Sybase® SQL Server™ Installation and Configuration Guide for Digital OpenVMS Alpha

Sybase SQL Server Release 11.0.x
Document ID: 34213-01-1102-01
Last Revised: November 11, 1996

Principal author: Server Publications Group

Contributing authors: Martin Ash, Ann Christina Rothchild

Technical Reviewers: Joel Brown, Brian Choy, Catherine Dimino, Joan Goodman

Document ID: 34213-01-1102

This publication pertains to Sybase SQL Server Release 11.0.x of the Sybase database management software and to any subsequent release until otherwise indicated in new editions or technical notes. Information in this document is subject to change without notice. The software described herein is furnished under a license agreement, and it may be used or copied only in accordance with the terms of that agreement.

Document Orders

To order additional documents, U.S. and Canadian customers should call Customer Fulfillment at (800) 685-8225, fax (617) 229-9845.

Customers in other countries with a U.S. license agreement may contact Customer Fulfillment via the above fax number. All other international customers should contact their Sybase subsidiary or local distributor.

Upgrades are provided only at regularly scheduled software release dates.

Copyright © 1989–1996 by Sybase, Inc. All rights reserved.

No part of this publication may be reproduced, transmitted, or translated in any form or by any means, electronic, mechanical, manual, optical, or otherwise, without the prior written permission of Sybase, Inc.

Sybase Trademarks

Sybase, the Sybase logo, APT-FORMS, Certified SYBASE Professional, Data Workbench, DBA Companion, Deft, First Impression, GainExposure, Gain *Momentum*, PowerBuilder, Powersoft, Replication Server, S-Designor, SQL Advantage, SQL Debug, SQL SMART, SQL Solutions, Transact-SQL, VisualWriter, and VQL are registered trademarks of Sybase, Inc. Adaptable Windowing Environment, Adaptive Server, ADA Workbench, AnswerBase, Application Manager, AppModeler, APT-Build, APT-Edit, APT-Execute, APT-Library, APT-Translator, APT Workbench, Backup Server, BayCam, Bit-Wise, Client-Library, Client/Server Architecture for the Online Enterprise, Client/Server for the Real World, Client Services, CodeBank, Column Design, Connection Manager, DataArchitect, Database Analyzer, DataExpress, Data Pipeline, DataWindow, DBA Companion Application Manager, DBA Companion Resource Manager, DB-Library, Deft Analyst, Deft Designer, Deft Educational, Deft Professional, Deft Trial, Designor, Developers Workbench, DirectCONNECT, Easy SQR, Embedded SQL, EMS, Enterprise Builder, Enterprise Client/Server, Enterprise CONNECT, Enterprise Manager, Enterprise SQL Server Manager, Enterprise Work Architecture, Enterprise Work Designer, Enterprise Work Modeler, EWA, Formula

One, Gateway Manager, GeoPoint, InfoMaker, InformationCONNECT, Intermedia Server, InternetBuilder, iScript, MainframeCONNECT, Maintenance Express, MAP, MDI, MDI Access Server, MDI Database Gateway, media.play, media.splash, MetaWorks, MethodSet, Movedb, Navigation Server Manager, Net-Gateway, Net-Library, New Media Studio, ObjectCONNECT, ObjectCycle, OmniCONNECT, OmniSQL Access Module, OmniSQL Server, OmniSQL Toolkit, Open Client, Open ClientCONNECT, Open Client/Server, Open Client/Server Interfaces, Open Gateway, Open Server, Open ServerCONNECT, Open Solutions, Optima++, PB-Gen, PC APT-Execute, PC DB-Net, PC Net Library, PowerBuilt, PowerBuilt with PowerBuilder, PowerScript, PowerSocket, Powersoft Portfolio, Power Through Knowledge, PowerWare Desktop, PowerWare Enterprise, ProcessAnalyst, Replication Agent, Replication Driver, Replication Server Manager, Report-Execute, Report Workbench, Resource Manager, RW-DisplayLib, RW-Library, SAFE, SDF, Secure SQL Server, Secure SQL Toolset, Security Guardian, SKILS, smart.partners, smart.parts, smart.script, SQL Anywhere, SQL Code Checker, SQL Edit, SQL Edit/TPU, SQL Remote, SQL Server, SQL Server/CFT, SQL Server/DBM, SQL Server Manager, SQL Server Monitor, SQL Server SNMP SubAgent, SQL Station, SQL Toolset, StarDesignor, Sybase Client/Server Interfaces, Sybase Development Framework, Sybase Gateways, Sybase Intermedia, Sybase Interplay, Sybase IQ, Sybase MPP, Sybase SQL Desktop, Sybase SQL Lifecycle, Sybase SQL Workgroup, Sybase Synergy Program, Sybase Virtual Server Architecture, Sybase User Workbench, SybaseWare, SyBooks, System 10, System 11, the System XI logo, SystemTools, Tabular Data Stream, The Architecture for Change, The Enterprise Client/Server Company, The Online Information Center, Turning Imagination Into Reality, Visual Components, VisualSpeller, Warehouse WORKS, Watcom, Watcom SQL, Watcom SQL Server, web.sql, web.works, WebSights, WebViewer, WorkGroup SQL Server, XA-Library, and XA-Server are trademarks of Sybase, Inc.

All other company and product names used herein may be trademarks or registered trademarks of their respective companies.

Restricted Rights

Use, duplication, or disclosure by the government is subject to the restrictions set forth in subparagraph (c)(1)(ii) of DFARS 52.227-7013 for the DOD and as set forth in FAR 52.227-19(a)-(d) for civilian agencies.

Sybase, Inc., 6475 Christie Avenue, Emeryville, CA 94608.

Table of Contents

About This Book

1. Introduction

2. Preparing for New Installation or Upgrades

3. Loading Software from Media

4. Installing SQL Server

5. Upgrading SQL Server

6. Installing Backup Server

7. Getting Started with SQL Server and Backup Server

8. Reconfiguring SQL Server and Backup Server

9. Administrative Tasks and Performance and Tuning

10. Migrating SQL Server from VAX to Digital OpenVMS Alpha

A. Recovery from Failure

B. *sybinit* Resource Files

C. Sample Sessions

D. Sample SQL Server System Specifications

E. Available Character Sets and Sort Orders

Index

List of Figures

List of Tables

About This Book

This guide provides instructions for:

- Preparing for New Installation or Upgrades
- Loading Software from Media
- Installing Sybase® SQL Server™
- Upgrading SQL Server
- Installing Backup Server™
- Getting Started with SQL Server and Backup Server
- Reconfiguring SQL Server and Backup Server
- Administrative Tasks and Performance and Tuning
- Migrating SQL Server from VAX to Alpha

➤ ***Note***

As used in this guide and other Sybase documents, release 11.0.x is a general term that refers to the series of 11.0.x releases. See the relevant *Release Bulletin* for release-specific issues.

Audience

This guide is for Sybase System Administrators or other qualified installers familiar with their system's environment, networks, disk resources, and media devices.

How to Use This Book

Table 1 provides a road map for how to proceed through the chapters in this guide.

Table 1: Road map to this guide

If You Are Doing This	Read or Follow the Instructions in This Chapter
Installing SQL Server and Backup Server for the first time	• Chapter 1, "Introduction" (for an overview of installation software and concepts) • Chapter 2, "Preparing for New Installation or Upgrades" • Chapter 3, "Loading Software from Media" • Chapter 4, "Installing SQL Server" • Chapter 6, "Installing Backup Server" • Chapter 7, "Getting Started with SQL Server and Backup Server"
Upgrading SQL Server and Backup Server	• Chapter 2, "Preparing for New Installation or Upgrades" • Chapter 3, "Loading Software from Media" • Chapter 5, "Upgrading SQL Server" • Chapter 6, "Installing Backup Server"
Installing Language Modules only	• Chapter 8, "Reconfiguring SQL Server and Backup Server"
Starting, monitoring, or shutting down SQL Server	• Chapter 7, "Getting Started with SQL Server and Backup Server"
Reconfiguring / customizing SQL Server and Backup Server	• Chapter 8, "Reconfiguring SQL Server and Backup Server"
Performing administrative tasks	• Chapter 9, "Administrative Tasks and Performance and Tuning"
Performing upgrades, VAX to Alpha	• Chapter 10, "Migrating SQL Server from VAX to Digital OpenVMS Alpha"

Incorporated *System Administration Guide Supplement*

This guide includes some subjects that were previously covered in the *System Administration Guide Supplement*.

For more information about system administration issues that are not covered in this manual, see the *System Administration Guide.*

Related Documents

Table 2 shows the documents you need for installing, setting up, and using SQL Server.

