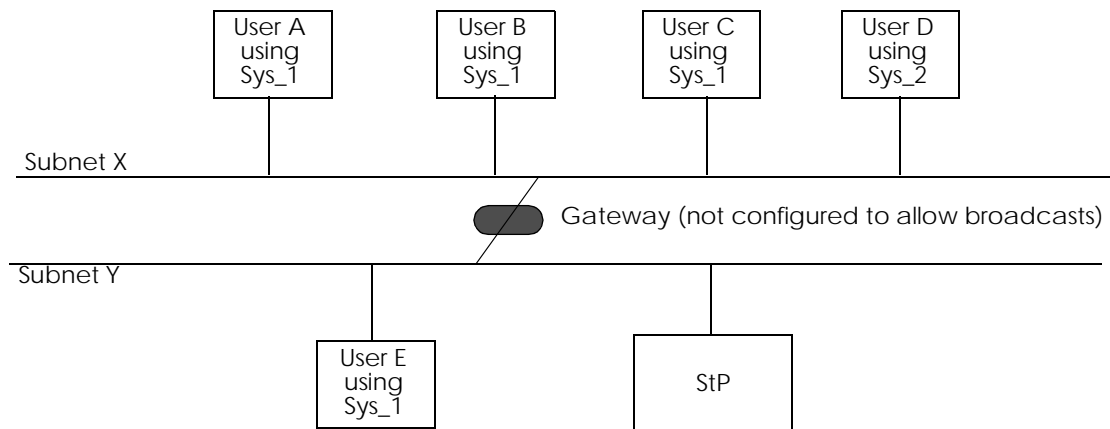
Message Routing Problems in a Multi-Subnet Environment

Message routing problems due to multi-subnet configurations occur only if all of the following conditions apply:

- You have more than one subnet
- Your gateway is configured to disallow broadcasts across subnet boundaries
- Users access the same StP system from different subnets simultaneously

Figure 1 shows a multi-subnet environment in which message router problems can occur.

Figure 1: StP users and tools on multiple subnets



In Figure 1:

- When Users A, B, or C access StP, a message router runs on Subnet X.
- When User E accesses StP, another message router runs on Subnet Y, since the presence of the first message router is not detected.
- Users of system *Sys_1* on one subnet will not detect the presence of a message router on the other subnet and therefore cannot receive messages from the router nor pass messages to it.
- User D on Subnet X uses system *Sys_2* and there are no *Sys_2* users on Subnet Y; this poses no message router problems, since there is no need for cross-subnet message passing.

Determining Where the Message Router is Running

To find out where the StP Message Router is running, run the **msgdiag** command from a command line. Run **msgdiag** for each subnet (if any) in the network.

The **msgdiag** command broadcasts an address request for the message router throughout the network, or within an existing subnet, if any, and

lists all replies it receives. See “msgdiag” on page B-17 for more information.

Correcting a Multiple Message Router Problem

To correct a multiple StP Message Router problem:

1. If StP is currently in use, request all users to exit StP.
2. Kill all existing **msgd** processes, if any, on each subnet involved.

For example, type:

```
stpem -C '>q'
```

The **stpem** command requests the termination of any message router that receives it within that subnet. For more information, see “stpem” on page B-37.

3. Decide on which host machine the message router should run (see “Running Message Router on a Nominated Host” on page 3-5).
4. In the ToolInfo file for each StP installation, set **msgd_host** to the name of the chosen message router host. Also make sure that **msgd_host** is not set to a different host in any user’s environment. See “Connecting Directly Using the msgd_host Setting” on page 3-7.
5. Start the StP Message Router on the host (see “Running Message Router as a Windows NT Service” on page 3-5).
6. Inform StP users they can restart StP.

Managing Sybase Database Placement

Sybase users can create system repositories (user databases) on a default or specified database device. When you install StP, the default device is *stp_device*. You can use the *syscreate_device* ToolInfo variable to control database placement for StP systems.

If the *syscreate_device* ToolInfo variable is not set, commands that create or expand a system allocate space for that system on any default device that has space available.

To create a system repository on a specific device, set *syscreate_device* before you create the system. StP then performs all repository tasks on the specified device, including any future expansion.

The environment variable equivalent of *syscreate_device* is *DBDEVICE* (see “ToolInfo Variables Versus Environment Variables” on page 2-9).

Creating Sybase Systems on Default Devices

To create systems on default devices, set the *syscreate_device* ToolInfo variable as follows:

```
syscreate_device=default
```

The next time you start StP, and create a system, StP will use the default Sybase device. In Adaptive Server or SQL Server, you can specify multiple devices; when the first device is filled, the next available device listed in the *sysdevices* system table is used. To designate default devices in Sybase, see the *StP Guide to Sybase Repositories*.

Advantages of creating a system on a default device are:

- You can create or expand the system repository on a default device without knowing specifically which device has space available.
- Space that is regained by removing a system repository is automatically reused when another system is created or expanded.

If a system repository grows too large for the originally allocated space, you can expand the size of the repository. If the system repository was created on an Adaptive Server or SQL Server default device (*syscreate_device* was not set when the system was created), and the device later becomes too full, StP can expand the repository to any additional default device that has space available. The expansion provides additional space that can be used by both data (repository) and log (transaction log).

Creating Systems on Specific Devices

You can create system repositories on a specific device by setting the *syscreate_device* ToolInfo file as follows:

```
syscreate_device=<device_name>
```

For example:

```
syscreate_device=stp_dev_9
```

The next time you start StP and create a new system, StP creates that system on the device you specified. However, when that device is filled, any StP commands will fail, including the creation of new systems.

The new system will have only its data fragments (6 MB by default) created on the database device you set as the default. The log fragment (2 MB by default) for the databases you create will be placed on any default device with space available when you create the system.

By default, StP also creates a 2-MB log fragment on any default device with space available when you create a new system. It is generally a good idea to store the transaction log on a different device from the repository data to enable data recovery if the repository's device fails. To specify a particular log device, set the *syscreate_log_device* ToolInfo variable before creating a new StP system with *sys_create* or before starting StP before creating the system from the Desktop.

Any specified device must already have been added to the Adaptive Server or SQL Server. See *StP Guide to Sybase Repositories* for information on creating Sybase devices.

You can also set the *syscreate_size* and *syscreate_log_size* ToolInfo variables to specify the size of the data and log fragments for the system to be created.

Note: If *syscreate_device* is set when the system is created, any future system expansion occurs only on the originally specified *syscreate_device*. If the device becomes full, it cannot expand to other devices. The repository must remain on the originally specified *syscreate_device*.

Be sure you unset the *syscreate_device* and *syscreate_log_device* ToolInfo variables after creating the system, so that future systems are created on the default devices.

Maintaining the Repository Manager

This section describes the StP administration tasks required to maintain the information in the repository manager. It covers:

- Routine maintenance
- Listing repositories
- Showing space available on database devices

You must be the StP administrator to perform these tasks.

Routine Maintenance

Managing the space in the repository manager involves performing routine maintenance tasks at regular intervals. These tasks include:

- Deleting old temporary tables
- Dumping the StP repository manager's transaction log (Sybase only)
- Updating the statistics used by the Adaptive Server and SQL Server indexes and optimizer (Sybase only)

Note: Microsoft Jet systems must be closed to perform routine maintenance.

To perform routine maintenance

1. From the **Repository** menu, choose **Perform Manager Maintenance**.
2. In the **Perform Repository Manager Maintenance** dialog box, select either **Sybase** or **MS Jet** from the **Server** menu.
3. Click **OK**.
4. If prompted, enter the system administrator's password.
In the lower left corner of the Desktop dialog box, a running status of the actions performed appears.

Alternatively, you can use the **repos_maint** function in a QRL script to perform routine maintenance. For details, see *Query and Reporting System*.

Listing Repositories

You can list all the repositories under a repository manager with the **List Repositories** command.

To list repositories:

1. From the **Repository** menu, choose **List Repositories**.
2. From the **List Repositories Under Manager** dialog box, choose either **Sybase** or **MS Jet** from the **Server** menu.
3. If prompted, enter the system administrator's password.
4. Click **OK**.

Alternatively, you can use the **repos_list_reps** function in a QRL script to list repositories. For details, see *Query and Reporting System*.

Showing Space Available on Devices

If you are using Sybase Adaptive Server or SQL Server as the repository, you can use the **Show Sybase Server Space** command to display space usage on the repository manager database devices.

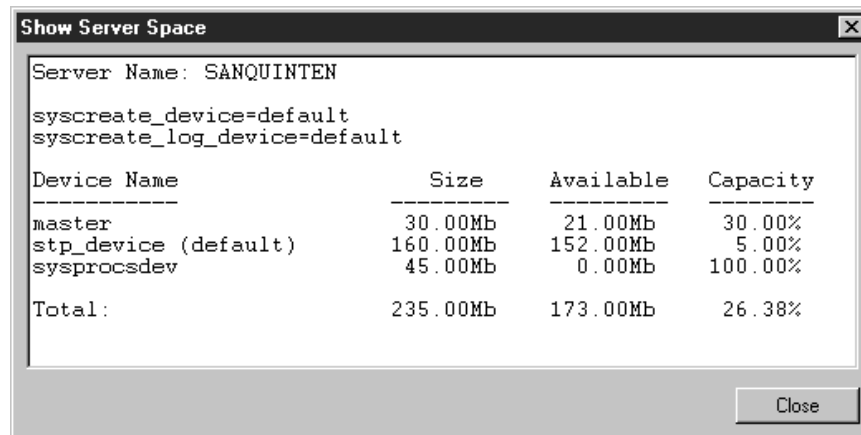
To display Sybase server device usage:

1. From the **Repository** menu, choose **Show Sybase Server Space**.
2. In the **Show Space Available Under Sybase Server** dialog box, choose one of the following options:
 - **Devices For Current Settings**—Reports space usage only on devices created by the **Create System** command (the *syscreate_device* and *syscreate_log_device* ToolInfo variables, if set, and any default devices, if either of these variables is unset).
 - **All Devices**—Reports space usage on all devices in the repository manager, including the *master* and *sysprocsdev* devices used by the repository manager to manage StP systems.
3. If prompted, enter the system administrator's password.
4. Click **OK**.

Alternatively, you can use the **repos_show_space** function in a QRL script to show device space usage. For details, see *Query and Reporting System*.

The command shows the device name, size, available space, and percentage full for the database device(s) in the current repository manager. The output in the following example is for the **All Devices** option:

Figure 2: Available Space on a Sybase Repository



The screenshot shows a window titled "Show Server Space" with a close button in the top right corner. The window displays the following information:

```
Server Name: SANQUINTEN
syscreate_device=default
syscreate_log_device=default
```

Device Name	Size	Available	Capacity
master	30.00Mb	21.00Mb	30.00%
stp_device (default)	160.00Mb	152.00Mb	5.00%
sysprocsdev	45.00Mb	0.00Mb	100.00%
Total:	235.00Mb	173.00Mb	26.38%

A "Close" button is located at the bottom right of the window.

The first three lines of output include the name of the repository manager Adaptive Server or SQL Server (the value of the DSQUERY environment variable) and the device names for the *syscreate_device* and *syscreate_log_device* ToolInfo variables, if set, or “default” as the device name(s) if not set. There may be more than one default device for the repository manager, in which case the output displays usage data for each default device, indicated by “default” following the device name.

If the repository manager database devices are getting too full, you can add devices, as explained in *StP Guide to Sybase Repositories*. To make an additional device available as a default device for system creation and expansion, use the **sp_diskdefault** Sybase system procedure, also discussed in *StP Guide to Sybase Repositories*.

Managing a System

It is the responsibility of the system owner or the StP administrator to manage the data and storage in a system. This can involve:

- Cleaning up the system
- Expanding the repository
- Deleting repository data
- Destroying a system repository
- Destroying a system

Cleaning Up

You can reclaim space in a system by deleting unreferenced objects and truncating history files on a regular basis.

Deleting Unreferenced Objects

When a model is fairly stable, you may want to make more room in the system repository by removing unreferenced objects. An unreferenced object is one that is not represented in any diagram or table in a system.

Before deleting unreferenced objects, make sure no one is currently using the system. To find the current users of the system, choose **Repository > Manage Users > Current System > List Current Users**.

To delete unreferenced objects:

1. From the **Repository** menu, choose **Maintain Systems > Delete Unreferenced Objects in System Repository**.
2. In the **Delete Unreferenced Objects** dialog box, select from the following options.
 - **Don't Delete If Annotated** deletes only unreferenced objects without annotations.
 - **Ask for Confirmation** prompts for confirmation before data is deleted. When you select this option, you can expect a large number of prompts.
3. Click **OK**.

Occasionally, you may get an error message such as:

```
Can't delete node with id: 503 because it has links
```

In this case, you need to execute the **Delete Unreferenced Objects in System Repository** command again. On the first pass, the command deleted any links that caused this error message to appear. On the second pass, the command deletes the now completely unreferenced node.

You can also delete unreferenced objects with the StP Utility **clean** command. See “clean” on page 4-6 for more information.

Truncating History

Whenever a user creates, updates, saves or views a file, a file history object is created. Over time, these file history objects accumulate.

To remove obsolete file history objects:

1. From the **Repository** menu, choose **Maintain Systems > Truncate History for Current System**.
2. From the **Truncate History for Current System** dialog box, choose **Last History Object per File** or **Last History Object per User** from the list.
3. Click **OK**.

In the lower left corner of the Desktop dialog box, a status reports the number of obsolete history files to be deleted and the number of file history objects to be retained. The objects are then deleted except the most recent one, for each file.

Alternatively, you can use the **sys_trunc_hist** function in a QRL script to truncate history files.

Expanding a Microsoft Jet Repository

When a system is created on a Microsoft Jet repository, the size of the repository depends on how large the system is. As the system grows, Microsoft Jet allocates space to accommodate the new size. You do not need to change any settings in StP or Microsoft Jet.

To manage the size of the Microsoft Jet repository:

- Perform routine maintenance on Microsoft Jet systems, which compacts the size of allocated space (see “Routine Maintenance” on page 3-14)
- Delete unreferenced objects (see “Deleting Unreferenced Objects” on page 3-17)

However, if the repository grows too large, you should upgrade it to a Sybase repository. Symptoms of an overgrown Microsoft Jet database are performance degradation and data corruption. Another factor to consider is the number of users who have access to the Microsoft Jet database. If the size is not that large, yet there are more than five users, you may still have performance and data corruption problems.

To upgrade a Microsoft Jet repository to Sybase, determine where you want to create the Sybase system:

- In place, which replaces the existing Microsoft Jet system
- In a new location, which preserves the original Microsoft Jet system

Upgrading to Sybase In Place

To upgrade a Microsoft Jet system to a Sybase system in place, which overwrites the original system:

1. Destroy the current repository using the **Destroy System Repository** command.
See “Destroying a System Repository” on page 3-23 for details.
2. Rebuild the repository by using the **Recover System Repository** command.

When the **Select Repository Type** window appears, choose **Yes** to create a Sybase system. See “Rebuilding the Repository from System Files” on page 3-27 for details.

Upgrading to Sybase in a New Location

To upgrade a Microsoft Jet system to a Sybase system and save it in a new location, preserving the original system:

1. Create a Sybase system in a new location.

2. Copy the subfolders in the original Microsoft Jet system folder to the new Sybase system folder.
3. Rebuild the repository for the Sybase system by using the **Recover System Repository** command.

When the **Select Repository Type** window appears, choose **Yes** to create a Sybase system. See “Rebuilding the Repository from System Files” on page 3-27 for details.

Expanding a Sybase Repository

When a system is created on a Sybase Adaptive Server or SQL Server repository, the size of its repository is set to the size designated by the value in the **New System** dialog box or the `syscreate_size` ToolInfo variable. If this variable is not set when the system is created, the size of the repository defaults to 6 MB.

If you are attempting to save to the repository and receive a message that the repository is full, you can create more space by deleting unreferenced objects and truncating obsolete history files, as described in the preceding section. If there is still not enough space, you can expand the repository using the **Expand Current System Repository** Desktop command. The expansion provides additional space that both the repository and the transaction log can use.

By default, if the `syscreate_device` ToolInfo variable is not set, repository expansion occurs on the device designated as a default device in your Sybase Adaptive Server or SQL Server installation. This allows for repository expansion to additional designated default devices, in the event that the original device on which your repository resides becomes full.

If `syscreate_device` is set, the expansion occurs only on the device defined by `syscreate_device`, on which your system’s repository already resides. If this device is full, no expansion occurs, regardless of any available space on the Sybase server-designated default devices. The repository must remain on the originally specified `syscreate_device`.

Before attempting to expand the repository, check its current size and percentage full with the **Show System Space** Desktop command (described in “Showing Space for the Current System” on page 2-25). If you know the StP administrator’s password, you can check the available

space on the devices in the repository manager, if necessary (see “Showing Space Available on Devices” on page 3-15).

To expand the repository from the StP Desktop:

1. From the **Repository** menu, choose **Maintain Systems > Expand Current System Repository**.
2. In the **Expand Current System Repository** dialog box, type the number of megabytes to be added in the **Size of Expansion** field. The default is 2 MB.
3. Click **OK**.

Alternatively, you can use the **sys_expand_rep** function in a QRL script to expand a repository. For details, see *Query and Reporting System*.

Adding Files to the Repository

The **Add New Files to Current System Repository** command:

- Updates out-of-date ASCII files and their corresponding diagrams, tables, or annotations in the repository
- Copies StP files from one system repository into another

This is useful if you have diagrams, tables, or annotations from one StP system that you want to include in another system.

This command includes a force option. If you use the force option, the file is added, even if it already exists in the repository. The force option is useful if a newer version of an existing file has been manually copied into the file system. In this case, the older repository data needs to be updated with the newer version of the ASCII file.

To add files to a repository:

1. From the **Repository** menu, choose **Maintain Systems > Add New Files to Current System Repository**.
2. In the **Add New Files to Current System Repository** dialog box, type a filename or filenames and StP filetype or filetypes.

If you do not specify filenames or filetypes, all files are added. These fields accept wildcard characters. If there is more than one filename or filetype, they can be separated by spaces or commas.

3. If desired, click the following options:
 - **Stop on Error** updates all the files it can and reports any non-writable files. Without this option, the command fails if it encounters any non-writable files.
 - **Force** updates all files and does not display a confirmation dialog box. If you suspect that some of the current files are newer than the files you are replacing them with, do not click the **Force** option.
4. Click **OK**.

Alternatively, you can use the **sys_update** function in a QRL script to add files to a repository. For details, see *Query and Reporting System*.

Deleting Repository Data

If the data in a system has become obsolete and you want to start working in a clean system, you can delete all the data in the repository without destroying the repository structure.

This is particularly useful for cleaning out data created by a user running an StP tutorial.

To delete repository data from the StP Desktop:

1. From the **Repository** menu, choose **Maintain Systems > Delete Repository Data in Current System**.
2. In the **System Clean Repository** confirmation dialog box, click **Yes** to confirm or **No** to cancel.

Alternatively, you can use the **sys_clean_rep** function in a QRL script to delete repository data. For details, see *Query and Reporting System*.