Table 2: Related documents

To Do This	See This Document
Install or upgrade SQL Server for Digital OpenVMS Alpha.	• *Installation and Configuration Guide for Digital OpenVMS Alpha* (this manual) • *Release Bulletin*
Configure SQL Server, Backup Server.	*Installation and Configuration Guide for Digital OpenVMS Alpha* (this manual)
Set up SQL Server to run on a network.	*Installation and Configuration Guide for Digital OpenVMS Alpha* (this manual)
Learn about OpenVMS Alpha-specific system administration issues.	*Installation and Configuration Guide for Digital OpenVMS Alpha* (this manual)
Use the utilities provided with SQL Server for OpenVMS Alpha, such as **isql** and **bcp**.	*SQL Server Utility Programs for Digital OpenVMS Alpha*
Learn about in-depth system administration issues for SQL Server, such as managing physical resources and user and system databases.	*SQL Server System Administration Guide*
Learn how to optimize queries, manage very large databases, handle disk and cache issues, and improve SQL Server performance.	*SQL Server Performance and Tuning Guide*
Learn how to use the security features in SQL Server.	*SQL Server Security Features User's Guide*
Learn how to administer a secure operating environment for SQL Server to control access to data; the manual includes information on adding users to the server, giving them controlled access to database objects and procedures, and managing remote servers.	*SQL Server Security Administration Guide*
Learn about the SQL Server commands, datatypes, functions, and system procedures.	*SQL Server Reference Manual*

Table 2: Related documents (continued)

To Do This	See This Document
Find a list of Transact-SQL® reserved words, definitions of system tables, a description of the *pubs2* sample database, a list of SQL Server error messages, and other reference information that is common to the SQL Server manuals.	*SQL Server Reference Supplement*
Learn how to use Transact-SQL, Sybase's enhanced version of the relational database language; the manual serves as a textbook for beginning users of the database management system.	*Transact-SQL User's Guide*
Find a synopsis of SQL commands.	*Quick Reference Guide for SQL Server Release 11.0*
See a diagram of the SQL Server system tables and their relationships.	*SQL Server System Tables Diagram*
Find out about the new features and changes in SQL Server release 11.0.x	*What's New in Sybase SQL Server Release 11.0?*
Find last-minute information about SQL Server release 11.0.x	*Release Bulletin*

Style Conventions

The following style conventions are used in this manual:

- In a sample screen display, OpenVMS commands are shown in a bold Courier font:

  `this font`

- In a sample screen display, words you should replace with the appropriate value for your installation are shown in a bold italic Courier font:

  `this font`

- In the regular text of this document, the names of files and directories appear in italics:

  [REL110.SYBASE]

- The names of programs, utilities, procedures and commands appear in bold Helvetica type:

  sybinit

SQL Syntax Conventions

SQL syntax statements display the syntax and all options for a command. They are printed like this:

sp_dropdevice [*device_name*]

Examples showing the use of Transact-SQL commands are printed like this:

```
select * from publishers
```

Examples of output from the computer are printed like this:

```
pub_id  pub_name                city        state
------- ----------------------  ----------- -----
0736    New Age Books           Boston      MA
0877    Binnet & Hardley        Washington  DC
1389    Algodata Infosystems    Berkeley    CA

(3 rows affected)
```

The conventions for syntax statements in this manual are as follows:

Table 3: SQL syntax conventions

Key	Definition
command	Command names, command option names, utility names, utility flags, and other keywords are in bold helvetica.
variable	Variables, or words that stand for values that you fill in, are in italics.
{ }	Curly braces indicate that you choose at least one of the enclosed options. Do not include braces in your option.
[ ]	Brackets mean choosing one or more of the enclosed options is optional. Do not include brackets in your option.
( )	Parentheses are to be typed as part of the command.
|	The vertical bar means you may select only one of the options shown.

Table 3: SQL syntax conventions (continued)

Key	Definition
,	The comma means you may choose as many of the options shown as you like, separating your choices with commas to be typed as part of the command.

If You Need Help

If you have questions that are not answered in this guide or related documents, you can refer to the online help in the **sybinit** utility or the online help described in "Online Syntax Help: sp_syntax" on page 7-6. For more information about specific operating system commands, see your Digital OpenVMS Alpha documentation.

You can also contact Sybase Technical Support. Each Sybase installation that has purchased a support contract has one or more designated people who are authorized to contact Sybase Technical Support.

What's Next?

Chapter 1, "Introduction," discusses concepts about Sybase software that you should understand before beginning the tasks described in later chapters, and introduces new features for SQL Server release 11.0.x.

1 Introduction

This chapter introduces important concepts about SQL Server software, provides conceptual overviews important to installing, upgrading, and localizing, and describes the new features in SQL Server release 11.0.x that affect installation and upgrade. Topics covered in this chapter are:

➤ ***Note***

See the *Release Bulletin* for last minute changes to this release of SQL Server that are not covered in this manual.

System Requirements for SQL Server Release 11.0.x

The following table shows the system requirements for running SQL Server release 11.0.x:

Table 1-1: System requirements for running SQL Server

Operating System:	OpenVMS 6.2
Memory:	30,000 pagelets
SQL Server:	280,000 disk blocks

"Installing" Compared to "Configuring"

In previous Sybase releases, the utility **VMSINSTAL** was used both to copy files from the distribution media and to set up the Sybase software. In this release, **VMSINSTAL** is used only for copying files from media. All other functions, including setting up a new SQL Server and upgrading an existing SQL Server to release 11.0.x, now use the Sybase utility **sybinit.**

To reflect this change, this guide uses the term "install" to mean copying files from the distribution media, and "configure" to mean setting up a new SQL Server.

➤ ***Note***

To create a new SQL Server, first install the Sybase files with **VMSINSTAL**, and then configure SQL Server with **sybinit**. To upgrade a SQL Server, install the new release's files with **VMSINSTAL**, and then upgrade SQL Server with **sybinit**.

SQL Server User Roles

The SQL Server installation and configuration process defines various user roles. Different user roles have different responsibilities and privileges.

- **Operating system administrator** – maintains the operating system. This person has privileges that are granted to the "SYSTEM" account.
- **Sybase System Administrator** – executes **sybinit** and performs SQL Server configuration and upgrade tasks. The "SYBASE" account on the operating system belongs to this person

About *VMSINSTAL*

Sybase software is installed using the OpenVMS software installation utility, **VMSINSTAL**. **VMSINSTAL** prompts for information recorded on the worksheet, such as the Sybase products to install and the Customer Authorization String (CAS). Once you've supplied all the necessary information, **VMSINSTAL** proceeds automatically.

➤ ***Note***

The "SYSTEM" user for your operating system executes **VMSINSTAL**. See "Defining SYBASE_SYSTEM" on page 2-1 for information on preparing this account to execute **VMSINSTAL**.

About *sybinit*

Use the **sybinit** utility to configure SQL Server and other Sybase products. Begin a **sybinit** session after you have loaded SQL Server software with **VMSINSTAL.** Use **sybinit** interactively or with a resource file. See Chapters 4, 5, 6 and 8 for information about using **sybinit** interactively. See Appendix B, "sybinit Resource Files" for information about using **sybinit** with a resource file.

Sybase System Structure

VMSINSTAL creates the Sybase system structure based on your definition for the logical name SYBASE_SYSTEM. The system structure is illustrated in Figure 1-1. It is organized by function rather than by product.

Installing SQL Server creates only the part of the system structure required for SQL Server software. When installing client software on a machine on which SQL Server has already been installed, **VMSINSTAL** adds to the existing SQL Server directories the components that have the same function. **VMSINSTAL** adds the command (*.COM*) and executable (*.EXE*) files it needs for installation in *[SYBASE.INSTALL].* Other *.COM* and *.EXE* files are added to directories like *[SYBASE.BIN].* New directories (for example,

[SYBASE.TERMDEF] for terminal definitions) are created for client-specific functions as required.

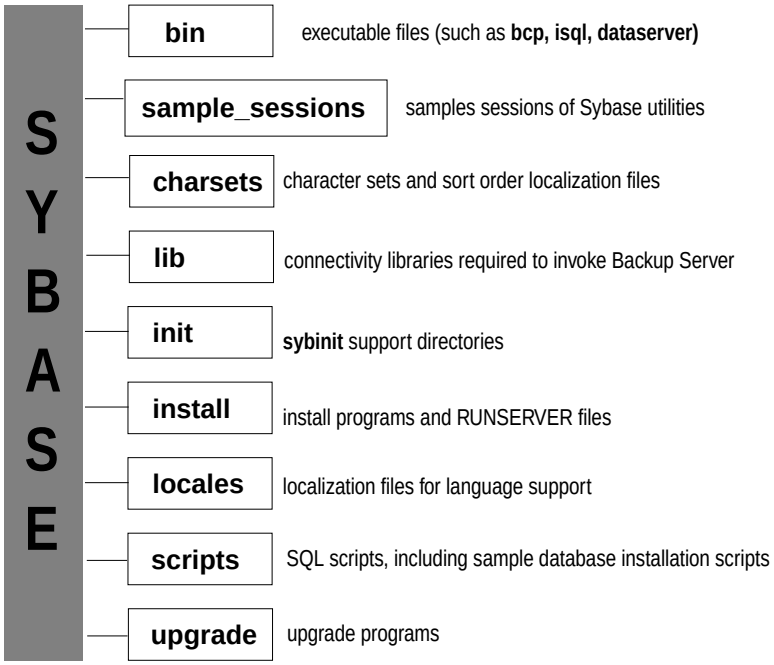


Figure 1-1: Sybase SQL Server installation structure

➤ ***Note***

If you are upgrading SQL Server, you need two Sybase release directories: the existing release directory, containing your current SQL Server installation, and the new release directory, where you will load your new Sybase software. The new release directory will contain your upgraded SQL Server.

Backup Server

In SQL Server release 10.0 and later, all dump and load operations use Backup Server. You must install a Backup Server if you plan to dump or load databases on SQL Server release 11.0.x. Use **sybinit** to

install a Backup Server as a part of a complete SQL Server configuration session.

The Interfaces File

SQL Server communicates with other SQL Servers, Open Server applications, and client software on your network. Clients can talk to one or more servers, and servers can communicate with other servers by remote procedure calls. A client cannot connect with a SQL Server if the server does not support the client's language or character set.

For Sybase products to interact with one another, each product needs to know where the others reside on the network. This information is stored in an interfaces file, usually named *interfaces*, located in the SQL Server installation directory. Once your SQL Server or client software is installed, it can connect with any server on the network as long as the server is listed in the interfaces file.

The interfaces file is like an address book. It lists the name and address of every known server (including Backup Server). When you use a client program and need to connect with a particular server, the client program looks up the server name in the interfaces file and connects to that server (see Figure 1-2). You can supply the name of the server by specifying the DSQUERY logical name or through a command line qualifier.

The port number (for TCP/IP networks) or object (for DECnet networks) lets clients identify the SQL Server to which they want to connect. It also tells SQL Server where to listen for incoming connection attempts from clients. SQL Server uses a single number for these two services (referred to as "query service" and "listener service," respectively).

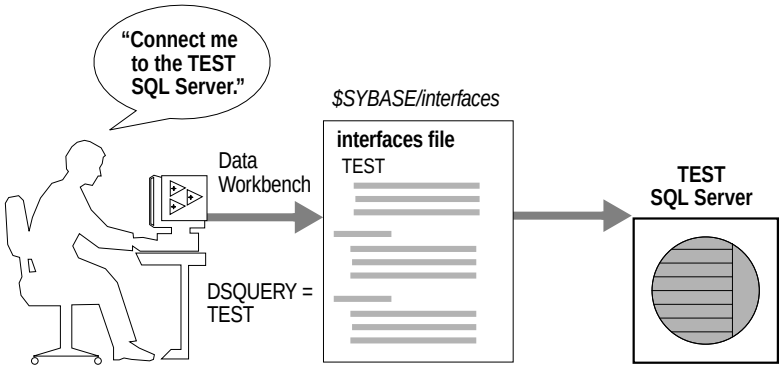


Figure 1-2: Communicating with a SQL Server

During the installation process, you will use **sybinit** to create entries in the interfaces file for your new SQL Server and Backup Server. Later on, you can use **sybinit** to modify the interfaces file. See "Using sybinit to Create or Edit Interfaces File Entries" on page 9-9 for more information.

How SQL Server Communicates with Clients

When client (front-end) software, or a server acting as a client by executing remote procedure calls (RPCs), needs to communicate with SQL Server, the client:

- Determines the name of the SQL Server, usually from the value of the logical name DSQUERY or from the value given on the command line.
- Looks for an entry in the interfaces file that has a SQL Server name matching the value of DSQUERY or the value of the command line.
- Connects to the SQL Server running on the address listed for that entry. If the client is unable to connect the first time, it makes additional attempts according to the delay and retry numbers indicated in the interfaces file. If no matching entry is found, an error message is written to the client's standard error file.

The client can connect to any SQL Server correctly listed in the interfaces file, as long as the matching SQL Server name is supplied via DSQUERY or a command line qualifier. If no server name is supplied, the software looks for an interfaces file entry with the default server name, SYBASE. You can use this feature if you always connect to a single SQL Server named SYBASE.

If you plan to use more than one SQL Server and to refer to all of them in the same interfaces file, choose a different name for each SQL Server. For clarity, you should choose different names for each SQL Server connected to your network. You may want to choose SQL Server names that describe your applications to help you remember them.

◆ ***WARNING!***

Do not use SQL Server names that are the same as any computer name on your network. Using the same name for SQL Server and your computer can cause serious problems on some networks.

Sybase Devices

SQL Server stores data on SQL Server virtual devices. You can create SQL Server virtual devices on OpenVMS file-oriented devices such as RMS files. You may need to configure the following devices during your **sybinit** configuration session:

- Master device, to store system databases
- *sybsystemprocs* device, to store system procedures
- Auditing device, to store the auditing database and auditing system procedures

The **sybinit** utility initializes the master device, *sybsystemprocs* device, and the *sybsecurity* device for SQL Server during the installation session.

You can use the **disk init** command to initialize additional devices after your SQL Server installation is complete. See the *System Administration Guide* for detailed information on **disk init**.

Table 1-2: Master device databases provides information on the contents of the master device. For more information on the *sybsystemprocs* and

auditing devices, refer to "New Features for Release 11.0.x" on page 1-12.

Table 1-2: Master device databases

Database Name	Size	Function
master	8MB min.	Contains information about users, databases and objects, and system catalog entries.
model	2MB	"Template" database with catalog entries.
tempdb	2MB min.	Work area for SQL Server. Cleared and rebuilt each time SQL Server is started.
pubs2 *interpubs* *jpubs*	2MB 2MB 2MB	Optional sample database in U.S. English, Western European, and Japanese versions; used in Sybase documentation examples.

Master Device

The master device file stores vital information about the databases you create on your system (user databases). This information is stored in the three system databases, *master*, *model*, and *tempdb*.

You may optionally create sample databases in the master device after successfully installing SQL Server.

See "Sample Databases" on page 7-9 for information on installing the sample databases.

➤ ***Note***

You cannot expand the *master* database beyond the initial capacity of the master device file. For this reason, you should determine your future needs for the master device before installing SQL Server.

Master Device Contiguity

sybinit also provides an option for you to specify whether the master device is contiguous. Your selection of "yes" for this option forces database creation for master to be on contiguous blocks of a disk. If the **buildmaster** utility (which is called by **sybinit)** is unable to create a contiguous file, it fails with an error message. If you select "no" for this option, the **buildmaster** utility (called by **sybinit**) still tries to create a

contiguous file for the master device, but if it fails, it creates a file that does not force contiguity.

sybsystemprocs Device

sybsystemprocs contains most of the system-stored and user-created procedures that were stored in the *master* database in pre-System 10™ SQL Server.

Configure a device for *sybsystemprocs* during your installation or upgrade session. The system procedures needed during difficult recovery situations remain in *master*. Your own locally created system procedures also remain in *master*, unless you move them after your upgrade is complete.

See Table 2-4 on page 2-15 for information about the recommended size for the *sybsystemprocs* device.

sybsecurity Device

The *sybsecurity* auditing system records system security information in a SQL Server audit trail. Use this audit trail to monitor the use of SQL Server or system resources. The audit system consists of the *sybsecurity* database and a set of system procedures with which you can configure auditing for your system. *sybsecurity* is created as part of the auditing installation process. It contains all the system tables found in the *model* database and two additional system tables:

- *sysaudits* – the audit trail
- *sysauditoptions* – the global auditing choices

For a new installation, you can activate auditing at the same time you install SQL Server or during a later reconfiguration session. For an upgrade, you can activate auditing during a reconfiguration session after your upgrade is complete. You should install the *sybsecurity* database on its own device and the *sysaudits* table on its own segment on that device. Use the threshold manager to monitor the available space in *sysaudits*.

See Table 2-3 on page 2-14 for information about the recommended size for the *sybsecurity* device.

Sybase Globalization Support

Sybase provides the following support for international systems:

- **Data support** – The Sybase Character Sets product contains character set and sort order definition files that SQL Server uses to process the characters used in different languages. The Sybase Character Sets product provides data processing support for the major languages in Western Europe, Eastern Europe, the Middle East, Latin America, and Asia.
- **Translated system messages** – SQL Server release 11.0.x has language modules for French, German, and Spanish.
- **Translated documentation** – Translated documentation is available in Chinese (Simplified), French, German, Japanese, and Spanish.

By default, SQL Server and Backup Server have the following:

- Character set definition files for Western European languages
- Sort order definition files for Western European character sets
- U.S. English system messages

By default, **sybinit** configures SQL Server on Digital OpenVMS Alpha with the ISO 8859-1 character set, binary sort order, and U.S. English messages. You can specify different values during installation.

To support a language other than a Western European language, or if you want system messages in another language, you must purchase one of the available Language Modules or the Sybase Character Sets product.