To delete only unreferenced objects, see “Deleting Unreferenced Objects” on page 3-17. To delete one or more specified diagrams or tables and their associated files, use the StP Utility’s **delete** command, described under “delete” on page 4-8, or use the **Delete Diagram** or **Delete Table** command on the Desktop **File** menu for the appropriate editor.

Destroying a System Repository

You can destroy a system repository without destroying the files that describe the rules for creating the repository. You may want to do this if the repository has become corrupted and you want to regenerate it. See “Rebuilding the Repository from System Files” on page 3-27 for information about regenerating a repository.

If you are destroying a repository to create more repository space temporarily, back up the repository first. See “Backup and Recovery” on page 3-24 for information about backing up a repository.

The **Destroy System Repository** Desktop command deletes all the data in the repository and then destroys the repository’s data structures.

A repository cannot be destroyed while users are accessing it. If you attempt to destroy a repository that is in use, a window appears displaying a list of current users.

To destroy a system repository:

1. From the **Repository** menu, choose **Maintain Systems > Destroy System Repository**.
2. In the **Destroy System Repository** dialog box, select the system name.
3. Click **Destroy Rep.**
4. In the **System Repository Destroy** confirmation dialog box, click **Yes** to destroy the system, or **No** to cancel.

Alternatively, you can use the `sys_destroy_rep` function in a QRL script to destroy a repository. For details, see *Query and Reporting System*.

Destroying a System

When you no longer need a system, you can destroy its repository and all its associated files with the **Destroy System** Desktop command. After a system is destroyed, it cannot be regenerated.

A system cannot be destroyed while users are accessing it. If you attempt to destroy a system that is in use, a window appears displaying a list of current users.

To destroy a system:

1. From the **File** menu, choose **Destroy System**.
2. In the **Destroy System** dialog box, specify the project directory and enter the name the system to destroy.
3. Click **OK**.
4. In the **System Destroy** confirmation dialog box, click **Yes** to destroy the system or **No** to cancel.

Alternatively, you can use the **sys_destroy** function in a QRL script to destroy a system. For details, see *Query and Reporting System*.

Backup and Recovery

In Sybase, if the repository manager crashes because of an Adaptive Server or SQL Server system failure, the repository is automatically recovered when the repository manager is restarted. The underlying Adaptive Server or SQL Server automatic recovery mechanism provides this functionality.

When the Sybase server is restarted, all transactions committed before the failure are written to the repository and all transactions not committed before the failure are rolled back. Sybase Adaptive Server or SQL Server uses the transaction log of committed and uncommitted transactions to direct automatic recovery. Automatic recovery occurs only when there is a system failure on the server.

In Microsoft Jet, if a system is corrupted, the Microsoft Jet database will attempt to repair it.

If the repository data is lost or damaged due to a storage device failure, or the data has become corrupted in some other way, there is no automatic recovery. In this event, StP provides the following methods for restoring data to the repository:

- Restore a repository from the most recent StP backup, consisting of ASCII dump files.
- Rebuild a corrupted repository from the system ASCII files, if these files are intact and up-to-date.

- Use a combination of the two procedures.

Optionally, you can load dump files into a different system repository from the one that produced the dump files. Dump files can thus be used for copying a system repository to a different project or machine as well as for backup and recovery. For this reason, it is important to back up repositories on a regular basis.

Backing Up a Sybase Repository to Dump Files

You can create an StP backup by dumping the entire Sybase repository to ASCII files. This performs a backup of the database underlying the repository, and is the recommended method for creating StP backup files. An alternative is to set up a Sybase Backup Server (see the Sybase documentation, which is available from the Sybase website at <http://www.sybase.com>).

Note: StP cannot use backup files produced by other methods, such as operating system backup utilities.

You should dump the repository to ASCII files on a regular basis. The frequency of backups depends on how critical the data is and how often it is updated. The repository can be dumped while users are accessing it. The dump files reflect the state of the repository at the time of the beginning of the dump.

The dump procedure stores the files that it creates in a dump directory that you specify. You can create a dump directory as a subdirectory of the system directory. The most recent dump overwrites any files that already exist in the specified dump directory. If you want to save the current dump directory, rename the files or move them to another directory before executing the dump.

Any user can dump a repository, but you must be the StP administrator or the owner of the system into which the files are being loaded to load a dump directory into a repository.

To dump the repository from the StP Desktop:

1. From the **Repository** menu, choose **Maintain Systems > Dump Current System Repository to Files**.

2. In the **Dump Current System Repository** dialog box, type the dump directory's pathname in the **Dump to Directory** field.
If the specified dump directory does not exist, StP creates it for you.
3. Click **OK**.

Alternatively, you can use the **sys_dump_rep** function in a QRL script to dump a repository. For details, see *Query and Reporting System*.

Restoring a Repository from Dump Files

In the event of media failure, you can restore a system repository from the most recent dump directory. This restores a repository to the state it was in at the time of the dump. Updates to the repository between the time of the dump and the time of the media failure are not recoverable through the StP dump and load facilities (see “Rebuilding the Repository from System Files” on page 3-27).

The repository being loaded does not have to be the same repository from which the files were originally dumped, although it can be. In either case, the load facility overwrites any data in the repository being loaded.

To load the repository from the StP Desktop:

1. Make sure there are no current users on the system (see “Listing Current Users for the Current System” on page 2-24).
2. Exit from any StP editors that may be running currently.
3. From the **Repository** menu, choose **Maintain Systems > Load Current System Repository from Files**.
4. In the **Load Current System Repository from Files** dialog box, type the dump directory's pathname in the **Load from Directory** field.
5. If desired, click the following options:
 - **Fast** drops all indexes and triggers in the repository before loading the files. If you do not specify the **Fast** option, the **Load Current System from Repository** performs a regular load, with all the indexes and triggers included. Remember that regular loads can be slower than fast loads.
 - **Load System User Info** loads user information, such as readers and writers of a system, with the system load.
6. Click **OK**.

7. In the **System Load Repository** confirmation dialog box, click **Yes** to reload the system or **No** to cancel.

Alternatively, you can use the **sys_load_rep** function in a QRL script to load a repository. For details, see *Query and Reporting System*.

Rebuilding the Repository from System Files

System data is stored in two places: in the system files, which are ASCII files, and in the system repository, which consists of objects that map to the underlying storage manager. The ASCII files contain the rules by which the objects are constructed, so it is possible to rebuild a repository from the information in the system files.

You may need to rebuild a repository if:

- The system repository is lost or corrupt.
- You need to copy or restore a system and you do not have a recent dump file.

You can verify the existence of a system repository with the **Check if System Repository Exists** Desktop command (described in “Checking for a Repository” on page 2-24).

When you rebuild a system repository, StP offers you the choice to:

- Remove all existing repository data before the recovery (essentially a full data recovery)
- Retain existing repository data and update it from differences found in the system files (essentially an incremental update or restore)

In either case, new information stored in the system ASCII files and shadow log files is added to the repository, and obsolete data in the repository, if any, is updated. Deletions occur only if you choose the option to remove all existing repository data prior to the recovery.

As this process executes, StP locks the system, so that other StP administration functions cannot be performed concurrently.

To successfully recover a repository, ensure that the user executing the script has write permissions on the file.

Rebuilding the repository entirely from the system files is somewhat costly, both in terms of time and CPU load. A recommended alternative is to use a combination of loading the data from dump files and recovering the most recent changes from the system files.

To rebuild an entire repository from system files, using the StP Desktop:

1. From the **Repository** menu, choose **Maintain Systems > Recover System Repository**.
2. In the **Recover System Repository** dialog box, type the project directory and the system name.
3. If you want to recover the system repository quickly, click the **Fast** option. The **Fast** option drops all indexes and triggers in the repository.
4. Click **OK**.
5. In the **Select Repository Type** window, choose **Yes** to create a Sybase system; choose **No** to create a Microsoft Jet system.
6. In the **Remove repository contents** confirmation dialog box:
 - Click **Yes** to remove any existing data
 - Click **No** to retain the existing data and update it from the system files

Alternatively, you can use the **sys_recover_rep** function in a QRL script to recover a system repository. For details, see *Query and Reporting System*.

4 Using StP Utility

StP Utility provides a set of commands that allow system owners to query, modify, and manipulate an individual system's repository and files.

Topics covered in this chapter are as follows:

- “StP Utility Subsystems” on page 4-1
- “Accessing the StP Utility” on page 4-2
- “Setting StP Utility Modes” on page 4-2
- “Getting Help on StP Utility Commands” on page 4-3
- “Using Character Strings with StP Utility” on page 4-4
- “StP Utility Main Level Commands” on page 4-5
- “Environment Subsystem” on page 4-18
- “Locking Subsystem” on page 4-19

StP Utility Subsystems

StP Utility commands are grouped into a Main (top level) set of general purpose commands and other groups of commands that apply to certain subsystems:

- Main—commands that manipulate objects in the repository and set or report on projects, systems, and system files.
 - Locking—commands that prevent simultaneous access to the same file in a multi-user environment.
-

- **Environment**—commands that check for certain discrepancies in system files and directories.

You can access the other subsystems through the Main level, or by choosing the subsystem on the Start StP Utility dialog box.

Accessing the StP Utility

To access StP Utility:

1. From the **Tools** menu, choose **StP Utility**.
2. In the **StP Utility** dialog box, select from the following options:
 - **Subsystem** lets you choose from **Main**, **Locking**, or **Environment**.
 - **Ask for Confirmation** displays a confirmation dialog box for each action that destroys data.

For more information on these options, see “Setting StP Utility Modes” on page 4-2.

3. Click **OK**.

A command window appears, displaying the StP Utility prompt:

To use StP Utility, enter StP Utility commands at the `->` prompt in the window, as described in the remainder of this chapter.

Setting StP Utility Modes

StP Utility modes determine the utility’s behavior, such as whether it asks for confirmation or reports its activities while executing a command. You set the utility’s modes by selecting options on the Start StP Utility dialog box (see “Accessing the StP Utility” on page 4-2) or by using the mode commands within StP Utility. You can set modes at the Main level of StP Utility or from within any of its subsystems.

Table 1: StP Utility Modes

Dialog Option	Set	Command	Description
Ask for Confirmation	On	I [interactive] on F [orce] off	Sets interactive mode on; prompts user for confirmation before executing any StP Utility command that destroys data.
	Off	F [orce] on I [interactive] off	Sets force mode on; executes StP Utility commands without asking for user confirmation.
Verbose	On	V [erbose] on Q [uiet] off	Sets verbose mode on; displays additional information about the tasks being performed during command execution.
	Off	Q [uiet] on V [erbose] off	Sets quiet mode on; suppresses the display of additional information about the tasks being performed during command execution.
Show Commands	On Off	(no equivalent)	When set on, echoes each command as it is executed.

To check the setting of a particular mode, enter the StP mode command (**I**, **F**, **V**, or **Q**), without the **on** or **off** option, as in the following example:

```
-> F
Force off
```

Getting Help on StP Utility Commands

To get help on a specific StP Utility command, type **help** followed by the name of a command in the current subsystem:

```
-> help copy
c[copy] [options] from_diag to_diag
Copy files matching pattern over system/project boundaries
Valid options for the command are:
    -e      Editor
            Set the current editor for this operation.
            This is the name of an editor in your environment.
            For a list of legal editors, execute the 'list' command
            in the env[ironment] subsystem.
    -T      Filetype
            Specify a filetype
    -v      Perform operation in verbose mode
    -f      Perform operation without asking for confirmation
    -n      Perform operation on annotations
-> █
```

To see a list of all applicable commands at the Main level or in any subsystem, type `help` or `?` at the StP Utility prompt, without specifying a command name.

Accessing Subsystem Command Groups

The following commands at the Main level provide access to the StP Utility subsystems:

- `env[ironment]`—environment subsystem
- `loc[k]`—locking subsystem

To leave a subsystem and return to the Main level of StP Utility, type `quit` at the prompt. If you type `quit` from the Main level, you exit StP Utility.

Using Character Strings with StP Utility

Some StP Utility commands accept full name or pattern character strings as arguments.

Full Names

Some StP Utility commands require the full name of a diagram as an argument. A full name may optionally specify the project and system of the diagram in addition to the diagram name.

The syntax of a full name is:

```
[ [<project_name>: ] <system_name>: ] <diagram_name>
```

If the <project_name> portion is omitted from the <diagram_name>, the current project is assumed. If the <system_name> is omitted, the current system is assumed.

Patterns

A pattern consists of a character string containing optional wildcard characters. The wildcard characters have special meanings that match characters other than themselves. Patterns are used for generating a list of filenames.

The * character matches any character string.

The ? character matches any single character.

The left square bracket ([) character introduces a range of characters, the right square bracket (]) terminates a range of characters, and the hyphen (-) character separates the initial and terminating characters in a range.

StP Utility Main Level Commands

This section describes the commands that are available from the main level of StP Utility and are common to all StP products. They are listed in alphabetical order by full command name. For information about commands that are specific to a particular StP product, invoke **help** with the command as its argument in StP Utility.

clean

The **clean** command removes unreferenced objects from the repository. Before executing this command, make sure that no editors are running and no one is using the system.

The syntax for the **clean** command is:

```
cle[an] [<options>]
```

Table 2: clean Command Options

Option	Description
-O	If invoked with the -O option followed by an object identifier, only the object referenced by that identifier is removed.
-n	If invoked with the -n option, the clean command removes the specified unreferenced objects even if they have annotations.
-v	Displays a detailed status (verbose) of each object being removed.
-f	Perform operation without asking for confirmation.

The following command removes from the current system any unreferenced objects that do not have annotations:

```
-> clean
```

See also the description of the **Delete Unreferenced Objects in Current System Repository** Desktop command in “Deleting Unreferenced Objects” on page 3-17.

copy

The **copy** command copies files from one system to another and updates the repository accordingly.

The syntax for the **copy** command is:

```
c[opy] | C[opy] [<options>] <from_diagram_name>
               <to_diagram_name>
```

Table 3: copy Command Options

Option	Description
-e	Editor. Set the current editor for this operation. This is the name of an editor in your environment. For a list of valid editors, execute the list command in the environment subsystem.
-T	Filetype. Specify a filetype.
-v	Perform operation in verbose mode.
-f	Perform operation without asking for confirmation.
-n	Perform operation on annotations.

It requires two arguments, the full name of the source diagram and the full name of the destination diagram. If you do not specify the source and the destination, StP Utility prompts for them.

The **copy** command supports an **-n** option which, if specified, copies all the diagram's annotations along with the diagram. If the **-n** option is not used, the annotations are not copied.

The following command copies diagram *Email*, with all its annotations, from system *sys1* of project *p3*, to diagram *Email* in system *sys4* of project *p6*. The user is prompted for source and destination arguments.

```
-> copy -n
From project:system:diagram: p3:sys1:Email
To project:system:diagram: p6:sys4:Email
```

delete

The **delete** command deletes from the system any diagrams, tables, or annotations with the specified name. It also deletes the associated ASCII files.

The syntax for the delete command is:

```
d[delete] [<options>] <pattern>
```

Table 4: delete Command Options

Option	Description
-e	Editor. Set the current editor for this operation. This is the name of an editor in your environment. For a list of valid editors, execute the list command in the environment subsystem.
-T	Filetype. Specify a filetype.
-v	Perform operation in verbose mode.
-f	Perform operation without asking for confirmation.
-n	Perform operation on annotations.

Before invoking the **delete** command, set the editor as described in “editor” on page 4-9.

The following command deletes the *Email* diagram in the *sys1* system of the *mysystems* project and all the files associated with that diagram:

```
-> delete mysystems:sys1:Email
```

To delete all the diagrams and files from a system, use the **Destroy System** Desktop command or the **sys_destroy** QRL function, described in “Destroying a System” on page 3-23. To delete the repository but keep the files, so that the repository can be regenerated, use the **Destroy System Repository** Desktop command or the **sys_destroy_rep** QRL function, described in “Destroying a System Repository” on page 3-23.

editor

The **editor** command sets or examines the current editor.

The syntax for **editor** is:

```
e[ditor] [<StP_editor_name>]
```

If invoked without an argument, the **editor** command reports the current editor. If invoked with a valid editor argument, it sets the current editor to the editor specified by its argument.

You can list the valid editors by entering the environment subsystem of StP Utility and executing the **ls** command. The fourth column in the results displays the editor names.

The following command sets the current editor to the Chen Diagram Editor:

```
-> editor uclassd
```

environment

The **environment** command activates the StP Utility environment subsystem.

The syntax for **environment** is:

```
env[ironment]
```

For more information, see “Accessing Subsystem Command Groups” on page 4-4.

files

The **files** command lists the files in the current system directory.

The syntax for **files** is:

```
f[files] [<options>]
```

Table 5: editor Command Options

Option	Description
-e	Editor. Set the current editor for this operation. This is the name of an editor in your environment. For a list of valid editors, execute the list command in the environment subsystem.
-T	Filetype. Specify a filetype.
-v	Perform operation in verbose mode.

With no argument, **files** lists all the files in the directory. With an argument, it lists all the files in the system directory that match the pattern described by its argument.

To specify one or more specific files, follow the command name with a pattern describing the names of the files that are to be listed. See “Patterns” on page 4-5 for information about patterns.

To list all the files in the system directory:

```
-> files
```

To list all the system files with names that begin with the character “f” followed by a single character, followed by “e_files”:

```
-> files f?e_files
```

To list all the system files with names that contain the substring “tr”:

```
-> files *tr*
```

To list the file objects as well as the files, use the **ls** command.

filetype

The **filetype** command sets or examines the current file type.

The syntax for **filetype** is:

```
filetype [<options>] [<type>]
```

Table 6: filetype Command Options

Option	Description
-v	Perform operation in verbose mode.
-h	Print help about command.

If invoked without an argument, the **filetype** command reports the current file type. If invoked with a valid file type argument, it sets the current file type to the type specified by its argument.

You can list the valid file types by entering the environment subsystem of StP Utility and executing the ls command. The second column in the results displays the file type names.

The following command sets the current file type to the UML Class Table file type:

```
-> filetype UmlClassTable
```

fix content

The **fix content** command makes sure the system is set up with a valid StP Message Router (msgd) context.

The syntax for **fix content** is:

```
fix[content]
```

force

The **force** command sets or examines the force mode value. When on, Force mode disables confirmation messages that appear after you copy or delete files.

The syntax for **force** is:

```
F[orce] | fo[rce] [on | off]
```

For more information, see “Setting StP Utility Modes” on page 4-2.

help

The **help** command lists all the commands that can be invoked from the current StP Utility subsystem.

The syntax for help is:

```
h[elp] [<command>]
```

With a valid command name as an argument, the **help** command displays options for that command in the current subsystem.

You can also type ? to invoke the **help** command. For help on a particular command, pass the command name as an argument.

To list all commands that can be invoked from the current subsystem:

```
-> help
```

To display information about the **files** command in the current subsystem:

```
-> help files
```

interactive

The **interactive** command sets or examines interactive mode.

The syntax for **interactive** is:

```
I[nteractive] | int[eractive] [on | off]
```

For more information, see “Setting StP Utility Modes” on page 4-2.

lock

The **lock** command activates the StP Utility locking subsystem.