For more information on Sybase's Globalization support, see Appendix E, "Available Character Sets and Sort Orders," of this book, and Chapters 12 and 13 in the *System Administration Guide*.

Language Modules

Language Module features are contained in localization files under the *[SYBASE.LOCALES]* and *[SYBASE.CHARSETS]* directories. When you install a Language Module, the language, character set, and sort order files to support the new language are automatically loaded into the Sybase directory tree in the correct locations. *Figure 1-3: Localization files in the Sybase directory tree* illustrates the structure of these localization files. This figure does not show complete lists of the files in all of the subdirectories.

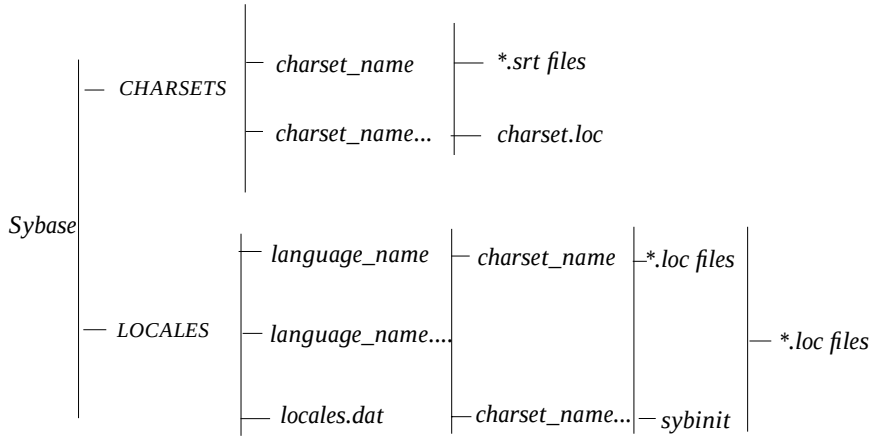


Figure 1-3: Localization files in the Sybase directory tree

locales Directory

In addition to the *locales.dat* file, the *[SYBASE.LOCALES]* directory contains a subdirectory for each available language. Each language subdirectory contains a subdirectory for each character set available with that language. The files in these subdirectories, *.loc* files, enable SQL Server or Backup Server to report errors in a specific language encoded in a specific character set. There are a variety of *.loc* files in each subdirectory. Most of these files contain translated error messages for a specific product or utility. The *common.loc* file in each subdirectory contains localized information used by all products, such as local date, time, and money formatting.

See “The locales.dat File” on page 8-4 and the *System Administration Guid*e for more information about the *[SYBASE.LOCALES]* directory structure and files.

Sample Language Databases

The sample language databases are *interpubs*, which is used with the French and German language modules. These scripts are located in the e *[SYBASE.SCRIPTS]* directory. See “Sample Databases” on page

7-9 for more information about these scripts and instructions for installing sample databases.

Charsets Directory

Files in the *[SYBASE.CHARSETS.CHARSET_NAME]* subdirectory contain information related to particular character sets, such as the definition of the character set and any sort orders available for that character set.

New Features for Release 11.0.x

For information about new functionality available with this release, see *What's New in Sybase SQL Server 11.0.x?*

What's Next?

Chapter 2, "Preparing for New Installation or Upgrades" provides instructions for preparing new configurations and upgrades of SQL Server. As you perform the tasks in Chapter 2, complete the SQL Server Configuration / Upgrade Worksheet that begins on the next page.

SQL Server Configuration/Upgrade Worksheet

Fill out this worksheet and refer to this information as you read through Chapters 2–7. Keep the completed worksheet in a safe place for future reference. You may need the information on it if you call Technical Support, and it will save you time if you have to reconfigure your installation for any reason.

Chapter 2, "Preparing for New Installation or Upgrades"	
Information for **VMSINSTAL**	SYBASE_SYSTEM definition for new SQL Server installation: _____ SYBASE_SYSTEM definition for existing SQL Server installation (upgrades only): _____ Type of installation (Circle one): Primary Secondary CD_ROM drive device name (CD-ROM installations only): _____ Tape drive device name (global tape installations only): _____ Customer Authorization String: _____
"SYSTEM" Account	Logged in as "SYSTEM" Minimum quotas set for "SYSTEM"
"SYSTEM" Account	Network Active DECnet ❒ TCP ❒

SQL Server Interfaces File Information	DECnet: Network node name: _____ Network object: _____ Alias (Optional): _____ TCP: Host address: _____ Port number: _____ Alias (Optional): _____ Retry count (Optional): _____ Retry delay (Optional): _____
Backup Server Interfaces File Information	DECnet: Network node name: _____ Network object: _____ Alias (Optional): _____ TCP: Host address: _____ Port number: _____ Alias (Optional): _____ Retry count (Optional): _____ Retry delay (Optional): _____
Nodes to Be Used by SQL Servers and Clients	Nodes: _____ _____ _____ _____ _____
Companion Server Information (OpenVMS Cluster Environments Only)	Network proxy set up

"Sybase" User Account	Login directory = Sybase Default privileges: [(ALTPRI), DETACH, GRPNAM, NETMBX, SYSLCK, SYSNAM, TMPMBX] Default privileges for SQL Server debug option: [SHARE, SYSPRIV] Identifier created for "Sybase"
OpenVMS Resources	System has at least 18,000 free blocks in pagefile(s) WSMAX is at least 30,000 pagelets (current value): System has at least 30,000 free pagelets (value of GBLPAGES): System has at least 2 free global sections (Value of GBLSECTIONS): GBLPAGFIL is at least 2500 pages (current value):
Master Device Information (New Configurations)	Size for device in megabytes (default is 24MB): Contiguous Device (circle one): Yes No Directory created and protections set (circle one): Yes No Directory Specification (including device):
sybsystemprocs Device Information	Logical Device Name (default is *sysprocsdev*): _____ Size of *sybsystemprocs* database (default is 21MB): _____ Size for Device in Megabytes (default is 21MB): _____ Directory created and protections set Directory Specification (including device): _____

Auditing Device Information	Logical device name (default is *sybsecurity*): _____ Size of *sybsecurity* database (default is 5MB): _____ Size for device in megabytes (at least 5MB): _____ Directory created and protections set Directory specification (including device): _____
Chapter 3, "Loading Software from Media"	Values for *locale_name* in *locales.dat* file entries match operating system definitions.
Chapter 4, "Installing SQL Server"	
SQL Server Name	Name of SQL Server (11 characters maximum; default is Sybase): _____
Backup Server Name	Name of Backup Server (default is SYB_BACKUP): _____
SQL Server Error Log Location	File specification for SQL Server error log: (default is *SYBASE_SYSTEM:[SYBASE.INSTALL]ERRORLOG*): _____
Language Information	SQL Server default language: _____ Additional SQL Server languages: _____
SQL Server Character Set Information	SQL Server default character set: _____ Additional character sets to install (circle all appropriate character sets): cp850 cp437 roman8 mac
SQL Server Sort Order Information	Sort order for SQL Server: _____

SQL Server Auditing information recorded above.	
SQL Server Quota File Location	File specification for SQL Server quota file: _____
Chapter 5, "Upgrading SQL Server"	Users logged off. checkpoint and dbcc commands executed on existing databases and errors corrected. sp_checkreswords installed and run on existing databases and reserved word conflicts eliminated. Existing databases backed up. Database objects re-created Location of runserver file verified. Runserver file contains *SYBASE_SYSTEM* definition for existing SQL Server installation. Verifing the size of *sybsystemprocs*. Removed database devices from *SYBASE_SYSTEM*. See "*sybsystemprocs* Device Information" and "Auditing Device Information" under the "Chapter 2" heading for information on sybsystemprocs and auditing options.
Chapter 6, "Installing Backup Server"	Name for Backup Server (maximum of 11 characters) (default is SYB_BACKUP): _____ File specification for Backup Server error log: (default is *SYBASE_SYSTEM:[SYBASE.INSTALL]BACKUP.LOG*): _____ See "Backup Server Interfaces File Information" under the "Chapter 2" heading for Backup Server interfaces file entry values. File specification for Backup Server quota file: _____ Backup Server language: _____ Backup Server character set:_____

Chapter 8, “Reconfiguring SQL Server and Backup Server,”	Changes for SQL Server Default Backup Server to use: SQL Server default language: Additional SQL Server languages: SQL Server default character set: Additional SQL Server character sets: SQL Server sort order: See “Preparing Auditing Device” under “Chapter 2” heading for information on auditing options. Changes for Backup Server Backup Server language: Backup Server character set:

2 Preparing for New Installation or Upgrades

This chapter provides detailed instructions for adjusting your environment before you begin a new installation or upgrade of SQL Server. Topics covered in this chapter are:

➤ ***Note***

Although this manual includes VMS system instructions, see your VMS documentation for more information about (or changes to) these instructions.

Defining *SYBASE_SYSTEM*

The logical name *SYBASE_SYSTEM* defines the location of the SQL Server 11.0.x installation. **VMSINSTAL** prompts you for this definition during the installation process.