The syntax for **lock** is:

```
loc[k]
```

For more information, see “Accessing Subsystem Command Groups” on page 4-4.

ls

The **ls** command lists both file objects in the repository and their associated ASCII files in the current system directory.

The syntax for **ls** is:

```
l[s] [<options>] <pattern>
```

Table 7: ls Command Options

Option	Description
-e	Editor. Set the current editor for this operation. This is the name of an editor in your environment. For a list of valid editors, execute the list command in the environment subsystem.
-T	Filetype. Specify a filetype.
-v	Perform operation in verbose mode.

With no argument, **ls** lists all the file objects and associated files in the system. With an argument, it lists all the file objects and associated files that match the pattern described by its argument.

To specify one or more specific file objects, follow the command name with a pattern specifying the names of the files to be listed. See “Patterns” on page 4-5 for information about patterns.

To list all the file objects and files in the system with names that begin with the characters “al”:

```
-> ls al*
```

To list all the file objects and their associated files that have names beginning with the character “s” followed by a character between “b” and “f” inclusive, followed by the characters “es”:

```
-> ls s[b-f]es
```

To list only the system files without the file objects, use the **files** command.

project

The **project** command sets or examines the current project directory.

The syntax for project is:

```
p[roject] [<projdir>]
```

If invoked without an argument, the **project** command displays the pathname of the current project. If invoked with an argument, it sets the current project to the directory specified by its argument.

The following command sets the current project to *C:\arch\evo\ps*:

```
-> project C:\arch\evo\ps
```

quiet

The **quiet** command sets or examines the quiet mode value.

The syntax for **quiet** is:

```
Q[uiet] | quie[t] [on | off]
```

For more information, see “Setting StP Utility Modes” on page 4-2.

quit

The **quit** command, invoked from the Main level, exits StP Utility.

The syntax for quit is:

q[uit]

rename

The **rename** command changes the name of a diagram, table, or annotation within a system.

The syntax for rename is:

r[ename] [<options>] <object_name> <object_newname>

Table 8: rename Command Options

Option	Description
-e	Editor. Set the current editor for this operation. This is the name of an editor in your environment. For a list of valid editors, execute the list command in the environment subsystem.
-T	Filetype. Specify a filetype.
-v	Perform operation in verbose mode.
-f	Perform operation without asking for confirmation.

The first argument is the diagram's current name. The second argument is its new name.

The following command changes the name of the *Email* diagram to *Mail*:

```
-> rename Email Mail
```

schema

The **schema** command creates a schema in the ODBC database with the same name as the system.

The syntax for **schema** is:

sc[hema]

status

The **status** command reports status settings for StP Utility. This information includes the current project and system, whether the database is enabled and open, the current editor, and the current mode settings.

The syntax for **status** is:

```
st[atus] [<option>]
```

The **status** command has one option, **-v** (perform operation in verbose mode).

```
->
status
StP-Utility Status
Project Dir: C:/StP/Examples/
System      : atm_sys
Database enabled, closed
Editor      : stputil
Verbose     : on
Quiet       : off
Interactive: on
Force       : off
->
```

system

The **system** command sets or examines the current system directory.

The syntax for **system** is:

```
s[ystem] [<system_name>]
```

If invoked without an argument, the **system** command displays the name of the current system.

If invoked with an argument, it sets the current system to the directory specified by its argument. The argument must specify an existing system under the current project directory. If necessary, use the **project** command to set the current project.

The following command sets the current system to *sys1*:

```
-> system sys1
```

undef

The **undef** command displays the names of all system files that have no corresponding repository file object. It also creates a repository entry for each missing file object, but does not update the repository with the file's contents. To update the repository with the missing file's contents, you can load the file into an appropriate StP editor and save it, or run the **sys_update** function in a QRL script. For more information, see *Query and Reporting System*.

The syntax for undef is:

```
u[ndef] [<options>]
```

Table 9: undef Command Options

Option	Description
-e	Editor. Set the current editor for this operation. This is the name of an editor in your environment. For a list of valid editors, execute the list command in the environment subsystem.
-T	Filetype. Specify a filetype.
-v	Perform operation in verbose mode.
-f	Perform operation without asking for confirmation.

You can use the **Recover System Repository** command to generate repository objects from system files. See “Rebuilding the Repository from System Files” on page 3-27 for information about this command.

verbose

The **verbose** command sets or examines the quiet/verbose mode value.

The syntax for **verbose** is:

```
v[erbose] | v[erbose] [on | off]
```

For more information, see “Setting StP Utility Modes” on page 4-2.

Environment Subsystem

To enter the StP Utility environment subsystem, type **environment** or **env** from the Main level.

This section describes the commands that are specific to the StP Utility environment subsystem and that are common to all StP products. They are listed in alphabetical order by full command name. See “StP Utility Main Level Commands” on page 4-5 for information about commands such as **help**, **quit**, and **verbose** that are available from all StP subsystems. For information about commands in this subsystem that are specific to a particular StP product, invoke **help** with the command as its argument in StP Utility.

check

The **check** command checks that all the files in the system directories are valid StP files. It also checks that the files have valid application types that allow them to be mapped into the repository. It then reports any discrepancies found.

The syntax for **check** is:

```
c[heck]
```

list

The **list** command displays information about the file types defined for the StP editors in the *editor.rules* file for the current system.

The syntax for **list** is:

```
l[ist]
```

update

The **update** command adds missing directories to the system. It is used to update the system directory structure when a new system is created or when new editors are added to the product.

The syntax for update is:

```
u[pdate]
```

Locking Subsystem

To enter the StP Utility locking subsystem, type **lock** or **loc** at the Main level. The locking subsystem of StP Utility provides command-line versions of:

- System-level locking commands that are accessible from the Desktop Commands menu for the StP Utilities tool
- Object-level locking commands that are accessible from the StP editors

For a list of locking commands and their descriptions, type **help** within this subsystem. For information on StP Utility command syntax, invoke the **help** command on specific locking commands.

See *Fundamentals of StP* for a discussion of locking, as well as for information about the equivalent commands available from the Desktop and the StP editors.

5 StP/DOORS Integration

This chapter describes the StP/DOORS integration, which links StP with DOORS, a tool for managing system requirements.

Topics covered are as follows:

- “Overview of StP/DOORS” on page 5-1
- “How DOORS Stores StP Objects” on page 5-2
- “Integrating StP with DOORS” on page 5-3
- “Starting StP/DOORS” on page 5-4
- “Navigating from StP to DOORS” on page 5-4
- “Navigating from DOORS to StP” on page 5-5
- “Updating Renamed Objects Exported to DOORS” on page 5-6

Overview of StP/DOORS

DOORS, developed by Quality Systems & Software (QSS), is designed to capture, manage, link, trace, generate, and analyze textual and graphical information to ensure a product’s compliance with system requirements and specifications.

The StP/DOORS integration allows you to:

- Export objects from StP to DOORS
 - Navigate from StP to the exported objects in DOORS
 - Allocate requirements to the exported objects using DOORS
 - Navigate from the exported objects in DOORS back to StP
-

The integration of StP and DOORS ensures that both requirements and application development models systematically reflect changes made throughout the development cycle.

The StP/DOORS integration is available from all StP editors and products. All objects that support allocated requirements can be exported to DOORS.

For detailed information on DOORS, refer to the DOORS product documentation.

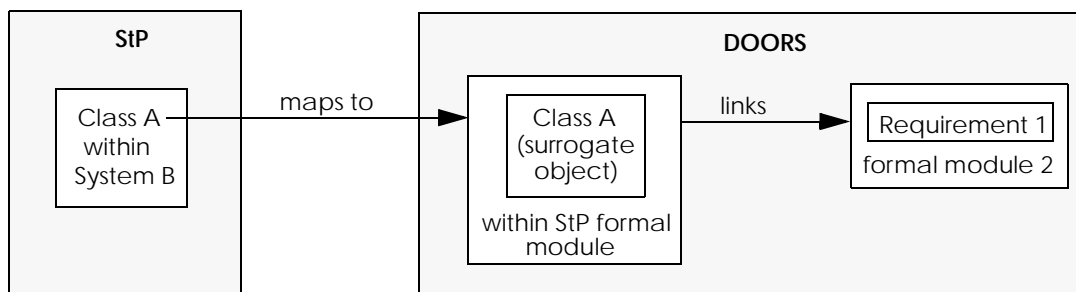
How DOORS Stores StP Objects

DOORS stores StP objects in formal modules. All objects from the same StP system are stored in a single formal module. DOORS manages all links from the formal module to requirements.

When you export an StP object to DOORS, it is stored in DOORS as a surrogate object. The surrogate object represents the actual StP object in DOORS. Figure 1 illustrates how StP objects are mapped in DOORS:

1. Class A is exported to DOORS.
2. StP/DOORS creates a surrogate object within a formal module to act as a representative for Class A.
3. The surrogate object Class A maps to the appropriate DOORS objects, which have specified requirements.

Figure 1: How StP objects map to DOORS



After you have exported an StP object to DOORS, you can perform any valid DOORS operation involving the exported object, such as linking the object with requirements, editing the object's attributes, and so on.

Integrating StP with DOORS

Both StP and DOORS must be installed before you can enable the StP/DOORS integration. To enable the integration, you need to do the following:

- Set the appropriate environment variables.
- Move several StP files to the DOORS installation directory.

Setting the User's Environment

Set the following variables for each Windows NT StP/DOORS user:

- *use_doors_integration* ToolInfo variable to "Yes":
`use_doors_integration=Yes`
"Yes" enables DOORS; "No" disables DOORS.
- PATH: Make sure that StP is included in the PATH environment variable.

Note: See "User Environments" on page 2-8 for more information on setting up the user's environment.

Moving StP Files to DOORS

Move the following directory and file content from the *StP\templates\ct\doors\addins* directory to the DOORS installation directory:

- *stp*: The *stp* directory contains the StP customization.
Copy the entire directory to the *%DOORSHOME%\lib\dxl\addins* directory.

The following is a command line example:

```
xcopy stp\*. * %DOORSHOME%\lib\dxl\addins\ /s /i
```

- *startup.dxl*: Add the contents of this file to %DOORSHOME%\lib\dxl\startup.dxl. If this file does not exist, create it.
- *addins.idx*: Add the contents of this file to %DOORSHOME%\lib\dxl\addins\addins.idx. If this file does not exist, create it.

After Integrating StP and DOORS

After you integrate StP and DOORS, use the following restrictions:

- Specify one project directory in the ToolInfo file.
- Only use one StP product (for example, StP/UML) with DOORS. If you try exporting an object from one StP product and then navigate to it from DOORS while running another StP product, the navigation will fail.

Starting StP/DOORS

After you install and integrate StP and DOORS, each product will have an additional menu allowing you to navigate to the other product: the StP editors will have a **DOORS** menu under the **GoTo** menu, and DOORS will have an **StP** menu.

Navigating from StP to DOORS

You navigate from a selected object in an StP editor to DOORS by choosing the **DOORS** command, available from the editor's **GoTo** menu.

If the **DOORS** command does not appear, and StP/DOORS is enabled, the selected object does not support allocated requirements and cannot be exported to DOORS.

To navigate from an StP object to DOORS:

1. Make sure DOORS is running.
2. In DOORS, open the project you want to export the StP object to.
3. In StP, select an object (class, use case, process, and so on).
You cannot export a group of objects to DOORS. Only export one object at a time, and wait until the object has completely exported before exporting the next object.
4. From the **GoTo** menu, choose **DOORS**.
If this is the first time you are exporting an StP object to the selected DOORS project, DOORS will prompt you to enter a name for the module.
5. Enter a name (DOORS will offer the name of the current StP system as the default) and click Create.

After you have exported an StP object to DOORS, you can perform any valid DOORS operation on the surrogate object, such as linking the object with requirements, editing the object's attributes, and so on.

Navigating from DOORS to StP

After you export an StP object to DOORS, DOORS lets you allocate requirements to the StP surrogate object. DOORS groups all StP objects from the same StP system into their own formal module. In other words, there is one formal module for every StP system being used with DOORS.

To navigate from DOORS to StP:

1. In DOORS, select an object (class, use case, process, and so on).
2. Choose **Navigate to StP** from the StP menu.
If StP has more than one object with the name of the selected object, an Object Selector dialog appears, listing the object locations.
3. Select the object location you need and click **OK** or **Apply**.

After you navigate from DOORS to StP, the Repository Browser appears, which enables the next navigation. An StP/DOORS navigation requires an StP editor because the navigation uses QRL statements. The

Repository Browser appears because it accepts QRL statements and is not StP product specific.

Updating Renamed Objects Exported to DOORS

Whenever you rename an object by using the **Rename Object Systemwide** command, you need to update the object's references in DOORS. To do so, choose **Synchronize DOORS with StP** from the **Tools** menu.

For more information on the **Rename Object Systemwide** command, see the StP product-specific documentation.

A ToolInfo Variables

This appendix provides a list of all StP ToolInfo variables.

For a detailed description of how ToolInfo variables work, see “User Environments” on page 2-8.

ToolInfo Variable Descriptions

Table 10 describes the StP ToolInfo variables.

Table 10: ToolInfo Variables

ToolInfo Variable	Type	Default Value	Description	Used in
dateformat	String	%M/%D/%Y	Specifies format used when printing the date.	StP Utility
do_not_use_dbsetversion	Boolean	False	If True, sets the Sybase Open Client Library to release 4.6. If False, sets Open Client Library to release 10.0.x.	all
filter_path	String		Specifies an alternate location to <i>stp_file_path</i> for looking for the location of filters, and is mainly used for debugging.	all

ToolInfo Variables

Table 10: ToolInfo Variables (Continued)

ToolInfo Variable	Type	Default Value	Description	Used in
includir	String		Adds a directory to the list of directories to search for files included using the #include directive.	Query and Reporting Processor
lock_auto_timeout	Integer	0	Specifies the number of minutes to keep an automatic lock; value of 0 means to keep the lock until the editing session finishes.	all
mach_name_sep	Char	@ (at sign)	Specifies the character separator to be used between the user name and machine name.	all
monthnames	Identifier list	January February March April May June July August September October November December	Specifies the names of the months in order.	StP Utility
msg_file	String		Specifies the location of the message file.	all
msgd_host	String		Specifies the host machine to be used for the client machine's message router. For more information, see Chapter 3, "Administering StP."	all
msgd_port	String	Specified by platform	Specifies the TCP/IP port number to be used by the message router service.	all

ToolInfo Variable Descriptions

Table 10: ToolInfo Variables (Continued)

ToolInfo Variable	Type	Default Value	Description	Used in
msjet_password	String		For Microsoft Jet, specifies the password for any user who is not a system administrator.	all
msjet_admin_password	String		Specifies the password for the Microsoft Jet system administrator.	all
oms_dbuser	String		Assigns an StP user name that differs from the user's login.	all
oms_debug_sql	Boolean		For Sybase, prints SQL either to the window that spawned StP or to the StP message window.	all
pick_fully_inside	Boolean		Does not allow partial overlap when picking symbols.	diagram editor
print_mif_importable	Boolean	False	If True, allows MIF (Maker Interchange Format) files to be imported by reference to FrameMaker files. Allows you to import the following types of files: <ul style="list-style-type: none"> • A single diagram on a page • More than one diagram, each on a single page • A single diagram printed over multiple pages ("multipage") 	diagram editor printing
product	String		Specifies the name of an StP product that will be used as a prefix to determine which <i>stp_file_path</i> ToolInfo variable to use, for example, <i>uml_stp_file_path</i> .	all

Table 10: ToolInfo Variables (Continued)

ToolInfo Variable	Type	Default Value	Description	Used in
projdir	String		Specifies the location of the project directory. (This variable is overridden by system repository functions' projdir parameter. See <i>Query and Reporting System</i> for details.)	all
rte_Cascade-Requirement-Allocation	Boolean	True	If True, permits the cascading of phase allocation, which means the requirement table will allocate to the current phase and all successive phases. If False, allocates a requirement only to the single phase the requirement table is set to.	requirements table editor
save_scale_info	Boolean		Controls whether to save scale, alignment, and pan information for a diagram when loading another.	diagram editor
scope_comp_separator	String	; (semi-colon)	Character for OMS to use to separate the name, AppType, and signature components of a scope attribute	all
scope_separator_char	String	. (period)	Character for OMS to use to separate scopes from each other	all
scriptman_ascii_view	String	notepad \${asciifile} &	Specifies command to use to view ASCII output from Script Manager.	Script Manager

ToolInfo Variable Descriptions

Table 10: ToolInfo Variables (Continued)

ToolInfo Variable	Type	Default Value	Description	Used in
scriptman_external_editor	String	notepad \${externalpath} &	Specifies editor used in Script Manager for editing external variables.	Script Manager
scriptman_product_externals_location	String	Optional. No default value.	Provides the directory for product externals.	Script Manager
scriptman_script_editor	String	notepad \${scriptpath} &	Specifies editor used in Script Manager for editing scripts.	Script Manager
scriptman_script_process	String	qrp -p \${projdir} -s \${system} \${scriptpath} \${targetargs}	Specifies process that executes QRL scripts from Script Manager.	Script Manager
scriptman_system_externals_location	String	Optional. No default value.	Provides the directory for system externals.	Script Manager
scriptman_ttywait	String	"ttywait_ "	Specifies a program to run inside the script execution window after the script has finished. Used to allow the execution window to remain open, so the user can see results.	Script Manager
scriptman_user_externals_location	String	Optional. No default value.	Provides the directory for user externals.	Script Manager
scriptman_user_scripts_location	String	Optional. No default value.	Provides location of user scripts.	Script Manager
scroll_line	Integer	40	Sets scroll line length.	all
<server>_admin_password	String		Specifies the password for the Sybase system administrator.	all

Table 10: ToolInfo Variables (Continued)

ToolInfo Variable	Type	Default Value	Description	Used in
<server>_password	String		For Sybase, specifies the password for the default Adaptive Server or SQL Server for any user who is not a system administrator.	all
source_editor	String		Specifies a non-default editor for source code navigations. The StP-installed ToolInfo file provides a commented out value set to MSDev.	all
stp_file_path	String		Specifies a set of paths used by StP tools to find information that resides in the template directory. The order of the directories specified is the order in which the information is found.	all
stp_ps_print	String	type \${psfile} > \${psprinter}	Specifies the default PostScript print command for printing from a diagram editor or the Desktop, even if the Print Server's default printer is set differently.	StP Desktop, diagram editor
stpdt_termwin	String	wstart sh	Specifies the terminal window to use for the StP Desktop.	StP Desktop
syscreate_device	String	default	Specifies the Sybase device to store the Adaptive Server or SQL Server repository data.	all
syscreate_log_device	String	default	For Sybase, specifies the device on which to store the repository's transaction log.	all

ToolInfo Variable Descriptions

Table 10: ToolInfo Variables (Continued)

ToolInfo Variable	Type	Default Value	Description	Used in
syscreate_log_size	Integer	2	For Sybase, sets the size of the repository data, in megabytes.	all
syscreate_size	Integer	6	For Sybase, sets the size of the repository, in megabytes	all
system	String		Specifies the name of the system being used. (This variable is overridden by the system repository functions' system parameter. See <i>Query and Reporting System</i> for details.)	all
time_format	String	%M/%h/%d/%m/%y	Used in stputil to overwrite the system-wide default time print format.	StP Utility
timeformat	String	%h:%.2m:%.2s	Specifies the format used when printing the time.	StP Utility
tmp_dir	String	/tmp/	Gives the location of temporary directory used by StP products.	all
ToolInfo	String		Gives the location of the ToolInfo file.	all
use_doors_integration	Boolean		If using the DOORS integration product, enables its integration with StP: "Yes" enables; "No" disables.	DOORS integration
use_offpages	Boolean		If True, uses offpage symbols in Data Flow Editor (DFE) decompositions.	diagram editor

ToolInfo Variables

Table 10: ToolInfo Variables (Continued)

ToolInfo Variable	Type	Default Value	Description	Used in
weekdays	Identifier list	Sunday Monday Tuesday Wednesday Thursday Friday Saturday	Specifies the names of the days of the week in order.	StP Utility

B Command Reference

This appendix provides reference pages for the following commands:

- `aupdate` (under “Update Commands”)
 - `ccparser`
 - `comments`
 - `dprint`
 - `dupdate` (under “Update Commands”)
 - `gde`
 - `gte`
 - `msgd`
 - `msgdiag`
 - `nav_source`
 - `oae`
 - `qrp`
 - `re_db_check`
 - `re_db_locks`
 - `re_gen_defines`
 - `re_make`
 - `repquery`
 - `repsync`
 - `scriptman`
 - `smdb2uml` (under “smbd2* Commands”)
 - `stp`
 - `stpem`
 - `stputil`
 - `toolloc`
 - `tprint`
 - `tupdate` (under “Update Commands”)
 - `xrb_search`
-

ccparser

Function	Parses C++ source code so that the smdb2uml command can read it.
Syntax	<code>ccparser [-p projdir] [-s system] [-nopp] [-i] [-man] [-MSExt] C++_file [C++_file...] -class classname</code>
Options	<p>Options are as follows:</p> <ul style="list-style-type: none">• <code>-p projdir</code>—Uses the specified <i>projdir</i> directory as the current project directory (for Windows NT, supply a drive designation). The <i>projdir</i> variable specified with <code>-p</code> overrides the default directory specified by the <i>projdir</i> ToolInfo variable.• <code>-s system</code>—Uses the named system as the current system name. This name overrides the default system name given by the <i>system</i> ToolInfo variable. The system name is appended to the project directory to specify the location of directories for the system.• <code>-i</code>—Incremental operation. This option prevents the parser from destroying information previously extracted from C++ source files. It parses only files that failed to parse or have never been parsed in previous runs, or that have been modified since they were last parsed.• <code>-nopp</code>—Disables preprocessing.• <code>-man</code>—Indicates that you want to use the filenames and settings that were created by manually running re_make.• <code>-MSExt</code>—Enables Microsoft extensions.• <code>-class <classname></code>—Runs the parser on the specified class only and can only be used in conjunction with the <code>-i</code> option. If the .C files for the class have been modified since last parsed, they are reparsed incrementally, and incremental comment gathering is run on the class. The class must exist in the semantic model.
Comments	<p>You can specify multiple source code files with ccparser. Usually parsing also involves full preprocessing of any header files, although preprocessing is optional (see the <code>-nopp</code> option, described above).</p> <p>Because the ccparser command needs many parameters in order to locate, preprocess, and parse source files, the parameters are read in from associated files. All the files are set up automatically when the Parse Source Code property sheet is run from the Class Capture tool in the Access Repository category on the StP Desktop.</p> <p>If you do not want to use the StP Desktop to set up these files, you can enter the names of the source files you want to parse into <code><project>/<system>/revc_files/Files.cc</code> file. Enter filenames one per line. You should add search paths for source files and user define header files</p>

(those included with `#include ""` notation) in the *UserIncludes.cc* file in the same directory. Store system include file paths in the *SystemIncludes.cc* file, in the same directory and the same format, one per line. You can specify any preprocessor `#defines` by entering them (one per line) into the *Defines.cc* file. This also resides in the `<project>\<system>\revc_files\` directory.

Alternatively, you can set up all the files automatically from a Makefile, by running the makefile reading utility, **re_make**. If you use **re_make** to create these files for you, you need to run **ccparser** with the `-man` option.

See Also**re_make, smdb2uml**

comments

Function	Runs the comment gathering-phase for information extracted from source code and puts the results into the specified semantic model.
Syntax	<code>comments [-p projdir] -s system -f filename [-i] [-verbatim 0 1 2] [-annot_len nn] [-start_delimiter token] [-end_delimiter token] [-version] [-help]</code>
Options	<p>Options are as follows:</p> <ul style="list-style-type: none"> • <code>-p projdir</code>—Uses the specified <i>projdir</i> directory as the current project directory (in Windows NT, requires a drive designation). The <i>projdir</i> variable specified with <code>-p</code> overrides the default directory specified by the <i>projdir</i> ToolInfo variable. • <code>-s system</code>—Uses the named system as the current system name. This name overrides the default system name given by the <i>system</i> ToolInfo variable. The system name is appended to the project directory to specify the location of directories for the system. • <code>-f filename</code>—Specifies the name of the file that holds the template of where to look for comments. The template file must reside in the <i>project/system/revc_files</i> directory. The comment template file contains the following: <ul style="list-style-type: none"> • <code>function_above</code>—Searches the function definitions above. 0 disables and 1 enables. The default is 1. • <code>function_below</code>—Searches the function definitions below. 0 disables and 1 enables. The default is 0. • <code>function_gap</code>—Sets the number of non-comment lines allowed between a definition and a comment. The default is 2. • <code>structure_above</code>—Searches the structure definitions above. 0 enables and 1 enables. The default is 1. • <code>structure_below</code>—Searches below the structure definitions. 0 enables and 1 enables. The default is 0. • <code>structure_side</code>—Searches to the right of structure definitions. 0 enables and 1 enables. The default is 0. • <code>structure_gap</code>—Sets the function gap. The default is 2. • <code>global_above</code>—Searches above the global definitions. 0 disables and 1 enables. The default is 1. • <code>global_side</code>—Searches to the right of the global definitions. 0 disables and 1 enables. The default is 0.

- `global_gap`—Sets the function gap. The default is 1.
- `startdelimiter`—Specifies the token that indicates the start of legitimate comments. The default is NULL.
- `enddelimiter`—Specifies the token that indicates the end of legitimate comments. The default is NULL.
- `-i`—Incremental mode. When searching for source code comments, this option considers only items for which comments have not been gathered in previous executions.
- `-verbatim`—Sets formatting as follows:
 - `0`—Reformats the extracted annotations into lines, with no line being longer than specified `-annot_len` length.
 - `1`—Preserves new lines, but removes leading white space.
 - `2`—Produces source code comments with exactly the same indentation and line breaks they have in the source code.
- `-annot_len nn`—Used in conjunction with `-verbatim 0`, reformats annotations of source code comments to fit the specified length. The default for `nn` is 64.
- `-start_delimiter token`—Allows you to specify a string that indicates the start of a legitimate comment. Useful if comments are formatted into official portions, and less visible parts, for example:

```
/* the general comment for this function begins here,
COMMENT : this is function foo()
ACTION  : a test harness
*/
```

If you set `start_delimiter` to “COMMENT,” everything before it is ignored. By default, `start_delimiter` is not set, causing the entire source comment to be turned into an annotation.

- `-end_delimiter token`—Acts like `start_delimiter`, but indicates if there is a terminating token for the formal part of the comment.
- `-version`—Displays the version of comments. It does not perform the normal action of the program.
- `-help`—Displays the syntax of the **comments** command.

Comments

Before you use the **comments** command, make sure that the semantic model must have already been populated with information by running the parser (see **ccparser** for StP/UML). **comments** takes many optional parameters that specify the search templates used for finding comments in the source code that has been parsed.

Diagnostic messages are produced, indicating what phase of comment gathering is being executed. At the end, percentages of comments

Command Reference

gathered for functions, structures, and global identifiers are displayed. The comment gathering program attempts to replace any offensive words with groupings of punctuation characters.

See Also **ccparser**

dprint

Function	Generates FrameMaker, Interleaf, RTF, Microsoft Word, and PostScript files from Software through Pictures (StP) diagrams.
Syntax	<code>dprint -ed editor diagram1 [diagram2...] [-p projdir] [-s system] [-target mif leaf rtf msword_printer post p rinter post_level2 printer_level2] [-out outputfile] [-printer printer_name] [-orientation landscape portrait] [-rules_print print_rules_file] [-print_setting setting_name] [-version]</code>
Options	<p>Options are as follows:</p> <ul style="list-style-type: none">• <code>-ed editor</code>—Specifies the StP editor used to create the diagram. This option is required.• <code>diagram</code>—Specifies which diagrams to format. This must be a diagram created with editor. Specify at least one diagram.• <code>-p projdir</code>—Uses the specified projdir directory as the current project directory (in Window NT, it requires a drive designation). The projdir variable specified with <code>-p</code> overrides the default directory specified by the <i>projdir</i> ToolInfo variable.• <code>-s system</code>—Uses the named system as the current system name. This name overrides the default system name given by the <i>system</i> ToolInfo variable. The system name is appended to the project directory to specify the location of directories for the system.• <code>-target mif leaf rtf msword_printer post printer post_level2 printer_level2</code>—Specifies whether the output is to be of the following:<ul style="list-style-type: none">• FrameMaker MIF (mif)• Interleaf-ASCII (leaf)• Rich Text Format (rtf)• Microsoft Word (msword_printer) on Windows NT only• Encapsulated PostScript to a file (post)• Encapsulated PostScript directly to a printer (printer)• Level 2 PostScript to a file (post_level2)• Level 2 PostScript directly to a printer (printer_level2).Without this option, dprint uses the target specified by the current print setting.• <code>-out outputfile</code>—Specifies the name of the output file. If you specify "-" for outputfile, the file is generated to stdout. If you provide no output filename, dprint generates to the file specified by the current print setting.

If `outputfile` is a filename without a complete path, then **dprint** generates the output file in the directory in which it is executed.

- `-printer printer_name`—Specifies the name of the printer to use when `printer` or `printer_level2` is used as the value for the `-target` option. Otherwise, it uses the default printer.
- `-orientation landscape | portrait`—Specifies the orientation of the diagram on the page.

Note that this option is normally specified in the print setting supplied to **dprint** in a rules file as the attribute `PageOrientation`. This option (`-orientation`) allows you to override the value in the rules file. The command-line option `landscape` yields a `PageOrientation` of `LandscapeFit`, while `portrait` yields `PortraitFit`.

- `-rules_print print_rules_file`—Specifies a customized print rules file that you want to use with diagram.

See the `-printsetting` option discussion for what happens when you use the two options together.

When you use this option without the `-print_setting` option, **dprint** looks for (in order of precedence):

1. A `DiagramPrintSetting` rule named `editor` in *print_rules_file*
2. A `DiagramPrintSetting` rule named “`editor`” in *print_rules_file*

- `-print_setting setting_name`—Specifies a print setting associated with diagram to be used to format the output.

If you use both the `-print_setting` and the `-rules_print` options, **dprint** looks for a `DiagramPrintSetting` rule named `setting_name` in *print_rules_file*.

If you use this option without the `-rules_print` option, **dprint** looks for (in order of precedence):

1. A `DiagramPrintSetting` rule named `setting_name` in the named setting file associated with diagram
2. A `DiagramPrintSetting` rule named `setting_name` in the *print_default.rules* file

If you use neither the `-print_setting` nor the `-rules_print` option, **dprint** looks for (in order of precedence):

1. A `DiagramPrintSetting` rule named “`editor`” in the named settings file associated with diagram
2. A `DiagramPrintSetting` rule named `editor` in the *print_default.rules* file

3. A DiagramPrintSetting rule named “editor” in the *print_default.rules* file

If **dprint** is printing more than one diagram, the named setting files associated with the diagrams are not used.

- -version—Displays version information for the StP Core and, if appropriate, product components. It does not perform the normal action of the program.

Comments

The **dprint** command generates formatted versions of Software through Pictures (StP) diagrams into one of several target formats:

- FrameMaker MIF
- Interleaf-ASCII
- Rich Text Format (RTF, used by PC-based publishing tools)
- Microsoft Word (Windows NT only)
- Encapsulated PostScript sent to a file or to a PostScript printer
- Level 2 PostScript sent to a file or to a PostScript printer.

Level 2 PostScript is required for those printers that need the PostScript operator `setpagedevice` to cut the paper to the proper size. For example, the HP 650 DesignJet printer/plotter, which uses rolls of paper rather than sheets, requires this command.

dprint uses print settings to format diagrams. A print setting for a diagram is defined with a DiagramPrintSetting rule. The DiagramPrintSetting rule’s attributes specify print formatting options, such as page size, margins, page orientation, and captions. Print settings are found in:

- The default print rules file (*print_default.rules*, located in the *templates/ct/rules* directory relative to the path specified by the `<stp_product>_stp_file_path` ToolInfo variable, where `<stp_product>` is determined by the *product* ToolInfo variable)
- A named setting file (a print rules file associated with a specific diagram, created using a diagram editor’s named Setting Dialog, which is accessible from the Print dialog box)
- A user-created customized print rules file, which can be passed to **dprint** on the command line

See *Fundamentals of StP* for information on named print settings.

The print rule in the *print_default.rules* file that provides default diagram print formatting options for dprint is the DiagramPrintSetting rule named “editor.” To use non-default values for a specific diagram, you can use a named print setting. To use non-default values for multiple diagrams, you normally use a customized rules file. Alternatively, you

can manually insert customized DiagramPrintSetting rules in the default print rules file.

For more detailed information, see the description under “Options” for `-rules_print` and `-print_setting`.

A note about printing display marks: The NoPrintDisplayMark rule, which suppresses the appearance of specific display marks on diagrams, occurs in the diagram editor’s rules file, not in the print rules file. The NoPrintDisplayMark rule holds for diagrams printed with **dprint**, **grp**, and those printed from the StP Desktop’s Print Diagram command. It does not hold for diagrams printed from a diagram editor, which are printed wysiwyg (what you see (on the displayed diagram) is what you get).

Files The **dprint** command affects the *print_default.rules* file, which defines the default settings for all StP printing, including printing diagrams and tables from the Desktop, QRL scripts, and **tpprint** and **dprint** commands.

Examples To generate a FrameMaker file for the UML diagram, MailSystem, using DiagramPrintSetting rule “uclassd” in the customized print rules file *Mysettings.rules*:

```
dprint -ed uclassd MailSystem -target mif -rules_print  
Mysettings.rules
```

See Also **gde**, **tpprint**, **grp**, **stp**

gde

Function	Invokes the Generic Diagram Editor (GDE).
Syntax	<code>gde -ed editor [diagram] [-p projdir] [-s system] [-C command]... [-version]</code>
Options	<p>Options are as follows:</p> <ul style="list-style-type: none">• <code>-ed editor</code>—Instantiates GDE as editor. The editor must be the name of an editor listed in the <i>editor.rules</i> file. The specified editor determines which <i>editor_rules</i> file is to drive this particular instantiation of GDE. This option is required.• <code>diagram</code>—Specifies which diagram to load at start-up time. This must be a diagram created with the editor specified by the <code>-ed</code> option. If you do not specify a diagram, the editor is started with a blank window.• <code>-p projdir</code>—Uses the specified <i>projdir</i> directory as the current project directory (in Windows NT, requires a drive designation). The <i>projdir</i> variable specified with <code>-p</code> overrides the default directory specified by the <i>projdir</i> ToolInfo variable.• <code>-s system</code>—Uses the named system as the current system name. This name overrides the default system name given by the <i>system</i> ToolInfo variable. The system name is appended to the project directory to specify the location of directories for the system.• <code>-C command ...</code>—Executes the specified StP built-in command at GDE start-up time.• <code>-version</code>—Displays version information for the StP Core and, if appropriate, product components. It does not perform the normal action of the program.
Comments	<p>The GDE the basis for all of the StP diagram editors. The GDE editor allows you to draw and manipulate node symbols and link symbols. A node symbol is usually a geometric shape of some sort, such as a circle or rectangle. A link symbol is usually shown as a straight or curved line between two node symbols. In addition, GDE uses the concept of a context symbol, which is a symbol that can be attached to a link symbol in order to give the link additional meaning.</p> <p>The particular set of node, link, and context symbols available in any given instantiation of GDE is determined by a rules file. You must invoke the gde command with the <code>-ed</code> parameter to specify the rules file to be used to drive this instantiation of GDE.</p> <p>In addition to the set of legal symbols, the rules file also contains other important information, such as:</p> <ul style="list-style-type: none">• Which symbols can have labels

- How to navigate from one symbol to another
- What menus should the editor contain and what commands should be available within those menus
- How the information contained in the diagram is to be stored in the OMS repository

The GDE uses a select-then-operate user interface. You insert symbols by selecting them from the symbol palette and inserting them in the drawing area. Once in the drawing area, symbols can be selected and manipulated with a host of editing commands.

Files

The **gde** command affects the following files:

- *rules/editor.rules*—Rules file that describes the available StP editors. A subset of these editors are diagram editors. The rest are table editors, or miscellaneous editors, such as the Object Annotation Editor (OAE). This file is searched for in the path specified with the *stp_file_path* ToolInfo variable.
- *rules/editor_rules*—Rules file that drives a particular instantiation of the GDE. This file is searched for in the path specified with the *stp_file_path* ToolInfo variable.

Examples

To instantiate GDE as the Bachman editor as a background process:

```
gde -ed bach &
```

To instantiate GDE as the Object Model Editor on the diagram DevelopmentProcess in the system Mysystem:

```
gde -ed ome DevelopmentProcess -s Mysystem
```

See Also

gte, **stp**

gte

Function	Invokes the Generic Table Editor (GTE).
Syntax	<code>gte -ed editor [table] [-p projdir] [-s system] [-C command]... [-version]</code>
Options	<p>Options are as follows:</p> <ul style="list-style-type: none">• <code>-ed editor</code>—Instantiates GTE as the editor. The editor must be the name of an editor listed in the <i>editor.rules</i> file. The specified editor is used to determine which <i>editor.rules</i> file is to drive this particular instantiation of GTE. This option is required.• <code>table</code>—Specifies which table to load at start-up time. This must be a table created with the <code>-ed</code> editor option. If you do not specify a table, the editor is started with a blank table.• <code>-p projdir</code>—Uses the specified <code>projdir</code> directory as the current project directory (in Windows NT, requires a drive designation). The <i>projdir</i> variable specified with <code>-p</code> overrides the default directory specified by the <i>projdir</i> ToolInfo variable.• <code>-s system</code>—Uses the named system as the current system name. This name overrides the default system name given by the <i>system</i> ToolInfo variable. The system name is appended to the project directory to specify the location of directories for the system.• <code>-C command ...</code>—Executes the specified StP built-in command at GTE start-up time.• <code>-version</code>—Displays version information for the StP Core and, if appropriate, product components. It does not perform the normal action of the program.
Comments	<p>The GTE is basis for all the StP table editors. The GTE editor provides a concise and ordered format for capturing detailed information about node objects defined in diagrams.</p> <p>The particular set of attributes that specify table attributes in any given instantiation of GTE is determined by a rules file. You must invoke the gte command with the <code>-ed</code> parameter to specify the rules file to be used to drive this instantiation of GTE.</p> <p>Table attributes specified by the rules file include:</p> <ul style="list-style-type: none">• General table appearance (borders, column width, row height, shading, and write protection)• Row attributes (mapping of a row, cell, or horizontal or vertical section to a repository object, actions to be performed on such objects, and generating a repository query from a row)• Cell attributes (label appearance, shading, writer protection, horizontal and vertical spanning, and mapping to repository objects)

- Row sorting and semantics in horizontal and vertical sections
- Actions executed from an editor or other StP interface
- Features to be acquired from the license server to allow the editor to run
- Definitions of the table's menus

The GTE uses a select-then-operate user interface. You insert data by selecting a cell and then typing in data or selected it from a menu. Some data entered using the diagram editor is displayed in the object's table. Once in the cell, data can be selected and manipulated with a host of editing commands.