SYBASE_SYSTEM may be defined either as a physical device (for example, *DUA0:*) or as a rooted and concealed logical name that points to a directory specification. In executive mode, *SYBASE_SYSTEM* must also be defined by adding the **EXECUTIVE** qualifier to the **define** command. Additionally, if you're running on a VMS cluster, *SYBASE_SYSTEM* must point to a location that is visible cluster-wide.

For example, if *DUA0:[REL110]* is the desired location for the installation, define the logical name to point to that location. This logical name will then be used to define *SYBASE_SYSTEM* during

the **VMSINTAL** session. The following example uses the logical name *SYBASE_11*:

```
$ DEFINE/SYSTEM/EXECUTIVE/TRANSLATION=CONCEALED -
_$ SYBASE_110 DUA0:[REL110.]
```

1. Determine the *SYBASE_SYSTEM* definition for your new SQL Server installation.

   If you plan to upgrade your SQL Server from an earlier release, *SYBASE_SYSTEM* must be defined to a different location than your current installation. The upgrade process uses both the old and new software during upgrades.

2. Record the *SYBASE_SYSTEM* definition for your new SQL Server installation on the worksheet. If you are upgrading, record the *SYBASE_SYSTEM* definition of your current system.

◆ ***WARNING!***

In a VMScluster environment, you cannot upgrade a SQL Server located on a different cluster than the cluster where your new software is installed. Be sure to define *SYBASE_SYSTEM* so that your new Sybase software is installed on the same cluster as the SQL Server you want to upgrade.

Determining Installation Type

VMSINSTAL asks you whether your SQL Server is a primary or secondary installation. It is possible to have more than one Sybase product lines coexisting on the same system. A primary (default) installation is intended to be used as the production environment. A secondary installation, on the other hand, is intended to be used as a test environment, or a second primary system.

The main difference between the two types of installations is the way Sybase logicals are defined. Logical names defined by the Sybase command file *SYBASE_SYSTEM:[SYBASE.INSTALL]SYLOGICALS.COM* are defined with the **/SYSTEM** qualifier in a primary system. In a secondary installation, these logicals are defined at a process level. This allows the same Sybase logical names to be used without overwriting the primary Sybase logicals. It is up to each user of the secondary system to execute the secondary installation's

SYLOGICALS.COM command file. It is also suggested that these users place the following command in their *LOGIN.COM* file:

```
@SYBASE_SECONDARY_PATH:[SYBASE.INSTALL]SYLOGICALS.COM
```

where *SYBASE_SECONDARY_PATH* is the logical name or device entered as the *SYBASE_SYSTEM* file specification during installation of the secondary system.

If the *SYBASE_SECONDARY_PATH* is a logical name, it can be defined in *SYS$MANAGER:SYLOGICALS.COM* (along with other system logicals).

1. Determine whether your SQL Server is a primary or secondary installation.
2. Record the correct installation type on the worksheet by circling "Primary" or "Secondary."

Recording Other Information

1. Record the name of the source device on your worksheet.

   You must include the name of the device from which the utility copies the Sybase software during your **VMSINSTAL** session. This device is the drive where you place your global tape or CD-ROM.

2. Record your Customer Authorization String (CAS) on the worksheet.

   The CAS is a 34-character alphabetic string that is used to identify the products you have purchased, that you can install and configure. It is printed on the distribution media label. **VMSINSTAL** prompts you for the CAS when the utility copies files from the distribution media.

➤ ***Note***

If you are installing Language Modules only, skip to "Creating a "SYBASE" User Account" on page 2-10.

Verifying Networking Software

Your network configuration must be correct for you to perform the tasks in this guide. **Sybase software uses the network even when a client program is connected to a SQL Server on the same machine**.

Verify the status of each network you plan to use, as explained on the following pages. SQL Server, Backup Server support the DECnet and TCP protocols.

You cannot configure a server until your network protocol is functional. Below are tests to help you verify the operation of your network. If any of these tests fails, your network is not functioning properly.

Verifying DECnet

1. Verify that DECnet is installed on your machine. From the OpenVMS prompt, type:

   ```
   $ SHOW LOGICAL SYS$NODE
   ```

   If this logical name is defined, DECnet has been installed.

2. Verify that DECnet is running and configured properly. From the OpenVMS prompt, enter the following command:

   ```
   $ SHOW NETWORK
   ```

   If DECnet is running, OpenVMS displays some information about the status of the network. If the network is unavailable, OpenVMS responds with the message:

   ```
   % SHOW-I-NONET, network unavailable
   ```

3. If you are installing on a SQL Server-only machine, verify that you can connect to another machine by typing the following from the OpenVMS prompt:

   ```
   $ SET HOST another_node_name
   ```

   Verify that you can log in properly, then log out.

4. If your machine is not part of a network, verify that you can make a loopback connection via DECnet. From the OpenVMS prompt, type:

   ```
   $ SET HOST 0
   ```

   If the network is properly configured, you receive a login prompt from the machine that you are currently using. Verify that you can log in properly, then log out.

5. Record the status of the DECnet protocol for your network on the worksheet.

Verifying TCP

On a TCP network, use **telnet** to verify your network connections.

1. If you are installing on a SQL Server-only machine, verify that you can connect to another machine. From the OpenVMS prompt, type:

   ```
   $ TELNET another_node_name
   ```

   Verify that you can log in properly, then log out.

2. If your machine is not on a network, verify that you can make a loopback connection by typing the following from the OpenVMS prompt:

   ```
   $ TELNET 127.0.0.1
   ```

   If the network is properly configured, you receive a login prompt from the machine that you are currently using. Verify that you can log in properly, then log out.

3. Record the status of the TCP protocol for your network on the worksheet.

Recording Interfaces File Information

When executing remote procedure calls to a SQL Server, Backup Server, client programs and SQL Servers locate that server with:

- DECnet – a combination of node name and network object
- TCP – a combination of host address or node name and port number

In addition, on networks that allow multiple protocols, you can use an alias (a “nickname”) to distinguish one listener service from another.

To create an interfaces file entry when you configure a new server, you provide information in the form required by each network protocol you plan to use.

See Chapter 9, “Administrative Tasks and Performance and Tuning” for more information on the interfaces file.

Recording Network Address Information

If you plan to use DECnet, record the node names for SQL Server Backup Server on your worksheet.

You can use network utilities to find an IP address of the machine on which you plan to configure a server. You may need CMKRNL and SYSPRV privilege(s). The following examples show how to find the IP address of the node "gummo" when logged in to MultiNet TCP or Wollongong's Pathway TCP networks.

Finding Node Names on DECnet

To find the network address and node name on DECnet, type:

```
$ show network
```

In the first line of the output from this command, OpenVMS displays the network address and node name. For example:

```
OpenVMS Network status for local node 1.53 WHITNEY on 8-JUN-1992
11:31:13.72
```

Finding Node Addresses on MultiNet

The following example shows how to find the Internet address of node "gummo" when logged in to MultiNet tcp network:

```
$ multinet ping gummo
```

OpenVMS displays:

```
PING GUMMO.SYBASE.COM (131.214.1.59) : 56 data
bytes
64 bytes from 131.214.1.59: icmp_seq=0 time=20ms
64 bytes from 131.214.1.59: icmp_seq=1 time=20ms
```

The Internet address appears in parentheses. Press Ctrl-y to stop the infinite loop.

Finding Node Addresses on WIN/TCP

The following example shows how to find the Internet address of node "gummo" when logged in to a WIN/TCP network:

```
$ search twg$etc:[000000]hosts. gummo
```

OpenVMS displays:

```
131.214.1.63    gummo   #ENG    PRV    NO    NO
```

The dotted Internet address appears at the left. If *TWG$ETC:[000000].* does not exist, or does not reference the appropriate node, type:

```
$ show symbol ping
```

If "ping" is defined, this response appears:

```
PING == "$TWG$TCP:[NETDIST.ETC]PING PING"
```

If it is undefined, then type:

```
$ PING == "$TWG$TCP:[NETDIST.ETC]PING PING"
$ ping gummo
```

The dotted Internet address appears:

```
64 bco

bytes from 131.214.1.59: icmp_seq=0 time=20ms
64 bytes from 131.214.1.59: icmp_seq=1 time=20ms
```

Press Ctrl-y to stop the infinite loop.

Digital UCX

To find the network address and node name on Digital UCX, enter:

```
UCX> ping/all (<cntrl-C to exit)
```

Make sure the inet (pseudo) driver is loaded.

Execute the command:

```
@$SYS$MANAGER:UCX$LOAD_INETDRIVER
```

Recording Object or Port Number

When SQL Server or Backup Server starts, it uses the number you provide to establish a network object.

Table 2-1 describes the two types of ports, or network objects, found in the interfaces file entries.

Table 2-1: Object/port types and their uses

Object/Port Type	Use	Protocol
Query	Used by client to request connections	DECnet, TCP
Master	Used by SQL Server to listen for connection requests from clients	DECnet, TCP

Remember that an object or port number must be unique to the network node on which you are running SQL Server or Backup Server. If you use Companion Servers, you must also ensure that the

object number is unique for all nodes in the VMScluster which will be running SQL Servers.

➤ ***Note***

For SQL Server, object names are not recommended. Sybase supports network object names for compatibility with previous releases. If you use an object name when running SQL Server on DECnet, Companion Server may have problems starting. For Backup Server, use named objects rather than object numbers.

Table 2-2 describes the port number ranges for various network protocols:

Table 2-2: Port number ranges for network protocol

Network Protocol	Port number range
Decnet	128 – 253
TCP / IP	5000 – 65535

1. Verify that the network objects or port numbers you plan to use for SQL Server and Backup Server are unique.

   You can verify in DECnet that an object is unique with the following commands:

   ```
   $ MCR NCP
   NCP>SHOW KNOWN OBJECTS
   NCP>EXIT
   ```

   A list of the network objects currently in use is displayed. Check to see that the object you have chosen does not appear.

   This test displays only the current DECnet network objects. If a product using the same network number starts before SQL Server, you cannot start SQL Server.

   To determine the uniqueness of a port number on a TCP network, see your network documentation.

2. Record the network object number or port number for SQL Server on your worksheet.
3. Record the object name or port number for Backup Server on your worksheet.

Recording Other Interfaces File Information

1. You can use an alias ("nickname") for any listener service in an entry in the interfaces file, as an easy way of distinguishing one listener service from another.

   If you want to use an alias for a DECnet or TCP listener service for SQL Server and Backup Server, record the aliases on your worksheet.

2. **sybinit** prompts you to enter a value for the "retry" and "delay" parameters. These numbers are added to the interfaces file, and determine how many times a client program or a SQL Server executing a remote procedure call try to connect with a a SQL Server, and how many seconds a client program waits between tries. The default for both "retry" and "delay" is zero. Note that remote procedure calls (RPCs) do not use the "delay" parameter.

   If you want to use the "retry" and "delay" parameters for SQL Server or Backup Server, enter values on your worksheet.

3. Make a list of all nodes on which any combination of SQL Servers and Companion Servers (VMScluster environments only) will run, and record this list on your worksheet.

Setting Up a DECnet Proxy

➤ ***Note***

This section applies only to companion servers

When you configure the Companion Server (Cluster Fault Tolerant, or CFT), the user account that you use to log in (which owns the configured objects) must be usable on each node on which an active server or Companion Server will run. In addition, you must set up a DECnet proxy for the "SYSTEM" and Sybase accounts on each node running a Companion Server.

Your proxy should map the owner account on each node on which an active or Companion Server resides to the account on the remote node. If you will be starting a Companion Server at system boot time, your proxy must also map the "SYSTEM" account on which the active or Companion Server resides to the "SYSTEM" account on the remote node. If the proxy does not contain these mappings, you cannot start a server on a remote node from the local node.

For example, if you had a SQL Server named Chicago that boots up on DECnet node Blue, and two companion SQL Server named New York and San Francisco, the two companion servers would have the following entries in the

For more information about the network proxy, see the authorization utility documentation for your operating system.

After you have set up a network proxy, check off “Network Proxy Set Up” on your worksheet

Creating a “SYBASE” User Account

VMSINSTAL sets protections on Sybase files to the “SYBASE” user. In order to avoid permissions problems, the “SYBASE” user should perform all **sybinit** tasks described in this guide. This user needs an account with the following properties:

1. If an account with the following properties is not currently available, create one now:
   - Default resource quotas
   - Default privileges:

     DETACH

     GRPNAM

     NETMBX

     SYSLCK

     SYSNAM

     TMPMBX
   - Login directory of *[SYBASE]*
   - Login device of *SYBASE_SYSTEM*:

   If SQL Server must run at a higher priority than the default priority for the account, ALTPRI must also be given to the account.

   It is highly recommended that the account also have SYSPRV and SHARE privileges, so that the DEBUG option can be activated on a running SQL Server if problems occur.
2. To create the account, use AUTHORIZE. The account privileges need to be set up as default and authorized privileges. For example:

```
$ SET DEFAULT SYS$SYSTEM
$ RUN AUTHORIZE
UAF> add SYBASE /uic=[101,101]/device=SYBASE_SYSTEM:/directory=[SYBASE] -
_UAF> /defprivilege=(SYSNAM,DETACH,GRPNAM,SYSLCK,ALTPRI,SYSPRV,SHARE) -
_UAF> /privilege=(SYSNAM,DETACH,GRPNAM,SYSLCK,ALTPRI,SYSPRV,SHARE) -
_UAF> /PGFLQUOTA=30000 /ADD_IDENTIFIER
%UAF-I-ADDMSG, user record successfully added
%UAF-I-RDBADDMSGU, identifier SYBASE value: [101,101] added to
RIGHTSLIST.DAT
UAF> EXIT
```

This example creates an account with the *UIC [101,101]* with a default device equivalent to the logical name *SYBASE_SYSTEM*. You can use any UIC for this user account.

3. Issue following command from the VMS authorize utility to ensure that an identifier has been created for SYBASE and added to the rights list:

   `UAF> SHOW /IDENTIFIER SYBASE`

   Sybase should be one of the identifiers displayed. If it is not, contact your system administrator for assistance if SYBASE is not displayed.

   After you have verified that the Sybase identifier has been created, check off this item on the worksheet.

4. When you create this user account, set its default directory to *[SYBASE]*.

5. On the worksheet, check off “Login directory = SYBASE”. Also check off the user account privileges.

Ensuring Availability of OpenVMS Resources

The availability of OpenVMS Resources is critical to the SQL Server installation. Be sure to check all items listed in the following text.

SQL Server uses about 30,000 OpenVMS virtual memory pagelets in its default configuration. (See Chapter 8, “Reconfiguring SQL Server and Backup Server”the for information on how to increase this amount.) Sybase recommends that the working set for the SQL Server process be at least 30,000 pagelets. Running SQL Server in a smaller working set could result in substantially reduced performance.

Avoid giving SQL Server a working set larger than the amount of free memory, as this may result in poor performance. Follow the guidelines in your OpenVMS documentation.

OpenVMS memory pages are 8192 bytes each. Pagelets are 512 bytes each.

System File Paging Space

You must have at least 18,000 free blocks of page file space available or you cannot start SQL Server. There may be more than one paging file on your system. OpenVMS uses space in any available paging file.

1. Check to see that your system has at least 18,000 free page file blocks. Refer to your OpenVMS documentation for instructions on how to change this value.
2. When you have at least 18,000 blocks available, check off "System has at least 18,000 free blocks in page files(s)" on the worksheet.

SYSGEN Parameters

The following SYSGEN parameters must be set to specific minimum values in order to run SQL Server:

- WSMAX
- GBLPAGES
- GBLSECTIONS
- GBLPAGFIL

Modify SYSGEN parameters with AUTOGEN.

1. To check the current value of a SYSGEN parameter, run the following commands:

```
$ SET DEFAULT SYS$SYSTEM
$ RUN SYSGEN
SYSGEN> USE CURRENT
SYSGEN> SHOW WSMAX     <--or any SYSGEN parameter

Parameter Name  Current  Default  Min.      Max.    Unit
--------------  -------- -------- --------  -------  -----
WSMAX            4000     4000    2048      800000    Pagelets
internal value    250     250     128        50000    Pages
```

In this example, the current value for the SYSGEN parameter WSMAX is 4000 pagelets, the amount listed under the heading "Current."

For information on modifying system parameters, see *Chapter 8, "Reconfiguring SQL Server and Backup Server,"* or your OpenVMS documentation.

2. You must reboot OpenVMS for SYSGEN parameter changes to take effect.
3. The SYSGEN parameter WSMAX determines the maximum working set for any process on OpenVMS. Increase the value of WSMAX to reflect the amount of memory used by SQL Server. Failure to do so will result in reduced performance. WSMAX must be at least 30,000 pagelets.
4. Record the value of WSMAX on the worksheet, and check off "WSMAX is at least 30,000 free pagelets".

➤ ***Note***

If you plan to reconfigure SQL Server to use more memory, you must also change the value of WSMAX to be at least as large as the amount of memory required by SQL Server. Refer to the section on AUTOGEN in the Chapter 8, "Reconfiguring SQL Server and Backup Server" for information on changing WSMAX.

5. Sybase recommends setting the GBLPAGES, GBLSECTIONS, and GBLPAGFIL parameters by using the ADD_ facility with AUTOGEN. To do so, add these three lines to your *SYS$SYSTEM:MODPARAMS.DAT* file:

```
ADD_GBLPAGES=30000
ADD_GBLSECTIONS=2
ADD_GBLPAGFIL=2500
```

These lines add an additional 30,000 global pagelets and 2 global sections to your OpenVMS system, and permit an additional 30,000 global pagelets to be paged to the system page file.