Files

The **gte** command affects the following files:

- *rules/editor.rules*—Rules file describing the available StP editors. A subset of these editors are table editors. The rest are diagram editors, or miscellaneous editors, such as the Object Annotation Editor (OAE). This file is searched for in the path specified with *stp_file_path* ToolInfo variable.
- *rules/editor_rules*—Rules file that drives a particular instantiation GTE. This file is searched for in the path specified with *stp_file_path* ToolInfo variable.

Examples

To instantiate GTE as the Attribute Table Editor as a background process:

```
gte -ed ate &
```

To instantiate GTE as the Class Table Editor on the table Message in the system Mysystem:

```
% gte -ed cte Message -s Mysystem
```

See Also

gde, stp

msgd

Function	Starts the StP Message Router, which is the central communication process to which all StP tools connect.
Syntax	msgd [&]
Comments	<p>The StP Message Router passes messages between StP tools in use by one or more StP users in order to synchronize information about the current StP system being edited. It also provides addresses to connection requests made by other StP processes.</p> <p>A single message router can handle message passing between StP tools invoked on one or more StP systems by one or more users on the network. However, no more than one message router can exist at any given time for any given StP system. If you are running an earlier release of StP on the same network as your current StP release, each requires its own message router.</p> <p>The StP Message Router can be:</p> <ul style="list-style-type: none">• Started explicitly and run continuously on a nominated host (recommended for most multi-user environments)• Started automatically by StP as needed, on the machine where StP was first started (default method, recommended for single-user environments) <p>Multi-user environments should run the StP Message Router continuously on a nominated host. Typically, the StP administrator issues the msgd command in a shell window or places it in a boot file to run the message router as a UNIX daemon. To run the message router as an automatic Windows NT service, see “Configuring StP Interprocess Communications” on page 3-1. Only the StP system administrator can invoke the msgd command.</p> <p>StP tools connect to the message router in either of two ways, depending on how the <i>msgd_host</i> ToolInfo variable is set in the user environment:</p> <ul style="list-style-type: none">• If <i>msgd_host</i> is unset, StP tools connect to the message router by broad casting a connection request.• If <i>msgd_host</i> is set to the name of a nominated host, StP tools connect to the message router on the host machine directly. <p>This setting must be set the same for all users. In single-user environments, use the default StP autostart and broadcast connection methods to start and connect to the message router. The autostart method requires no special configuration (<i>msgd_host</i> is unset) and you do not need to explicitly start the message router with the msgd command. If msgd is</p>

already running, StP tools connect to the message router by broadcasting a connection request. If **msgd** is not running, the StP process that initiated the connection request starts **msgd** locally.

See the **msgdiag** command for information on debugging **msgd**.

For more information, see “Configuring StP Interprocess Communications” on page 3-1.

Example

To start the StP Message Router as a UNIX daemon on a nominated host:
msgd &

See Also

stp, msgdiag

msgdiag

Function	Invokes the troubleshooter utility for msgd , the StP Message Router.
Syntax	msgdiag [-C command...]
Options	Option is as follows: <ul style="list-style-type: none">• -C arg1—Execute the specified command.
Comments	<p>The msgdiag command broadcasts an address request like a msgd client and displays all the replies it receives. It indicates where msgd is currently running and, optionally, what connections each msgd maintains. This command is generally used for troubleshooting the StP Message Router, and should be used only by the system administrator. No more than one message router should be running at any given time for any given StP system. If you are running an earlier release of StP on the same network as StP release 2.4, each requires its own message router.</p> <p>The msgdiag command operates on all releases of StP running on the same network or subnet. If the network contains subnets that disallow broadcasts across subnet boundaries, you should issue this command from each subnet for which you want information on msgd.</p>
Examples	<p>To find out where msgd is running, type:</p> <pre>msgdiag</pre> <p>This broadcasts an address request and displays all the replies it receives.</p> <p>To ask msgd to display its current connections:</p> <pre>msgdiag -C '>#'</pre> <p>The output contains both the location of the receiving msgd (hostname and port number) and details of each connection that it is maintaining.</p> <p>To kill all existing msgd processes for a single StP release, use stpem rather than msgdiag as follows:</p> <ol style="list-style-type: none">1. Make sure there are no current connections to any msgd for this StP release.2. Send the following stpem command requesting the termination of all message routers receiving it:<pre>stpem -C '>q'</pre> <p>When you need to debug the routine performed by msgd, or if connections are not being established correctly, you can run msgd in debug mode as follows:</p> <ol style="list-style-type: none">1. Kill all currently running msgd processes.

2. Then type:

```
msgd -#
```

This causes **msgd** to print logging information to standard error output, detailing all received address requests, all connections (as they are established), and all incoming and outgoing message strings.

Note: If you run a **msgd** client (such as **gde**) in debug mode, using the **-#** argument), it logs its address request broadcast messages and its connection establishment in the Message Log window. This allows you to see which network interface it is broadcasting on and where it connects (host and port of the **msgd** process).

See Also

stpem, msgd

nav_source

Function	Finds a selected item in the source code.
Syntax	nav_source [-p projdir] [-s system] -n name -t type [-class classname] -f file [-d directory] [-version] [-help]
Options	<p>Options are as follows:</p> <ul style="list-style-type: none">• -p projdir—Uses the specified <i>projdir</i> directory as the current project directory (in Windows NT, requires a drive designation). The <i>projdir</i> variable specified with -p overrides the default directory specified by the <i>projdir</i> ToolInfo variable.• -s system—Uses the named system as the current system name. This name overrides the default system name given by the <i>system</i> ToolInfo variable. The system name is appended to the project directory to specify the location of directories for the system.• -n name—The name of the function, data structure, or class you want to view.• -t type—Specifies a type; this can be one of function, global, library, sequence, selection, enumeration, typedef, class, or classmember.• -class classname—Specifies the scoping class when -t type is classmember.• -f file—Specifies a source file to search for the desired item. This can be a filename with the fully-specified path, or just the filename. If you specify only the filename, and the file is not in the current directory, you must also specify -d directory.• -d directory—Specifies the directory in which the file exists. Use this option when you do not specify the directory as part of -f file.• -version—Displays the version of nav_source. It does not perform the normal action of the program.• -help—Displays the syntax of the nav_source program.
Comments	<p>nav_source navigates from StP/UML to source code that has not been parsed by the reverse engineering tool. Navigations to source code from the StP editors all work by searching for the item's location in the semantic model, if any. If there is no semantic model, then nav_source is called to locate the line number of the desired item in the specified file. If an item is found, its name, file, line number, and directory are printed on the standard output. A message indicates when a navigation fails.</p>
See Also	xrb_search

oae

Function	Invokes the StP Object Annotation Editor (OAE).
Syntax	<code>oae [-p projdir] [-s system] [-version] [object_id]</code>
Options	<p>Options are as follows:</p> <ul style="list-style-type: none">• <code>-p projdir</code>—Uses the specified <code>projdir</code> directory as the current project directory (in Windows NT, requires a drive designation). The <i>projdir</i> variable specified with <code>-p</code> overrides the default directory specified by the <i>projdir</i> ToolInfo variable.• <code>-s system</code>—Uses the named system as the current system name. This name overrides the default system name given by the <i>system</i> ToolInfo variable. The system name is appended to the project directory to specify the location of directories for the system.• <code>-version</code>—Displays version information for the StP Core and, if appropriate, product components. It does not perform the normal action of the program.• <code>object_id</code>—Specifies the ID of the object to annotate.
Comments	<p>The oae command allows users to add additional information to certain CASE objects that cannot be represented in a diagram or table. Information is in the form of OMS items (typed name/value pairs and notes) with associated descriptions. The OAE uses a template to decide which notes and items are valid for any given OMS object type. The oae command is normally invoked by other StP tools and is controlled by means of a programmatic interface. If you invoke it directly from a shell, then specify the <code>object_id</code> of the object to annotate to cause an annotation to be loaded.</p>
Files	<p>The oae command affects the following files:</p> <ul style="list-style-type: none">• <i>rules/editor.rules</i>—The rules file describing all of the available StP editors. This is searched for in the path specified by the <i>stp_file_path</i> ToolInfo variable.• <i>rules/oae.rules</i>—The rules file that drives OAE. This is searched for in <i>stp_file_path</i>.• <i>oms/app.types</i>—The file describing (among other things) what annotation template to use for each OMS object type.• <i>annotation/template</i>—The file describing the legal notes and items for a given object type.
See Also	stp , repquery

qrp

Function	Processes scripts written in QRL (Query and Reporting Language).
Syntax	<code>qrp script_name [-p projdir] [-s system] [-t target] [-f format_file] [-o output_file] [-I include_directory]... [-x external_variable_name value]... [-trace argument] [-trace_file file] [-version]</code>
Options	<p>Options are as follows:</p> <ul style="list-style-type: none">• <code>script_name</code>—The name of the QRL script to be processed. If you specify a complete pathname, it will be used as given. If you specify only a simple filename, it will be searched for in the directories specified by the <code><stp_product>_stp_file_path</code> ToolInfo variable, with no relative offset, where <code><stp_product></code> is determined by the <code>product</code> ToolInfo variable. Therefore, to run a script from the current working directory, either the current working directory has to be one of the listed in <code><stp_product>_stp_file_path</code> (which is normally the case), or you have to provide the pathname. For example, <code>./myscript</code> runs the script <i>myscript</i> from the current directory.• <code>-p projdir</code>—Uses the specified <code>projdir</code> directory as the current project directory (in Windows NT, requires a drive designation). The <code>projdir</code> variable specified with <code>-p</code> overrides the default directory specified by the <code>projdir</code> ToolInfo variable.• <code>-s system</code>—Uses the named system as the current system name. This name overrides the default system name given by the <code>system</code> ToolInfo variable. The system name is appended to the project directory to specify the location of directories for the system.• <code>-t target</code>—The publishing target for generated documents. Allowable values are as follows:<ul style="list-style-type: none">• <code>ascii</code>—For normal ASCII output• <code>html</code>—For HTML output• <code>mif</code>—For output to FrameMaker-MIF• <code>leaf</code>—For output to Interleaf-ASCII• <code>rtf</code>—For output to RTF (Rich Text Format, used by PC-based publishing tools, including Microsoft Word)If you omit this option or use an illegal value, the default output is ASCII.• <code>-f format_file</code>—Controls the structure of a document produced by <code>qrp</code>, and provides the paragraph and character format selections available for use in a QRL script.

Each output target has its own particular type of `format_file`. They are:

- ASCII—StP rules file with ‘Document’ and ‘Paragraph’ rules.
- FrameMaker—FrameMaker document in MIF form.
- Interleaf—Interleaf document in Interleaf-ASCII form.
- HTML—Cascading Style Sheets (CSS).
- RTF—RTF-form document.

The `-f` option can be omitted for the ASCII target if you are using only system default formats, but is otherwise mandatory.

If you specify a complete pathname for the `format_file`, it will be used as given. If you specify only a simple filename, then it will be searched for, relative to the directories specified in the `<stp_product>_stp_file_path` ToolInfo variable, where `<stp_product>` is determined by the *product* ToolInfo variable. The relative location will be `/print_format/target`, where `target` in this case is either `ascii`, `framemaker`, `interleaf`, or `rtf`, corresponding to the selection made with the `-t` option. Therefore, to use a `format_file` in the current working directory, you either have to provide the complete pathname using `./format_file_name`, or the current working directory would have to have the correct name and relative offset from a `<stp_product>_stp_file_path` directory.

- `-o output_file`—Redirects generated output (from print statements) in the script to `output_file`. If you omit this option, output is generated to standard out.
- `-I include_directory`—Adds `include_directory` to the list of directories in which to search for files included by the `#include` or `#include_if_exists` directive. You can specify more than one `include_directory`.
- `-x external_variable_name value`—Initializes `external_variable_name` with the value specified by `value`. If this option is used, `value` overrides the initial values specified in the script, as well as values input through `scriptman`. The `external_variable_name` must correspond to an external variable declared in `script_name`, and `value` must be a value of the same type.
- `-trace argument`—Turns tracing on for argument. Values for argument are:
 - `calls`—Turns tracing on for all calls to user-defined functions
 - `name`—Turns tracing on for all calls to the `name` function, and causes any debugging message associated with the `name` function to be displayed if reached.
 - `returns`—Turns tracing on for all function returns, displaying the return value of each function called.

- **builtins**—Turns tracing on for all calls to built-in QRL functions, such as the list and string manipulation functions.
- **timestamps**—Turns profiling on in conjunction with any other trace output that is requested. Each trace message is preceded by a timestamp indicating the elapsed time since the QRP program began executing.
- **all**—Turns tracing on for all trace options except timestamps.
- **-trace_file file**—Redirects trace output from debugging messages to file. If you do not use this option, trace output is directed to standard error. Any **trace_file** built-in function in the script overrides the command-line option. Both the **-trace_file** option and the **trace_file()** function append messages to the file. When you rerun the script, new messages are added to the old ones, rather than writing over them.
- **-version**—Displays version information for the StP Core and, if appropriate, product components. Does not perform the normal action of the program.

Directories

qrp searches through the following directories:

- *projdir/system/qrl_files/scripts*—Location of system-specific scripts. Not automatically searched for by **qrp**.
- *qrl*—Location of StP product scripts, relative to directories specified with the ToolInfo variable, *<stp_product>_stp_file_path*, where *<stp_product>* is determined by the *product* ToolInfo variable. Not automatically searched for by **qrp**.
- *print_format/ascii*, *print_format/framemaker*, *print_format/interleaf*, *print_format/html*, *print_format/rtf*—Location of format files. These directories are searched for in the path specified with the *<stp_product>_stp_file_path* ToolInfo variable, where *<stp_product>* is determined by the *product* ToolInfo variable.

File

The **qrp** command affects the *templates/ct/app.types* file, which contains print methods for FileObjects, including diagram and table files, which are invoked from **qrp**.

Comments

The **qrp** command processes scripts written in QRL (Query and Reporting Language). One major function of **qrp** is to extract information from the repository and generate formatted reports. These reports may be documents or tables viewable in the StP Repository Browser.

Diagrams and tables can be included in non-ASCII documents produced by **qrp**. They are generated by print methods defined in the *app.types* file, which will normally invoke the StP diagram and table printing programs, **dprint** and **tprint**. The contents and appearance of diagrams and tables can be specified either in the QRL script or by using the Named print

Examples

settings, which are saved using the editor's Named Setting dialog box, accessible from the Print dialog box.

To generate ASCII output to standard output with no format file, using a script called *example_script*:

```
grp example_script
```

The following script, *script1.qrl*, generates FrameMaker output to the outfile file, using the *report.mif* format file, in the *print_format/framemaker* directory, and specifies an extra include directory, *my_include_dir*:

```
grp -t mif -f report.mif -I my_include_dir -o outfile  
script1.qrl
```

This script, *script1.qrl*, initializes two external variables, *int_var* and *string_var*. *int_var* is initialized with the value 1, and *string_var* is initialized with the value "String Value":

```
grp -x int_var 1 -x string_var "String Value" script1.qrl
```

This script, *script1.qrl*, generates ASCII output to the outfile file, using formats from the *report.asc* format file:

```
grp -o outfile -f report.asc script1.qrl
```

This script, *debug.qrl*, redirects normal output to *outfile*, and debugs the output to the *msgs.txt* file:

```
grp -o outfile -trace_file msgs.txt debug.qrl
```

This script, *traceprog.qrl*, records how long actions take. The output shows the elapsed time in milliseconds at each function call and return:

```
grp -trace all -trace timestamps traceprog.qrl
```