System 11 SQL Server uses global memory to share data among engines. (Both multiprocessor and uniprocessor machines use global memory.) The default amount of global memory used is 30,000 pagelets. The GBLPAGES and GBLPAGFIL parameters must be adjusted so that at least 30,000 pagelets are always available to SQL Server. In addition, SQL Server uses two global sections for the global memory. If this memory is not available, SQL Server may not start.

If you reconfigure SQL Server to use more memory, you must also change the values for GBLPAGES and GBLPAGFIL. Refer

to the Chapter 8, “Reconfiguring SQL Server and Backup Server” for further information on configuring these resources.

6. Record the values of GBLPAGES, GBLSECTIONS, and GBLPAGFIL on the worksheet, and check off these items.

➤ ***Note***

If you are installing Language Modules only, skip to Chapter 3.

Preparing Devices

During a **sybinit** session for a new configuration, you must provide the information listed on your worksheet for the master and *sybsystemprocs* devices. During a **sybinit** session for an upgrade from pre-10.0 SQL Server, you must provide this information for *sybsystemprocs*. You can provide information for the auditing device during a new configuration session or in a reconfiguration session.

◆ ***WARNING!***

Sybase recommends that you do not put the *sybsystemprocs* or auditing databases on the master device.

Device Sizes

The following table lists the recommended device sizes.

Table 2-3: SQL Server device size requirements

Device	Default Size	Minimum Size	Recommended Size
Master	24MB	24MB	
sybsystemprocs	21MB	21MB	21MB; more for added stored procedures
Auditing (*sybsecurity*)	None	5MB	7MB; more for specialized auditing

Database sizes

Table 2-4 describes the contents of the master device.

Table 2-4: Database sizes

Database Name	Size	Function
master	8MB minimum	Contains information about users, databases and objects, and system catalog entries.
model	2MB	Template database with catalog entries.
tempdb	2MB minimum	Work area for SQL Server. Cleared and rebuilt each time SQL Server is started.
pubs2 *interpubs* *jpubs*	3MB 3MB 3MB	Optional sample database in U.S. English, Western European, and Japanese versions; used in SQL Server documentation examples.

Recording Device Information

Refer to "Sybase Devices" on page 1-7 for information to help you determine what size to use for these devices and whether or not you want to specify "contiguous" for the master device.

1. Record logical device names for the *sybsystemprocs* and auditing devices on your worksheet. The logical device name is the internal name used within Sybase to refer to the device. The default for *sybsystemprocs* is *sysprocsdev*; the default for auditing is *sybsecurity.*
2. Record the sizes for each device, and for the *sybsystemprocs* and *sybsecurity* databases, on the worksheet.
3. Determine whether or not you want to specify "contiguous" for the master device and circle the selected option on your worksheet.
4. Repeat the instructions in the following section as many times as necessary to complete the remaining items in the "Master Device" (new installations only), "*sybsystemprocs* Device" and "Auditing Device" sections of the worksheet.

➤ ***Note***

Place the Sybase software and all disk devices, including the master device, on the same machine. Placing the software and devices in remote locations affects performance adversely.

Selecting Files for Devices

When you use a file for a device:

- The directory must exist prior to running the configuration program.
- The amount of free space necessary for each device is:

Table 2-5: Free space necessary for Sybase devices

Device	Space necessary	Space in blocks
master	24MB	49,152
sybsystemprocs	21MB	43008
auditing	5MB	10,240

- The device directory protections must be set to SYSTEM=RWED, OWNER=RWED, and the directory owner should be the user SYBASE.

◆ ***WARNING!***

Avoid placing devices on *SYBASE_SYSTEM* because the *SYBASE_SYSTEM* logical changes with each update of SQL Server.

Perform the following steps to select a physical device and a directory on that device for the SQL Server device:

1. To find free space on disk prior to configuration, type the following command from the OpenVMS prompt:

   `$ SHOW DEVICE` *DEVICE_NAME*

   where device_name is the device on which you want to find space. Select a device with enough space for the SQL Server device.

2. If the directory does not already exist, create it with a command like the following:

```
$ CREATE/DIRECTORY/PROTECTION=(S:RWED,O:RWED)/OWNER=[SYBASE]-
_$ DUB0:[sybasedevices.master]
```

   If the directory does exist, make sure that it has the correct protections and the Sybase user is the owner.

3. Check off "Directory created" on the worksheet. Record the directory specification for the device.

Operating System Setup

This section discusses operating system issues that must be considered in order to run SQL Server. Topics covered include:

- Installing *DATASERVER.EXE* shared on multiprocessor machines
- Shared memory and paging files
- Automating foreign command and symbol definition
- Logical names
- Companion Servers
- Starting servers during system restart
- Restarting an application

Installing *DATASERVER.EXE* For Multi-engines

If you plan to run multiple engines, use the VMS install utility to make *SYBASE_SYSTEM*:[SYBASE.BIN] DATASERVER.EXE as a known shared image so that the engines share part of the memory used by SQL Server. To do this, you need 5500 global pages and 1 global section in addition to those needed to start and run SQL Server (see Chapter 4, "Installing SQL Server" to determine what you need for your system). To find out how many global pages and global sections you have available, follow these steps:

1. Invoke the **SYSGEN** utility with this command:

   ```
   $ RUN SYS$SYSTEM:SYSGEN
   ```

2. Find out how many global sections are configured for your system:

   ```
   SYSGEN> show gblsections
   ```

3. Find out how many global pages are configured for your system:

   ```
   SYSGEN> show gblpages
   ```

4. Run the following command to find out how many global sections are in use and how many global pages are used and unused:

   ```
   $ INSTALL LIST/GLOBAL/SUMMARY
   ```

If you don't have enough of either of these resources, perform the following steps:

1. Add the following two lines to your *SYS$SYSTEM:MODPARAMS.DAT* file or your *min_parameter*:

   ```
   ADD_GBLPAGES=5500
   ADD_GBLSECTIONS=1
   ```

2. Run **AUTOGEN** in the normal way and reboot your VMS system for the changes to take effect. When your system is up, type the command:

```
$ @SYBASE_SYSTEM:[SYBASE.INSTALL]INSTALL_DATASERVER.COM
```

This command configures your SQL Server so that the amount of extra memory used by additional engines is minimal.

Shared Memory and Paging Files

By default, the shared memory pages are stored in the system paging file(s). The location of the paging file(s) can be determined as shown below:

```
$ show memory/files
     System Memory Resources on  4-FEB-1991 14:19:49.63

Paging File Usage (pages):                Free   Reservable Total
DISK$BEAR_SYS:[SYS0.SYSEXE]SWAPFILE.SYS  16200    16200     16200
DISK$BEAR_SYS:[SYS0.SYSEXE]PAGEFILE.SYS  11254    35351    131600
```

Sybase recommends that the shared memory be left mapped to your system's page files. However, if it is better for your site to map these to another file, this can be done at server startup time. A properly configured Sybase installation should give SQL Server a large enough working set to prevent paging. (see Chapter 8, "Reconfiguring SQL Server and Backup Server" for more information). You can select the shared memory files to use by adding the **/section_file** qualifier to the server command line in your runserver file, located in *SYBASE_SYSTEM*:[SYBASE.INSTALL]. By

default, the runserver file is named RUN_servername.COM. The runserver file is created by sybinit.

This command:

```
SERVER/SECTION_FILE=SYBASE_SYSTEM:[SYBASE]SYBASE_SECTION
```

creates and uses the shared memory file:

SYBASE_SYSTEM:[SYBASE]SYBASE_SECTION.SERVERNAME_KRG

If you use the **/section_file** qualifier, make sure you have enough space on the disk where you are placing the shared memory file.

Start the server as usual, using the Sybase **startserver** utility. See the *SQL Server Utility Programs for OpenVMS* for more information on using the **startserver** and **dataserver** commands.

➤ ***Note***

When the **/section_file** qualifier is used, SQL Server overwrites any existing section file. As a result, the startup time is slightly slower than it is without the qualifier.

Foreign Commands, Symbol Definitions, and Logical Names

SYBASE_SYSTEM:[SYBASE.INSTALL]SYBASE_USER.COM defines symbols and foreign commands for running stand-alone utilities such as **isql** and **bcp**.

The *SYBASE_SYSTEM:[SYBASE.INSTALL]SYLOGICALS.COM* file contains the definitions for the installation-specific logical names. These Sybase-specific logicals are generated during the **sybinit** procedure and need to be defined in order to access the Sybase products. In a primary installation, these logical names are defined at the system level; in a secondary installation, they are defined at the process level.

Depending on the number of users that access Sybase products, the system administrator can add the following lines to either the system *SYS$MANAGER:SYLOGIN.COM* or a user's private *LOGIN.COM* file:

```
@SYBASE_SYSTEM:[SYBASE.INSTALL]SYBASE_USER.COM
@SYBASE_SYSTEM:[SYBASE.INSTALL]SYLOGICAL.COM
```

SYBASE_SYSTEM:[SYBASE.INSTALL]SYBASE_ MANAGER.COM defines all Sybase system administration symbols and foreign commands (for example, **startserver** and **showserver**).