See Also

scriptman, **dprint**, **tprint**

re_db_check

Function	Checks for a corrupted semantic model
Syntax	re_db_check [-p projdir] [-s system] [-version] [-help]
Options	<p>Options are as follows:</p> <ul style="list-style-type: none">• -p projdir—Uses the specified projdir directory as the current project directory (in Windows NT, requires a drive designation). The projdir variable specified with -p overrides the default directory specified by the <i>projdir</i> ToolInfo variable.• -s system—Uses the named system as the current system name. This name overrides the default system name given by the <i>system</i> ToolInfo variable. The system name is appended to the project directory to specify the location of directories for the system.• -version—Displays the version of re_db_check. Does not perform the normal action of the program.• -help—Displays the syntax of the re_db_check program.
Comments	<p>re_db_check examines every table in the <i>revc_files/db_dir</i> database directory beneath the specified semantic model. It checks each table's index to ensure that key order and page order are correct, and that each entry points to a legitimate space in the data table.</p> <p>If the index table is correct, then the data table is traversed in order, and each piece of data is checked to see if it is legitimate and within a normal range.</p> <p>Some data will fail the range test, which may or may not indicate a problem about which you need to be concerned. The control table contains some timestamps that produce range warnings. Do not worry about these, even if it generates warnings about importance being very large. However, other range check failures are a legitimate concern.</p> <p>When re_db_check completes, it displays a message reporting whether the tables are safe to use. If the message reports that the database is inconsistent, re-parse the source files non-incrementally to produce a neat and consistent database.</p> <p>re_db_check only operates on tables that have not been opened by any process in a write mode. Use re_db_locks to find out who holds the locks on any table.</p>
See Also	re_db_locks

re_db_locks

Function	Checks for and reports database locks.
Syntax	re_db_locks [-p projdir] [-s system] [-version] [-help]
Options	<p>Options are as follows:</p> <ul style="list-style-type: none">• -p projdir—Uses the specified projdir directory as the current project directory (in Windows NT, requires a drive designation). The projdir variable specified with -p overrides the default directory specified by the <i>projdir</i> ToolInfo variable.• -s system—Uses the named system as the current system name. This name overrides the default system name given by the <i>system</i> ToolInfo variable. The system name is appended to the project directory to specify the location of directories for the system.• -version—Displays the version of re_db_locks. Does not perform the normal action of the program.• -help—Displays the syntax of the re_db_locks program.
Comments	<p>re_db_locks examines every table in the <i>db_dir</i> database directory beneath the specified semantic model. It removes redundant locks, and reports active locks. The program returns a non-zero return code if any tables are locked.</p> <p>Each table in the database consists of an index file, a data file, and, optionally, a lock file. When the table is opened, a line is added to the lock file stating the time, the process id, and the user and machine that applied the lock. When the table is closed, the line is removed from the lock file, and if the file is empty, it is removed. A maximum of one write lock can operate on a table at a given time. Many read locks can be applied at once, as long as no write locks exist.</p> <p>In addition to this visual locking strategy, system-level locking is also applied to the index file of any table that is opened. This mechanism provides safe network operation and ensures that whenever a process dies, it removes all its locks.</p> <p>If a table is found to contain no system-level lock, although a visual write lock is still held in the lock file, the process that was writing the table must have terminated unexpectedly. This can result in the table having an inconsistent state.</p> <p>re_db_locks warns if the database has possibly been corrupted. If you see this warning, run the re_db_check program to ensure it is consistent.</p>
See Also	re_db_check

re_gen_defines

Function	Regenerates #define information for a file that has been parsed by the reverse engineering tool
Syntax	re_gen_defines [-p projdir] [-s system] -f filename [-d directory] [-o output_file] [-quiet] [-version] [-help]
Options	<p>Options are as follows:</p> <ul style="list-style-type: none">• -p projdir—Uses the specified <i>projdir</i> directory as the current project directory (in Windows NT, requires a drive designation). The projdir variable specified with -p overrides the default directory specified by the <i>projdir</i> ToolInfo variable.• -s system—Uses the named system as the current system name. This name overrides the default system name given by the <i>system</i> ToolInfo variable. The system name is appended to the project directory to specify the location of directories for the system.• -f filename—Specifies the file for which #define information is to be regenerated. This can be a simple filename or a fully-specified filename. If the file is not in the current directory and you do not specify its location, you must use -d directory.• -d directory—Specifies the directory containing the file for which #define information is to be regenerated. Use this option when other files with the same name as filename exist in other directories in the semantic model.• -o output_file—Specifies where to write the output. Without this option, the output is written to <i>/tmp/Defines</i>.• -quiet—Suppresses the display of success and failure messages.• -version—Displays the version of re_gen_defines. It does not perform the normal action of the program.• -help—Displays the syntax of the re_gen_defines program.
Comments	<p>re_gen_defines regenerates #define information for a source or header file that has been parsed by the reverse engineering tool. The code generator cannot generate #define statements, so re_gen_defines provides that functionality for systems that originally were reverse-engineered.</p> <p>re_gen_defines finds all the #defines for the specified file and writes them to the <i>/tmp/Defines</i> file, unless an alternative output file is specified.</p>

re_make

Function	Reads a specified makefile and extracts information for parsing with ccparser .
Syntax	<code>re_make [-p projdir] [-s system] [-cd directory] [-f makefile] [-depth nn] [-keep] [macro=value ...] [target ...] [-nmake gmake]</code>
Options	<p>Options are as follows:</p> <ul style="list-style-type: none"> • <code>-p projdir</code>—Uses the specified <i>projdir</i> directory as the current project directory (in Windows NT, requires a drive designation). The <i>projdir</i> variable with <code>-p</code> overrides the default directory specified by the <i>projdir</i> ToolInfo variable. • <code>-s system</code>—Uses the named system as the current system name. This name overrides the default system name given by the <i>system</i> ToolInfo variable. The system name is appended to the project directory to specify the location of directories for the system. • <code>-cd directory</code>—Changes directory to the specified location before processing the makefile. • <code>-f makefile</code>—Reads the file specified as the makefile. If no filename is specified, then re_make looks for and reads a file called <code>makefile</code> or <code>Makefile</code> in the current directory. • <code>-depth nn</code>—Specifies how many levels of makefiles to recurse through. The default is not to specify any limit, in which case re_make continues until there are no sub-makefiles left to read. • <code>-keep</code>—Does not remove the previously gathered information. By default, re_make removes the information that was extracted the last time it was run. Sometimes you may want to run re_make incrementally on one or two specific makefiles, without recursing through many unrelated makefiles. To do this, run re_make without including the <code>-keep</code> option on the first makefile, with a <code>-depth</code> limit of 1. Then run re_make on the second Makefile with <code>-keep</code> set and a <code>-depth</code> limit of 1. • <code>macro=value</code>—Assigns a value to a macro. The specified value replaces the macro in the makefile. Many macros can be specified this way. • <code>target</code>—Specifies a target that needs to be analyzed in the makefile. By default, re_make analyzes the first target it finds in a makefile, although you can override that default by specifying one or more alternative targets. • <code>-nmake</code>—Specifies that the file is in a format suitable for Microsoft Nmake. • <code>-Gmake</code>—Specifies that the file is in a format suitable for the Free Software Foundation's Gnu Make.
Comments	The re_make command reads a makefile and determines the source files, search directories and preprocessor defines that are used to build the

executable. This information is written into the `<projdir>/<system>/revc_files` directory and can be subsequently used during parsing with `ccparser`.

The **re_make** command can recurse through a hierarchy of makefiles and gather all the source file names that build all the executables beneath a top-level makefile.

See Also**ccparser, smdb2uml**

repquery

Function	Queries the StP repository.
Syntax	repquery [-p projdir] [-s system] [-version]
Options	<p>Options are as follows:</p> <ul style="list-style-type: none">• -p projdir—Uses the specified <i>projdir</i> directory as the current project directory (in Windows NT, requires a drive designation). The <i>projdir</i> variable specified with -p overrides the default directory specified by the <i>projdir</i> ToolInfo variable.• -s system—Uses the named system as the current system name. This name overrides the default system name given by the <i>system</i> ToolInfo variable. The system name is appended to the project directory to specify the location of directories for the system.• -version—Displays version information for the StP Core and, if appropriate, product components. It does not perform the normal action of the program.
Comments	<p>repquery is an StP command-line utility program used to query the repository using the Object Management System (OMS) Query Language.</p> <p>When executed, repquery responds with an enter query now: prompt for an OMS Query Language query. As soon as the query is entered, it is executed and the results are displayed without pause, followed by another prompt to enter another query. Type quit to exit repquery.</p> <p>Note: repquery is generally only useful to the experienced OMS user or for debugging purposes. In many cases, the StP Repository Browser provides a more thorough, graphical means to query and browse the repository.</p>
Example	<p>To run repquery, execute a query, see the results, and then quit:</p> <pre>repquery enter query now: file[BachDiagram] [... multi-line display of objects found ...] enter query now: quit %</pre>
See Also	stp

repsync

Function	Updates the StP repository from a repsync file.
Syntax	repsync [-p project_dir] [-s system] [-verbose] [-Force] [-trans0 -trans1] [-version] [repsync_file]
Options	<p>Options are as follows:</p> <ul style="list-style-type: none">• -p projdir—Uses the specified <i>projdir</i> directory as the current project directory (in Windows NT, requires a drive designation). The projdir variable specified with -p overrides the default directory specified by the <i>projdir</i> ToolInfo variable.• -s system—Uses the named system as the current system name. This name overrides the default system name given by the <i>system</i> ToolInfo variable. The system name is appended to the project directory to specify the location of directories for the system.• -verbose—Lists the objects being manipulated by repsync, along with the action being performed and its status.• -Force—Forces an update of the repository, even if repsync considers the repository up-to-date with respect to the repsync_file.• -trans0—Does not use Object Management System (OMS) transactions and “commits” when updating the repository. This allows repsync to store a number of objects in the repository that would be too large for a transaction.• -trans1—Uses Object Management System (OMS) transactions and “commits” when updating the repository. (This is the default.)• -version—Displays version information for the StP Core and, if appropriate, product components. It does not perform the normal action of the program.• repsync_file—The specified repsync file(s) are used as input to update the repository. If no files are present, it is expected that a file is piped to repsync.
Comments	<p>The repsync command is used to update (synchronize) the repository from a repsync_file. Examples of files that are stored in repsync format are annotation files and rename shadow log files.</p> <p>When executed, repsync extracts the objects and object references that are specified in textual form in the repsync_file and stores them in the repository.</p> <p>Note: repsync is used for creating integrations with StP. It provides a higher level means of storing information in the repository than an OMS program.</p>
See Also	gde, stp

scriptman

Function	Invokes the Script Manager, which you can use to create, modify, copy, and run Query Repository System (QRS) scripts.
Syntax	scriptman [-p projdir] [-s system] [-version]
Options	<p>Options are as follows:</p> <ul style="list-style-type: none">• -p projdir—Uses the specified <i>projdir</i> directory as the current project directory (in Windows NT, requires a drive designation). The projdir variable specified with -p overrides the default directory specified by the <i>projdir</i> ToolInfo variable.• -s system—Uses the named system as the current system name. This name overrides the default system name given by the <i>system</i> ToolInfo variable. The system name is appended to the project directory to specify the location of directories for the system.• -version—Displays version information for the StP Core and, if appropriate, product components. It does not perform the normal action of the program.
Comments	<p>Use QRS scripts to browse and extract information from a Software Through Pictures (StP) repository and the operating system file system. The extracted information can be used to generate documents for popular Desktop Publishing packages (FrameMaker and Interleaf), as well as ASCII documents.</p> <p>The Script Manager allows users to:</p> <ul style="list-style-type: none">• Select, display, and browse QRS scripts in three areas: Product, System, and User• Create and generate documents using the scripts• Create new scripts, edit and copy existing scripts• Display and change the external variables associated with scripts
Files	<p>Files affected by the scriptman command are as follows:</p> <ul style="list-style-type: none">• <i>rules/scriptman.rules</i>—Rules file that drives the Script Manager. Searched for in the <stp_product>_stp_file_path ToolInfo variable.• <i>qrl</i>—Directory containing Product scripts. Searched for in <stp_product>_stp_file_path.• <projdir>/<system>/qrl_files/scripts—Directory containing System scripts. <p>Note: In these file search paths, <stp_product>_stp_file_path is a Tool-Info variable, where <stp_product> is determined by the <i>product</i> ToolInfo variable.</p>

Example To start the Script Manager in the background, using the */des/work* directory as the project directory:
`scriptman -p /des/work &`

See Also **qrp**

smbd2* Commands

smbd2uml	Generates StP/UML models from previously parsed C++ code.
Syntax	<code>smbd2uml [-p projdir] [-s system] [-external -noexternal] [-if exists [ignore overwrite merge]] [-norepsync] [-nodigram] [-notable] [-noannot] [-show_system] [-nooperationbodies]</code>
Options	<p>Options are as follows:</p> <ul style="list-style-type: none">• <code>-p projdir</code>—Uses the specified <i>projdir</i> directory as the current project directory (in Windows NT, requires a drive designation). The <i>projdir</i> variable specified with <code>-p</code> overrides the default directory specified by the <i>projdir</i> ToolInfo variable.• <code>-s system</code>—Uses the named system as the current system name. This name overrides the default system name given by the <i>system</i> ToolInfo variable. The system name is appended to the project directory to specify the location of directories for the system.• <code>-external</code>—Annotates captured classes as external.• <code>-noexternal</code>—Does not annotate captured classes as external.• <code>-if exists [ignore overwrite merge]</code>—Specifies whether to refresh the definition of any class that already exists in the repository if one of the same name is captured. Options are:<ul style="list-style-type: none">• <code>ignore</code>—Ignores the new class entirely; keeps the existing definition.• <code>overwrite</code>—Recaptures classes that already exist in the repository and overwrites any user annotations.• <code>merge</code>—Recaptures classes that already exist in the repository and merges their user annotations with any new annotations.• <code>-norepsync</code>—Only creates the files (diagrams, tables, and annotations) without updating the repository. It is assumed that the user will update the repository later from the files.• <code>-nodigram</code>—Does not create class diagrams for the captured classes.• <code>-notable</code>—Does not create class tables for the captured classes.• <code>-noannot</code>—Does not create annotation files for the captured classes.• <code>-show_system</code>—Creates classes for information defined in system header files.• <code>-nooperationbodies</code>—Disables the gathering of source code for operation bodies. By default, the source code for these is copied into an appropriate annotation.

Comments	<p>The smbd2uml command generates StP/UML models from information extracted from C++ source code during parsing. The parser executable can be run either from the StP Desktop using the Reverse Engineering > Parse Source Code command from the Code menu, or by directly running ccparser in a command window.</p> <p>For each parsed C++ class, smbd2buml creates:</p> <ul style="list-style-type: none">• A class table• Annotation files• A class diagram for each inheritance relationship that exists <p>All files, class table information, and annotations are registered in the repository, making the captured classes immediately available for reuse within the current system.</p> <p>When a parsed class has the same name as a class already existing in the repository (as when classes are recaptured), you can choose to ignore the new definition or overwrite the one in the repository. If you overwrite the old one, you can either merge old and new annotations or overwrite old ones.</p>
See Also	ccparser, re_make

stp

Function	Invokes the StP Desktop.
Syntax	<code>stp [-p projdir] [-s system] [-version]</code>
Options	Options are as follows: <ul style="list-style-type: none"> • <code>-p projdir</code>—Uses the specified <i>projdir</i> directory as the current project directory (in Windows NT, requires a drive designation). The <i>projdir</i> variable specified with <code>-p</code> overrides the default directory specified by the <i>projdir</i> ToolInfo variable. • <code>-s system</code>—Uses the named system as the current system name. This name overrides the default system name given by the <i>system</i> ToolInfo variable. The system name is appended to the project directory to specify the location of directories for the system. • <code>-version</code>—Displays version information for the StP Core and, if appropriate, product components. It does not perform the normal action of the program.
Comments	<p>The StP Desktop starts all StP product-specific editors and utilities. It also provides access to general StP features, including system administration and repository administration commands. The editors, utilities, and general features are displayed as icons organized within folders (called categories), which are arranged in a scrolling list on the StP Desktop.</p> <p>Users can select the project or system to be made current for the StP Desktop by selecting File > Open System, or by specifying the <code>-p projdir</code> and <code>-s system</code> options on the stp command line.</p> <p>While the StP Desktop appears to be tailored to a particular StP product, it is actually a generic mechanism that can be configured to provide a graphical interface to almost any command-line program. You may specify, in the <i>stp.rules</i> file, dialog boxes that simplify users' selection of exclusive, non-exclusive, and optional command-line arguments.</p>
File	The stp command affects the <i>rules/stp.rule</i> file, which drives the StP Desktop. It searches for this file in the path specified with <code><stp_product>_stp_file_path</code> ToolInfo variable, where <code><stp_product></code> is the name of an StP product.
Examples	<p>To start StP Desktop for the default project and system:</p> <pre>stp</pre> <p>To start StP Desktop for system design in project /des/work:</p> <pre>stp -s design -p /des/work</pre>
See Also	gde , gte , scriptman , stputil , toolloc , and the administrative functions described in <i>Query and Reporting System</i> .

stpem

Function	Sends messages to StP tools.
Syntax	stpem [-ed editor] [-p projdir] [-s system] [-first -handle handle] [-context context] -C command
Options	<p>Options are as follows:</p> <ul style="list-style-type: none"> • -ed editor—Specifies the name of the editor to which the message should be delivered. This option is required except when used to troubleshoot msgd, the message daemon. See “msgdiag” on page B-17 for information on troubleshooting msgd. • -p projdir—Uses the specified <i>projdir</i> directory as the current project directory. The projdir variable specified with -p overrides the default directory specified by the <i>projdir</i> ToolInfo variable. • -s system—Uses the named system as the current system name. This name overrides the default system name given by the <i>system</i> ToolInfo variable. The system name is appended to the project directory to specify the location of directories for the system. • -first—Delivers the message to just one instance of the given editor, no matter how many are running. • -handle handle—Delivers the message to the editor instance whose handle matches handle. • -context context—Specifies the context of the message. The default context is projdir/system. Only tools registered with a context matching that of the outgoing message are eligible to receive this context. • -C 'command'—Specifies the message contents. This message is typically a built-in stpem command and any arguments.
Comments	Use the stpem (StP Execution Manager) utility to send messages to StP tools. It permit actions to be initiated within a tool that are normally only available interactively from the graphic user interface. The stpem utility controls the execution of tools, starting new instances where necessary, to ensure that a valid recipient is available for all message requests.
Files	The stpem utility uses the <i>rules/editor.rules</i> file, which describes the available StP editors. stpem searches for this file in the path specified by the <i>stp_file_path</i> ToolInfo variable. Each editor rule contains a CreationCommand attribute that instructs stpem how to invoke a new instance of that editor.
Example	To iconify all instances of the UML Class Editor: stpem -ed uclassd -C EditorIconify
See Also	stp , msgd , msgdiag

stputil

Function	Invokes the StP Utility.
Syntax	stputil [-C command] [-Interactive] [-Quiet] [-verbose] [-Force] [-p projdir] [-s system] [-version]
Options	<p>Options are as follows:</p> <ul style="list-style-type: none">• -C command—Executes the StP Utility command non-interactively. (See “Commands,” below.)• -Interactive—Runs the StP Utility in interactive mode. If a -C command is specified, changes to interactive command mode after the command is executed.• -Quiet—Executes in quiet mode, without showing prompts and commands.• -verbose—Executes the commands in verbose mode.• -Force—Performs all requested actions without asking for confirmation.• -p projdir—Uses the specified <i>projdir</i> directory as the current project directory (in Windows NT, requires a drive designation). The <i>projdir</i> variable specified with -p overrides the default directory specified by the <i>projdir</i> ToolInfo variable.• -s system—Uses the named system as the current system name. This name overrides the default system name specified by the <i>system</i> ToolInfo variable. The system name is appended to the project directory to specify the location of directories for the system.• -version—Displays version information for the StP Core and, if appropriate, product components. It does not perform the normal action of the program.
Commands	<p>When run interactively, the StP Utility prompts for commands. The valid commands are:</p> <ul style="list-style-type: none">• cle[an] [options]—Deletes from the repository all objects that are no longer referenced. Valid options are -v[erbose], -f[orce] and:<ul style="list-style-type: none">-O obj_id—Applies operation to object with specified id-n—Performs operation on annotated objects• c[opy] [options] fromfile tofile—Copies files matching fromfile filename pattern over system/project boundaries. See “Names” on page B-40 for a description of filename patterns. Valid options are -e[ditor], -v[erbose], -f[orce] and:<ul style="list-style-type: none">-T filetype—Specifies a file type-n—Performs operation on annotations

- `d[ele] [options] fullname`—Deletes all diagram files matching `fullname` and all its associated files. See “Names” on page B-40 for a description of filename patterns. Valid options are `-e[ditor]`, `-v[erbose]`, `-f[orce]` and:
 - `-T filetype`—Specifies a file type
 - `-n`—Performs operation on annotations
- `e[ditor] [editor]`—Sets the editor to `editor` or examines the current editor setting. The editor is specified as `bach`, `chen`, `ate`, `dete`, and so on.
- `env[ironment]`—Enters environment subsystem.
- `f[iles] [options] [pattern]`—Lists files in the project/system directory that match the specified pattern. See “Names” on page B-40 for a description of filename patterns. Valid options are `-e[ditor]`, `-v[erbose]` and `-T filetype`, which specifies a file type
- `filet[ype] [options] [type]`—Sets the filetype to `type` or examines the current filetype. Valid options are `-v[erbose]` and `-h[elp]`.
- `fo[orce] [on | off]`—With no argument, shows status of force mode. Otherwise, turns force mode on or off.
- `h[elp]`—Prints out list of commands.
- `int[eractive] [on | off]`—With no argument, shows status of interactive mode. Otherwise, turns interactive mode on or off.
- `loc[k]`—Enters locking subsystem.
- `l[s] [option] [pattern]`—Lists the files in the repository that match the pattern. See “Names” on page B-40 for a description of file name patterns. Valid options are `-e[ditor]`, `-v[erbose]` and `-T filetype`, which specifies a filetype.
- `p[roject] [projdir]`—Sets a project directory to *projdir* or examines the current project directory value.
- `quie[t] [on | off]`—With no argument, shows status of quiet mode. Otherwise, turns quiet—mode on or off.
- `q[uit]`—Quits program.
- `r[ename] [options] fromname toname`—Changes filename from `fromname` to `toname`. Valid options are `-e[ditor]`, `-v[erbose]`, `-f[orce]`, and `-T filetype`, which specifies a filetype
- `st[atus] [option]`—Shows values of a project, system, and flags. Valid option is `-v[erbose]`.
- `s[ystem] [system]`—Sets a system directory to `system` or examines the current system directory value.
- `u[ndef] [options]`—Checks for files that are not known to the repository and imports them. Valid options are `-e[ditor]`, `-v[erbose]`, `-f[orce]`, and:
 - `-T filetype`—Specifies a filetype

Flags	<p>Flags are as follows:</p> <ul style="list-style-type: none">• V[erbose]—Toggles verbose mode. 0 disables and 1 enables.• Q[uiet]—Toggles quiet mode. 0 disables and 1 enables.• F[orce]—Toggles force mode. If on (1), stputil does not ask for confirmation before deleting files. 0 disables.• I[nteractive]—Toggles interactive mode, usually off if -C command is used. 0 disables and 1 enables.
Names	<p>Names are as follows:</p> <ul style="list-style-type: none">• name—The name of a diagram or table file.• pattern—A name with wildcard characters that are matched against the diagram or table names in the database.• fullname—A complete project/system/diagram name, specified in the form [[project:]system:]diagram. <p>If the higher parts of the directory hierarchy are omitted, the current values are used.</p>
Comments	<p>The StP Utility provides a variety of utilities to query and modify the project and system directory structure, including diagram and table files and the underlying repository.</p> <p>Note: The StP Utility is best accessed from StP Desktop (see “stp” on page B-36), rather than from the shell command. The StP Desktop provides a graphical user interface to all of the subsystems and most of the individual commands of the StP Utility.</p> <p>The following subsystems serve to group the available StP Utility commands:</p> <ul style="list-style-type: none">• Main Subsystem (general commands)• Locking Subsystem• Environment Subsystem <p>You enter the Main Subsystem by default when you run the StP Utility. The lock and env commands described below can be used to enter the Locking and Environment Subsystems, respectively.</p>
Examples	<p>To run StP Utility interactively, get help, then quit:</p> <pre>stputil ->help [... multi-line help display ...] ->quit %</pre>

To enter the Locking Subsystem in order to enter locking commands:

```
stputil -C lock -I
```

```
lock->
```

or

```
stputil
```

```
->lock
```

```
lock->
```

See Also **stp**

toolloc

Function	Displays the value of a ToolInfo variable.
Syntax	<code>toolloc [-p] [-w] ti_variable [default_value]</code>
Options	Options are as follows: <ul style="list-style-type: none"> • <code>ti_variable</code>—The ToolInfo variable being examined. • <code>default_value</code>—The value printed to standard output if no corresponding <code>ti_variable</code> is located in the ToolInfo file. This value is useful when piping the output of toolloc to other commands. • <code>-p</code>—When used without any arguments, prints the path to the first ToolInfo file that StP reads, which is at the top of the ToolInfo file hierarchy. • <code>-w</code>—Displays a table of ToolInfo files containing a <code>ti_variable</code> definition, along with the variable value set in each ToolInfo file. The file with the current definition is marked with an asterisk.
Comments	The toolloc command searches the chain of ToolInfo files for the <code>ti_variable</code> variable and prints the current value to standard output. If the specified <code>ti_variable</code> variable does not appear in the ToolInfo file, toolloc echoes the <code>ti_variable</code> string to standard output. For more information on ToolInfo variables, see “User Environments” on page 2-8.
Files	The <i>toolloc</i> file affects the ToolInfo file. The standard ToolInfo file supplied with StP products sets various StP configuration variables that need to be consistent among all users on a project. Users can also create other ToolInfo files that set ToolInfo variables to user-specific values for the user’s local environment. User-specific ToolInfo files should include the location of the project-wide ToolInfo file, so that StP can find the other configuration values it needs. Examples of commonly used ToolInfo variables are <i>projdir</i> and <i>system</i> , which specify, respectively, the default StP project directory and system. For more information, see “How ToolInfo Files Work” on page 2-8.
Examples	<p>To print the value of the <i>projdir</i> ToolInfo variable to standard output:</p> <pre>toolloc projdir</pre> <p>To print the value of the <i>system</i> ToolInfo variable to standard output. If no <i>system</i> variable is specified in the ToolInfo files, this example prints the default value, <i>system1</i>, to standard output:</p> <pre>toolloc system system1</pre> <p>To print a table of ToolInfo files containing the <i>system</i> variable:</p> <pre>toolloc -w system</pre>

To print the path to the first ToolInfo file that StP reads:

```
toolloc -p
```

tprint

Function	Generates FrameMaker, Interleaf, HTML, RTF, and Microsoft Word files from StP tables
Syntax	<pre>tprint -ed editor table1 [table2...] [-p projdir] [-s system] [-target mif leaf html rtf msword_printer] [-out outputfile] [-orientation landscape portrait] [-rules_print print_rules_file] [-print_setting setting_name] [-version]</pre>
Options	<p>Options are as follows:</p> <ul style="list-style-type: none"> • -ed editor—Specifies the StP editor used to create the table. This option is required. • table—Specifies which tables to format. The specified table need to have been created with editor. Specify at least one table. • -p projdir—Uses the specified <i>projdir</i> directory as the current project directory (in Windows NT, requires a drive designation). The <i>projdir</i> variable specified with -p overrides the default directory specified by the <i>projdir</i> ToolInfo variable. • -s system—Uses the named system as the current system name. This name overrides the default system name given by the <i>system</i> ToolInfo variable. The system name is appended to the project directory to specify the location of directories for the system. • -target mif leaf rtf msword_printer—Specifies one of the following output formats: <ul style="list-style-type: none"> • mif—FrameMaker MIF • leaf—Interleaf-ASCII • rtf—Rich Text Format • msword_printer—Microsoft Word (on Windows NT only) • html—HTML <p>If you do not specify the -target option, tprint uses the format specified by the current print setting.</p> • -out outputfile—Specifies the name of the output file. If you specify "-" as the outputfile, tprint generates the file to standard output. If you provide no output filename, tprint generates to the file specified by the current print setting. <p>If outputfile is a filename without a complete path, then tprint generates the output file in the directory in which it is executed.</p> • -orientation landscape portrait—Specifies the orientation of the table on the page.

Note that this option is normally specified in the print setting supplied to **tprint** in a rules file as the PageOrientation attribute. The **-orientation** option allows you to override the value in the rules file. The command-line option **landscape** yields a PageOrientation of LandscapeFit, while **portrait** yields PortraitFit.

- **-rules_print print_rules_file**—Specifies a customized print rules file that you want to use with table.

See the **-print_setting** option discussion for what happens when you use the two options together.

When you use this option without the **-print_setting** option, **tprint** looks for (in order of precedence):

1. A TablePrintSetting rule named **editor** in **print_rules_file**
2. A TablePrintSetting rule named “**editor**” in **print_rules_file**

- **-print_setting setting_name**—Specifies a print setting associated with table to be used to format the output.

If you use both the **-print_setting** and **-rules_print** options, **tprint** looks for a TablePrintSetting rule named **setting_name** in *print_rules_file*.

If you use this option without the **-rules_print** option, **tprint** looks for the following rules, in order of precedence:

1. A TablePrintSetting rule named **setting_name** in the named setting file associated with table
2. A TablePrintSetting rule named **setting_name** in the *print_default.rules* file

If you use neither the **-print_setting** nor the **-rules_print** option, **tprint** looks for (in order of precedence):

1. A TablePrintSetting rule named “**editor**” in the named setting file associated with table
2. A TablePrintSetting rule named **editor** in the *print_default.rules* file
3. A TablePrintSetting rule named “**editor**” in the *print_default.rules* file

If **tprint** is printing more than one table, it does not use the named setting files associated with the tables.

- **-version**—Displays version information for the StP Core and, if appropriate, product components. Does not perform the normal action of the program.

Comments	<p>The tprint command generates formatted versions of StP tables one of the following target formats:</p> <ul style="list-style-type: none">• FrameMaker MIF• Interleaf-ASCII• Rich Text Format (RTF, used by PC-based publishing tools)• Microsoft Word (Windows NT only)• HTML <p>tprint uses print settings to format tables. A print setting for a table is defined with a TablePrintSetting rule. The TablePrintSetting rule's attributes specify print formatting options, such as page size, margins, page orientation, and captions. Print settings are found in the following files:</p> <ul style="list-style-type: none">• The default print rules file (<i>print_default.rules</i>, located in the <i>template_s/ct/rules</i> directory relative to the path specified by the <i><stp_product>_stp_file_path</i> ToolInfo variable, where <i><stp_product></i> is determined by the <i>product</i> ToolInfo variable)• A named setting file (a print rules file associated with a specific table, created using a table editor's Named Setting dialog box, which is accessible from the Print dialog box)• A user-created customized print rules file, which can be passed to tprint on the command line <p>See <i>Fundamentals of StP</i> for information on named print settings.</p> <p>The editor TablePrintSetting print rule in the <i>print_default.rules</i> file provides default table print formatting options for tprint. To use non-default values for a specific table, you can use a named print setting. To use non-default values for multiple tables, you normally use a customized rules file. Alternatively, you can manually insert customized TablePrintSetting rules in the default print rules file.</p> <p>For more detailed information, see the <i>-rule_print</i> and <i>-print_setting</i> descriptions under "Options" on page B-44.</p>
File	<p>The tprint command uses the <i>print_default.rules</i>, which is the rules file defining the default settings for all StP printing, including printing tables from the Desktop, QRL scripts, and tprint and dprint commands.</p>
Examples	<p>To generate a FrameMaker MIF document for the Message table made with the Class Table Editor:</p> <pre>tprint -ed cte Message -target mif</pre>

To generate an Interleaf-ASCII document for the Message table made with the State Table Editor, using the named print setting “ileafdoc” created for the table from the State Table Editor:

```
tprint -e ste Message -target leaf -print_setting ileafdoc
```

To generate a FrameMaker document for the User class table, using TablePrintSetting rule “cte” in the customized print rules file *Mysettings.rules*:

```
tprint -ed cte User \  
-rules_print Mysettings.rules -target mif
```

See Also **gte, dprint, qrp, stp**

Update Commands

aupdate	Updates an annotation file.
dupdate	Updates a diagram file.
tupdate	Updates a table file.
Syntax	<p>aupdate filename... [[-replace] -o outputfile] [-p projdir] [-s system] [-update] [-batch]</p> <p>dupdate -ed editor filename... [[-replace] -o outputfile] [-Type table_type] [-p projdir] [-s system] [-update] [-Force] [-batch]</p> <p>tupdate -ed editor filename... [[-replace] -o outputfile] [-p projdir] [-s system] [-update] [-Force] [-batch]</p>
Options	<p>Options are as follows:</p> <ul style="list-style-type: none"> • -ed editor—Names the editor used to create the diagram or table files being updated. The specified editor must be an editor named in the <i>editor.rules</i> file. This option is used only with dupdate or tupdate, and is required. • filename—Specifies a diagram, table, or annotation file to be updated. You can specify multiple files. This argument is required for dupdate, tupdate, and aupdate. • -replace—Replaces each filename in place with an updated version. Output is therefore not written to standard output. • -Force—Causes dupdate or tupdate to update files whether or not they are identified as out-of-date. This is useful for updating files that exist in the system, when more recent versions have been manually copied over them. • -o outputfile—Updates filename and writes the output to outputfile, which must specify the full pathname and filename of the file. If you do not specify an outputfile (and also do not use -replace), the output is written to standard output. • -p projdir—Uses the specified <i>projdir</i> directory as the current project directory (in Windows NT, requires a drive designation). The <i>projdir</i> variable specified with -p overrides the default directory specified by the <i>projdir</i> ToolInfo variable. • -s system—Use the named system as the current system name. This name overrides the default system name given by the <i>system</i> ToolInfo variable. The system name is appended to the project directory to specify the location of directories for the system. • -update—Removes the cached object ids and remaps the symbols in the file.

- **-batch**—Processes large batches of files. This options speeds up processing by truncating the transaction log at intervals, and only stops if the repository becomes full. If you do not specify the **-batch** option, update process stops whenever any file update fails. Displays a running report on files being processed.

Comments

The update commands synchronize diagram, table, and annotation files with the repository in a system.

Under normal circumstances, you do not have to call these programs directly. They are called by other StP utility programs, such as the SysUpdate QRL function.

The **Rename Object Systemwide** command requires caching object ids in the diagram, table, and annotation files. For performance reasons, StP does not update these files with the new name until the next time it loads them into memory. As a result, these files should be updated when:

- Copying a diagram, table, or annotation to a new system or within an existing system
- Renaming a diagram or table within an existing system
- Doing anything that directly manipulates a diagram, table, or annotation file (such as some customizations)

The executor must have write access to all files to be processed. If an out-of-date file is under version control and checked out by another user or does not have proper file permissions, the command fails.

The update commands:

1. Look for any pending renames in the file being updated.
2. If no pending renames are found, leave the file as is.
3. If pending renames are found, resolve them and write out the file as if the editor had performed a save operation. Unless either the **-replace** or **-o outputfile** option is specified, the update commands send the output to standard output.

Examples

The following examples illustrate how the update commands are typically used.

To update the Message diagram file created with the Object Model Editor:

```
dupdate -ed ome -replace Message
```

To update the *a10* annotation file and write the output to the *10.ant.new* file:

```
% aupdate -o a10.ant.new a10
```

See Also

SysUpdate function, described in *Query and Reporting System*.

xrb_search

Function Search tool for the semantic model.

Syntax `xrb_search [-p projdir] [-s system] -T obj_type [-name name] [-key obj_id] [-scope name] [-read] [-write] [-def] [-ref] [-decs] [-ncs] [-tal] [-ff focus_file] [-version] [-help]`

Options Options are as follows:

- `-p projdir`—Uses the specified *projdir* directory as the current project directory (in Windows NT, requires a drive designation). The *projdir* variable specified with `-p` overrides the default directory specified by the *projdir* ToolInfo variable.
- `-s system`—Uses the named system as the current system name. This name overrides the default system name given by the *system* ToolInfo variable. The system name is appended to the project directory to specify the location of directories for the system.
- `-name name`—Specifies the name of the object to search for.
- `-key obj_i`—Specifies the object ID of the object to search for. If you specify both the `-name` and `-key` options, `-key` overrides `-name`.
- `-T obj_type`—Specifies the type of the object to search for. *obj_type* can be one of the following strings:

comment	function
constant	identifier
datamember	global
datastructure	library
define	literal
file	typedef

- `-scope name`—Limits the scope of the search. The *name* parameter specifies the name of a parent function or data structure.
- `-rea`—Specifies that read accesses should be located during the reference search. The default is to find both reads and writes.
- `-write`—Specifies that write accesses should be located during the reference search.
- `-def`—Finds the object's definitions.

- -decs—Finds declarations of the specified type, if the type category is typedef or datastructure.
- -ncs Not Case Sensitive mode—Searches for the object regardless of the case of name. The default search mode is case sensitive.
- -tal Treat As Literal mode—Used for searching in obj_type of literal or comment. The default is to interpret wildcard characters. Using this option prevents this interpretation.
- -version—Displays the version of **xrb_search**. It does not perform the normal action of the program.
- -help—Displays the syntax of the **xrb_search** program.

Comments

xrb_search is a powerful search engine that acts over the specified semantic model. You can use **xrb_search** to find definitions and references of many object categories in the semantic model.

The result of **xrb_search** queries is a textual list of source files and line numbers where the search criteria were satisfied.

See Also

nav_source

Index

- (hyphen) in StP Utility 4-5
* (asterisk) in StP Utility 4-5
? (question mark) in StP Utility 4-5
[] (square brackets) in StP Utility 4-5

A

Adaptive Server
 devices
 creating systems on 3-12
 default 3-12
 transaction log 3-13
Add New Files to Current System
 Repository command 1-13, 3-21
Add Repository Manager User(s)
 command 2-7
Add Sybase User(s) (Repository Manager)
 command 1-10
Add Sybase Users (Current System)
 command 1-14, 2-22
Add Users (Current System)
 command 2-22
annotations
 invoking OAE B-20
 missing from OAE display 3-8
 saving 1-2
 updating files B-48 to B-49
Aonix
 documentation comments x
 Technical Support xi
 websites xi

C

C++ code
 StP models, generating B-34 to B-35
C++ source code, parsing B-2
Change Owner (Current System)
 command 1-14
Change Sybase Owner of Current System
 command 2-19
Change Sybase User(s) Group 2-21
Change Sybase User(s) Group (Current System) command 1-14
Change Sybase User(s) Password
 command 1-11
Change User(s) Group (Current System)
 command 2-23
Change User(s) Password
 command 2-32
Change User(s) Sybase Password
 command 2-33
check command (StP Utility) 4-18
Check if System Repository Exists
 command 1-12, 2-24, 3-27
clean command (StP Utility) 4-6
commands
 aupdate B-48 to B-49
 ccparser B-2 to B-3
 comments B-4 to B-6
 dprint B-7 to B-10
 dupdate B-48 to B-49
 gde B-11 to B-12
 gte B-13 to B-14

msgd B-15 to B-16
msgdiag B-17 to B-18
nav_source B-19
oae B-20
qrp B-21 to B-24
re_db_check B-25
re_db_locks B-26
re_gen_defines B-27
re_make B-28 to B-29
repquery B-30
repsync B-31
scriptman B-32 to B-33
smbd* B-34 to B-35
stp B-36
stpem B-37
stputil B-38 to B-41
toolloc B-42 to B-43
tprint B-44 to B-47
tupdate B-48 to B-49
xrb_search B-50 to B-51
comments, gathering B-4 to B-6
communications between StP
 tools 3-1 to 3-11
copy command (StP Utility) 4-7
Copy System command 1-12, 2-27
Create System command 1-11

D

dateformat (TI variable) A-1
delete command (StP Utility) 4-8
Delete Repository Data in Current System
 command 1-13, 3-22
Delete Repository Manager User(s)
 command 2-5
Delete Sybase User(s) (Current System)
 command 1-14, 2-22
Delete Sybase User(s) (Repository
 Manager) command 1-10
Delete Unref'd Objects in System
 Repository command 3-17
Delete Unreferenced Objects in Current
 System Repository
 command 1-13
Destroy System command 1-12, 3-23

Destroy System Repository
 command 1-12, 3-23

devices
 adding 3-16
 default 2-17, 2-18, 3-16
 advantages 3-12
 master 3-15
 repository placement 3-11 to 3-13
 showing space available on 1-11, 3-15
 specifying for system creation 2-17
 storage device failure 3-24
 sysprocsdev 3-15
 system creation on 3-12
 transaction logs 3-13

diagrams
 GDE, invoking B-11 to B-12
 printing B-7 to B-10
 problems loading 3-8
 removing from system 4-8
 renaming 4-15
 saving 1-2, 1-3
 updating files B-48 to B-49

do_not_use_dbsetversion (TI
 variable) A-1

DOORS
 integrating with StP 5-1 to 5-6
 See also StP/DOORS

DSQUERY environment variable 3-16

Dump Current System Repository to Files
 command 1-13, 2-27, 3-25

dump directories 3-25

E

editor command (StP Utility) 4-9
editor.rules file 4-19

environment
 customizing 2-8
 Windows NT users 2-15

environment command (StP Utility) 4-4,
 4-9, 4-18

environment subsystem (StP
 Utility) 4-18 to 4-19

Environment variables
 See also ToolInfo variables

environment variables

- <server>_PASSWD 2-29
- DBDEVICE 2-10
- DSQUERY 2-11, 3-16
- IDE_MSGD_HOST 2-10, 3-7
- IDE_MSGD_PORT 2-10
- IDE_PRODUCT 2-10
- IDE_PROJDIR 1-8, 2-10
- IDE_SYSTEM 1-8, 2-11
- listed 2-9
- LOGDEVICE 2-10
- LOGSIZE 2-11
- PATH 5-3
- precedence over ToolInfo variables 2-9
- setting (WIndows NT) 2-15
- SIZE 2-11
- system defaults, setting 2-17 to 2-18
- system versus user variables 2-15
- ToolInfo 2-8, 2-11, 2-15
- ToolInfo equivalents 2-9
- USER 2-11

Expand Current System Repository
command 1-12, 3-20, 3-21

F

files

- adding/copying to systems 1-13, 3-21, 4-7
- contents undefined in repository 4-17
- current file type 4-11
- deleting from system 4-8
- dump 3-26
- editor.rules 4-19
- file types, displaying 4-19
- history 1-13, 3-18
- listing for system 4-10, 4-13
- .repinfo file 2-19
- restoring repository from 1-13, 3-24, 3-26 to 3-28
- saving repository to 1-13, 2-27, 3-25
- system 1-2, 1-3, 3-27

files command (StP Utility) 4-10

filetype command (StP Utility) 4-11

filter_path (TI variable) A-1

fix content command (StP Utility) 4-11

force command (StP Utility) 4-3, 4-12

functions

- repos_add_maker 2-6, 2-7
- repos_add_user 2-4, 2-7
- repos_change_password 2-32, 2-33
- repos_delete_maker 2-7
- repos_delete_user 2-5
- repos_list_current_users 2-7
- repos_list_reps 3-15
- repos_list_users 2-7
- repos_maint 3-14
- repos_show_space 3-15
- sys_add_user 2-21, 2-22
- sys_change_repowner 2-20
- sys_change_user 2-23
- sys_clean_rep 3-22
- sys_create 2-17, 2-19
- sys_delete_user 2-23
- sys_destroy 3-24
- sys_destroy_rep 3-23
- sys_dump_rep 3-26
- sys_expand_rep 3-21
- sys_has_rep 2-25
- sys_list_current_users 2-24
- sys_list_users 2-24
- sys_load_rep 3-27
- sys_recover_rep 3-28
- sys_show_space 2-26
- sys_trunc_hist 3-18
- sys_update 3-22

functions, QRL

- sys_create 3-13

G

Generic Diagram Editor (GDE)

- invoking B-11 to B-12

Generic Table Editor (GTE),

- invoking B-13 to B-14

Grant User(s) Sybase System Creation
Privileges command 1-10

Grant User(s) System Creation Privileges
command 2-5, 2-7

H

help command (StP Utility) 4-3, 4-12
history files 1-13, 3-18

I

IDE_MSGD_HOST environment
variable 3-7
IDE_PROJDIR environment variable 1-8
IDE_SYSTEM environment variable 1-8
includir (TI variable) A-2
interactive command (StP Utility) 4-3,
4-12

L

List All Users (Current System)
command 1-13, 2-24
List All Users (Repository Manager)
command 2-7
List All Users command 1-10
list command (StP Utility) 4-19
List Current Users (Current System)
command 1-14, 2-24
List Current Users (Repository Manager)
command 1-10, 2-7
List Locks command 1-9
List Repositories command 1-11, 3-15
Load Current System Repository from
Files command 1-13, 3-26
lock command (StP Utility) 4-4, 4-13, 4-19
lock_auto_timeout (TI variable) A-2
locks
finding B-26
listing 1-9
lock administrators, setting 1-9, 2-19
locking subsystem (StP Utility) 4-4,
4-13
managing 1-9
ls command (StP Utility) 4-13

M

mach_name_sep (TI variable) A-2

makefiles, reading and
extracting B-28 to B-29
Manage Locks command 1-9
message router (msgd)
broadcast connection method 3-6
connecting to 3-2, 3-6 to 3-7
default setup 3-3
defined 3-2
determining where running 3-10
IDE_MSGD_HOST and 3-7
invoking with msgd
command B-15 to B-16
losing connection to 3-8, 3-9
msgd_host ToolInfo variable 3-3
multi-user configuration 3-4 to 3-5
nominated host 3-2, 3-5 to 3-6
nominated host, questionable
settings 3-3
running as Windows NT service 3-5
single-user configuration 3-3
starting 3-2, 3-5, 3-6
terminating 3-8, 3-9, 3-11
troubleshooting 3-7 to 3-11
troubleshooting with
msgdiag B-17 to B-18
using across subnets 3-4, 3-9 to 3-10
Windows NT start-up modes 3-6
Micorosoft Jet
repositories, too large 3-19
Microsoft Jet 1-4
documentation 1-4
system administrator 1-5
monthnames (TI variable) A-2
msg_file (TI variable) A-2
msgd_host (TI variable) A-2
msgd_host ToolInfo variable 3-2
msgd_host ToolInfo variable, *See also*
IDE_MSGD_HOST environment
variable
msgd_port (TI variable) A-2
msgdiag command 3-10

N

names

- repository 2-19, 2-25
- system 2-16, 2-25

non-secure repository manager 1-6, 1-7, 2-29, 2-30

O

Object Annotation Editor (OAE)

- invoking B-20

objects

- deleting unreferenced 1-13, 3-17, 4-6
- regenerating from system files 4-17

objects, exporting to DOORS 5-2

Open System command 1-12

P

partially secure repository manager 1-6, 2-30

passwords

- <server>_PASSWD environment variable 2-28 to 2-29
- security and 1-6, 2-27, 2-29, 2-29 to 2-31
- syntax 2-31
- system administrator 1-6, 2-27, 2-30
- system administrator, changing 2-32
- troubleshooting 2-34
- user, changing 1-11, 2-33 to 2-34
- user, new 2-7, 2-33

pattern matching in StP Utility 4-5

PDM, *See* Persistent Data Model

Perform Manager Maintenance
command 1-11, 3-14

Persistent Data Model 1-3

pick_fully_inside (TI variable) A-3

print commands

- dprint B-7 to B-10
- tprint B-44 to B-47

print_mif_importable (TI variable) A-3

privileges

- StP administrator 1-5
- system creation 1-10, 2-5

system read/write user group 2-21, 2-23

product (TI variable) A-3

projdir (TI variable) A-4

projdir ToolInfo variable 1-8

project

- setting 1-8

project command (StP Utility) 4-14

projects

- current 1-8, 4-14
- current project, displayed 4-16
- description 1-2

Q

Query Reporting Language (QRL), script
processing B-21 to B-24

Query Reporting System (QRS) scripts,
managing B-32 to B-33

quiet command (StP Utility) 4-3, 4-14

quit command (StP Utility) 4-4, 4-14

R

read group privileges 2-21, 2-22, 2-23

Recover System Repository

- command 1-13, 3-28, 4-17

rename command (StP Utility) 4-15

renamed objects, with StP/DOORS 5-6

renaming objects, incomplete results 3-8

.repinfo file 2-19

repositories

See also systems

copying files between 3-21, 3-25

debugging B-30

deleting data 1-13, 3-22

description 1-2

destroying 1-12, 3-23

devices, assigning to 3-11 to 3-13

expanding 3-12

files lacking defined objects 4-17

files, updating 3-21

history files, removing 1-13

listing 1-11, 3-15

Microsoft Jet system defaults 2-18

names 2-19, 2-25
 owners 2-15
 querying B-30
 recovering incremental changes 3-27
 restoring 1-13, 2-25, 3-24, 3-26 to 3-28
 saving to dump files 1-13, 2-27, 3-25
 size, expanding 1-12, 2-18, 3-20
 size, setting default 3-20
 Sybase system defaults 2-17
 unreferenced objects, deleting 1-13,
 3-17, 4-6
 updating with repsync files B-31
 validating contents of 4-18
 verifying existence of 1-12, 2-24, 3-27
 repository manager
 adding users 1-10, 2-4
 administration commands 1-10 to 1-11
 deleting users 1-10, 2-5
 description 1-3
 listing all users 2-7
 listing current users 2-7
 listing users 1-10
 maintenance 1-11, 3-14 to 3-16
 passwords 2-27
 security 1-6, 2-2, 2-29, 2-29 to 2-31
 showing device space usage 1-11, 3-15
 system administrator privileges 1-7
 tasks 1-3, 1-5, 3-14 to 3-16
 users, current 1-10
 users, described 2-2, 2-4
 Repository menu 1-10 to 1-14
 reverse engineering, #define
 information B-27
 Revoke User(s) Sybase System Creation
 Privileges command 1-10
 Revoke User(s) System Creation Privileges
 command 2-6
 rte_CascadeRequirementAllocation (TI
 variable) A-4

S
 sa, *See* StP administrator
 save_scale_info (TI variable) A-4
 schema command (StP Utility) 4-15

 scope_comp_separator (TI variable) A-4
 scope_separator_char (TI variable) A-4
 Script Manager, invoking B-32 to B-33
 scriptman_ascii_view (TI variable) A-4
 scriptman_external_editor (TI
 variable) A-5
 scriptman_script_editor (TI variable) A-5
 scriptman_script_process (TI
 variable) A-5
 scriptman_ttywait (TI variable) A-5
 scriptman_user_scripts_location (TI
 variable) A-5
 scroll_line (TI variable) A-5
 secure repository manager 1-6, 2-30
 security, *See* repository manager
 semantic model search tool B-50 to B-51
 semantic models, checking for
 corruption B-25
 Set Lock Administrators command 1-9,
 2-19
 Show Sybase Server Space
 command 1-11, 3-15
 Show System Space 1-12
 Show System Space
 command 2-25 to 2-26
 Show ToolInfo Variable command 1-9,
 2-13, 2-26
 source code, searching through B-19
 source_editor (TI variable) A-6
 sp_diskdefault Sybase system
 procedure 3-16
 SQL Server
 devices
 creating systems on 3-12
 default 3-12
 transaction log 3-13
 indexes, updating 3-14
 optimizer 3-14
 passwords 2-28
 specified in system directory 2-19
 system failure 3-24
 status command (StP Utility) 4-16
 storage manager, *See* repository manager

StP

- configuring for multiple users 3-1 to 3-11
- configuring for subnet environments 3-4
- different releases on same network 3-5
- editor, current setting 4-9, 4-16
- installation 1-7
- products, described 1-1
- structural overview 1-4
- terminology defined 1-1
- tools, sending messages to B-37

StP administration

- See also* repositories, repository manager, StP administrator, systems
- command usage, Desktop 1-9 to 1-14
- commands, described 1-9 to 1-14
- prerequisites for 1-7
- repository manager
 - commands 1-10 to 1-11
- repository manager tasks 1-3, 1-5
- system repository and file manipulation
 - commands 1-9 to 1-14
- system-level commands 1-11 to 1-14
- system-level tasks 1-5, 2-20

StP administrator

- described 1-5
- password, changing 2-32
- password, using 1-6, 2-27 to 2-31
- privileges 1-5, 1-6, 1-7

StP Desktop

- Administer category 1-9
- description 1-7
- invoking B-36
- setting project and system 1-8
- starting (NT procedure) 1-8

StP message router *See* message router (msgd)

StP Utility

- character string arguments 4-4
- commands 4-5 to 4-19
- Commands menu 1-9 to 1-14
- commands, using 4-2
- described 4-1
- environment subsystem
 - commands 4-18 to 4-19
- exiting 4-4, 4-14
- exiting subsystems 4-4
- force mode 4-3, 4-12
- help 4-3, 4-12
- interactive mode 4-3, 4-12
- invoking B-38 to B-41
- listing commands 4-4
- locking subsystem 4-4, 4-13, 4-19
- main level commands 4-5 to 4-18
- modes 4-2 to 4-3, 4-16
- names in command arguments 4-5
- pattern matching in arguments 4-5
- quiet mode 4-3, 4-14, 4-18
- starting 1-9, 4-2
- status 4-16
- subsystems 4-1, 4-4
- verbose mode 4-3, 4-18

StP Utility command 1-9, 4-2

StP/DOORS

- integrating 5-1 to 5-6
- navigating from DOORS to StP 5-5
- navigating from StP to DOORS 5-4
- starting 5-4

stp_file_path (TI variable) A-6

stp_ps_print (TI variable) A-6

stpd_ttermwin (TI variable) A-6

stpem command 3-11

stputil command 1-9

subnets, configuring StP for use in 3-4

Sybase 1-4

- documentation 1-4
- system administrator 1-5

Synchronize DOORS with StP

- command 1-9

sys_create QRL function 3-13

syscreate_device ToolInfo variable 3-11

syscreate_log_device ToolInfo variable 3-13

syscreate_size ToolInfo variable 3-13

system

- setting 1-8

system (TI variable) A-7

system administrator, *See* SQL Server, StP administrator
system command (StP Utility) 4-16
system ToolInfo variable 1-8
systems
 See also repositories
 administration commands 1-11 to 1-14
 administration tasks 1-5, 2-20
 administrator privileges 1-6
 copying 1-12, 2-27, 3-27
 creating 1-11, 2-16 to 2-19
 creating on default devices 3-12
 creating on specific devices 3-12
 creation privileges, granting 1-10, 2-5
 creation privileges, revoking 1-10, 2-6
 current system, displaying 4-16
 current system, setting 4-16
 default settings 2-17 to 2-19
 description 1-2
 destroying 1-12, 3-23
 devices, specifying 2-17 to 2-18
 directory structure 2-16, 4-19
 file contents undefined in
 repository 4-17
 files, adding/copying 1-13, 3-21, 4-7
 files, ASCII 1-2, 1-3
 files, deleting 4-8
 files, listing 4-10, 4-13
 management 3-17 to 3-24
 names 2-16, 2-25
 opening 1-12
 owners 1-6, 2-15, 2-20
 ownership, changing 1-14, 2-19
 repository name 2-19
 security 2-2
 server name 2-19
 size, setting default 2-17 to 2-18
 storage space, expanding 1-12,
 3-20 to 3-21
 storage space, reclaiming 3-17 to 3-18
 storage space, showing 1-12, 2-25
 user description 2-2, 2-21
 user group read/write privileges 1-14,
 2-21, 2-23
 users, adding 1-14, 2-22

users, deleting 1-14, 2-22
users, listing 1-13, 1-14
users, listing all 2-24
users, listing current 2-24

T

tables
 GTE, invoking B-13 to B-14
 printing B-44 to B-47
 saving 1-2, 1-3
 updating files B-48 to B-49
Technical Support xi
time_format (TI variable) A-7
timeformat (TI variable) A-7
tmp_dir (TI variable) A-7
ToolInfo (TI variable) A-7
ToolInfo file 2-8
 multiple ToolInfo files 2-8
ToolInfo variables
 creating your own 2-13
 description 2-11 to 2-12
 displaying values B-42 to B-43
 environment variables, precedence 2-9
 finding the current value 2-13
 ide_product, *see* ToolInfo variables,
 product
 msgd_host 3-2
 msgd_host 2-10
 msgd_port 2-10
 msjet_admin_password 2-10, A-3
 msjet_password 2-10, A-3
 oms_dbuser 2-3, 2-10, A-3
 oms_debug_sql A-3
 product 2-10
 projdir 1-8
 projdir 2-10
scriptman_product_
 externals_location A-5
scriptman_system_
 externals_location A-5
scriptman_user_
 externals_location A-5
<server>_admin_password 2-10, 2-28,
 A-5

- `<server>_password` 2-10, 2-28, A-6
- showing 1-9, 2-26
- `syscreate_device` 2-10, 2-17, 3-11, 3-20, A-6
- `syscreate_log_device` 2-10, 2-17, 3-13, A-6
- `syscreate_log_size` 2-11, 2-17, A-7
- `syscreate_size` 2-11, 2-17, 3-13, 3-20, A-7
- system 1-8
- `system` 2-11
- `ToolInfo` 2-11
- types of 2-11
- `use_doors_integration` 5-3
- `toolloc` 2-13 to 2-14
- transaction logs
 - assigning to a device 3-13
 - dumping 3-14
 - repository recovery 3-24
- troubleshooting
 - annotations don't appear in OAE 3-8
 - diagrams won't load from Desktop 3-8
 - lost connection to msgd 3-8, 3-9
 - message router different from `msgd_host` setting 3-8
 - message router terminates on startup 3-9
 - message routing problems 3-7 to 3-11
 - object renaming is incomplete 3-8
 - passwords 2-34
- Truncate History for Current System
 - command 1-13, 3-18

U

- undef command (StP Utility) 4-17
- unreferenced objects, deleting 1-13, 3-17, 4-6
- update command (StP Utility) 4-19
- `use_doors_integration` (TI variable) A-7
- `use_offpage` (TI variable) A-7
- user accounts
 - Microsoft Jet 2-3
 - Sybase 2-2
- user logins 2-2
- user names
 - different from user login 2-3

users

- adding to repository manager 1-10, 2-4
- adding to systems 1-14, 2-22
- deleting from repository manager 1-10, 2-5
- deleting from systems 2-22
- description 2-1 to 2-3
- listing all for current system 2-24
- listing all for repository manager 2-7
- listing current for current system 2-24
- listing current repository manager 2-7
- listing for current system 1-13, 1-14
- listing for repository manager 1-10
- passwords 1-11, 2-7, 2-27 to 2-31, 2-33 to 2-34
- repository manager 2-2, 2-4 to 2-7
- revoking system access 1-14
- security 2-2
- system 2-2, 2-21 to 2-27
- system creation privileges,
 - granting 1-10, 2-5
- system creation privileges,
 - revoking 1-10, 2-6
- system read/write privileges 1-14, 2-21, 2-23

V

- verbose command (StP Utility) 4-3, 4-18

W

- weekdays (TI variable) A-8
- wildcard characters in StP Utility 4-5
- write group privileges 2-21, 2-22, 2-23