Insert the following into the Sybase system administrator's *LOGIN.COM* file:

```
@SYBASE_SYSTEM:[SYBASE.INSTALL]SYBASE_MANAGER.COM
```

Installation-Specific Logical Names

SYBASE_SYSTEM

The person who installs SQL Server specifies the *SYBASE_SYSTEM* logical name during a **VMSINSTAL** session. *SYBASE_SYSTEM* defines the location of the Sybase products on your system. The Sybase product line contains the directories with the Sybase libraries and include files.

SYBASE

Sybase specifies the location of the interfaces file used when opening a connection to SQL Server. **VMSINSTAL** defines the **Sybase log**ical name as *SYBASE_SYSTEM:[SYBASE]*.

User-Specified Logical Names

Each user can define the values of the following process-level logical names by entering the values in the *LOGIN.COM* file, located in the user's *SYS$LOGIN* directory.

DSQUERY

The DSQUERY logical name specifies which server in the interfaces file is to be used by a client requesting a connection. If the SQL Server is named SYBASE, this logical name need not be defined.

SYBASE_EDITOR

The *SYBASE_EDITOR* logical name specifies which text editor to call from within **isql** and APT Workbench™. The possible values of *SYBASE_EDITOR* are *CALLABLE_LSE*, *CALLABLE_EDT*, and *CALLABLE_TPU*.

You can also set the *SYBASE_EDITOR* logical name to point to a command file. The command file should first redefine *SYS$INPUT* to the translation of *SYS$OUTPUT*, and then invoke the user's editor. The parameter **P1** in the command file will contain the name of the temporary file to be edited.

Setting the logical name to a callable editor does not spawn a subprocess, but setting it to a command file does. The subprocess is given the command:

```
$ @command_file file_name
```

where the *SYBASE_EDITOR* logical points to the *command_file* and *file_name* is the *fpl* procedure that you have selected within APT-Edit™, or the temporary *sql* file created by **isql** when you type "edit". Here is an example of such a command file:

```
$ define/nolog/user sys$input 'f$trnlnm("SYS$OUTPUT")
$ edit/tpu 'p1'
```

Companion Servers

The Sybase Companion Server (CFT) feature makes use of the fault-tolerant capabilities of the VMScluster environment. This feature allows you to install SQL Server processes on multiple nodes in the cluster, with only one server active at a time. If the node running the active SQL Server fails, or if the SQL Server process itself fails, a Server process on one of the other nodes takes over.

The server you normally use is called the **Master Server**; the other server processes in the cluster are called **Companion Servers**. All SQL Server processes in the cluster that participate in the Companion Server scheme are initialized up to the beginning of the master device initialization phase, and then wait for an exclusive lock on the master device.

If the Master Server is functioning normally, none of the Companion Servers is granted the lock for the master device. If the Master Server fails, one of the Companion Servers is granted the lock and becomes the active server.

The exclusive locking used by the Companion Server scheme is based on the server name as entered in the interfaces file.

◆ ***WARNING!***

Make sure each SQL Server in the VMScluster has a unique name.

For the Companion Server feature to work, all virtual devices and interfaces files that the servers use must be on disks that are accessible to all nodes in the cluster. This accessibility ensures that the transaction logs and all the data are available to any Companion Server when it becomes the active server, and the Sybase recovery

procedure can make sure the database remains consistent. Also, the disks must be either dual ported or connected to a Hierarchical Storage Controller (HSC) so that failure of a single node does not prevent access to the disks containing database and log information.

The production Master Server at your site may have been tuned and configured for an expected workload. The particular model computer and its configuration were probably chosen to match the server's configuration. Run the Companion Server on a similar machine since it will inherit the configuration of the Master Server. Performance may suffer if the node is less powerful or smaller. For example, the Master Server will not run more engines than available CPUs. If failover is to a uniprocessor machine, only one engine will start, even if the server is configured for more.

To start a remote Companion Server process (that is, to start the Companion Server on a different node from the one you're logged in to), the DECnet *TASK 0* object must be accessible. If the default access permissions to *TASK 0* are modified or restricted, the SQL Companion Server process cannot start remotely, although it can be started locally on the node where it will execute.

If you use an object name rather than an object number when running SQL Server on DECnet, *TASK 0* access may prevent the server from starting. All object names must attach to *TASK 0*.

To find out the status of the *TASK 0* object, use this series of commands:

```
$ run sys$system:ncp
NCP> show known objects
```

Using Companion Servers

The following instructions assume you have already configured SQL Server and set up a DECnet proxy. If you are configuring SQL Server for the first time, you may also activate the Companion feature at that time. You can use **sybinit** to upgrade an existing SQL Server to include the companion feature.

Starting Companion Servers

To start Master or Companion Servers, use the **startserver** utility. **startserver** can start Companion Servers from a remote node. You can also start Companions locally once the Master Server is running.

The Sybase logical names need to be defined on each node where you plan to run Sybase software.

Copy the *SYBASE_SYSTEM:[SYBASE.INSTALL]SYB_EXEC.COM* command procedure to the *SYS$LOGIN* directory for the Sybase and/or *SYSTEM* proxy login for each node where you plan to run a server.

Use the following command to start Companion Servers:

```
$ startserver /server=server_name -
_$ [/masterserver=master_node] -
_$ /companions=(remote_node1,remote_node2..)
```

where the remote nodes are listed in the order of server recovery. If the Companion Servers are started locally, the following command starts a Companion Server on the local node, if a Master Server is already active on some other node in the cluster:

```
$ startserver /server=server_name[/wait]
```

Stopping Companion Servers

To stop Companion Servers, use the **stopserver** utility. Specify the **/server, /companions** and **/abort** qualifiers to stop Companion Servers. Be sure to include **/companions.** The **/server** qualifier without the **/companions** qualifier stops the Master Server.

Locating the Master and Companion Servers

To list the Master and Companions for a particular server, use the **showserver** utility with the **/server** qualifier. The Companion Servers are listed in the order that they become the Master Server. The order of precedence is the order in which you started the servers with the **startserver /companions** command.

Restarting a Server After a Failure

If the active SQL Server exits, but the node it is on does not fail, the companion lock is released and the first Companion Server started becomes the master SQL Server. If the node itself fails, the VMScluster software suspends all users of the cluster for a time. When cluster operation resumes, the first Companion Server started becomes the active server, even if the node running the Master Server is the first to restart.

Restarting an Application in a Companion Server Environment

You can have your own applications restart either manually or automatically in a Companion Server environment. If you want them to restart automatically, you must write a routine that handles Open Client DB-Library™ error status codes. The Open Client DB-Library calls that may produce errors significant to the application are those that perform I/O to SQL Server, namely, **dbopen**, **dbsqlexec**, **dbnextrow**, **dbcancel**, and **dbcanquery**.

dbopen reads the "retries" and "delay" numbers from the interfaces file. These numbers tell dbopen how many times to try to connect to SQL Server and how long to pause between attempts.

The constants listed in *Table 2-6: Constants in the SYBDB.H Header File* are defined in the *SYBDB.H* header file and are associated with the error messages shown.

Table 2-6: Constants in the SYBDB.H Header File

Constant	Error Message
SYBECONN	Unable to connect: SQL Server is unavailable or does not exist.
SYBEFCON	SQL Server connection failed.
SYBEOOB	Error in sending out-of-band data to SQL Server.
SYBEREAD	Read from SQL Server failed.
SYBESEOF	Unexpected EOF from SQL Server.
SYBESOCK	Unable to open socket.
SYBETIME	SQL Server connection timed out.
SYBEUHST	Unknown host machine name.
SYBEVMS	Sendflush: VMS I/O error.
SYBEWRIT	Write to SQL Server failed.

See:

SYBASE_SYSTEM:[SYBASE.SAMPLE_SESSIONS]RESTART_EXAMPLE.C

for an example of a program fragment that illustrates an error-handling routine. The code includes comments.

What's Next?

Chapter 3, "Loading Software from Media" provides instructions for loading your Sybase distribution files from media.

3 Loading Software from Media

This chapter provides instructions for loading Sybase software from distribution media onto your system with **VMSINSTAL**. Topics covered in this chapter are:

See "Sample VMSINSTAL Session" on page C-2 for more information about using **VMSINSTAL** to unload the software from the media.

Overview of the Procedure

The software loading procedure can be summarized as follows.

1. Log in with the user name SYSTEM.
2. Place the global tape or CD-ROM containing the Sybase software in the appropriate drive.
3. (CD-ROM only) Mount the CD-ROM on your system.
4. Run the **VMSINSTAL** utility to load your SQL Server software into the installation directory.

Loading the software may take 15 minutes or more, depending on how many products you are loading onto your system.

Prerequisites

1. Complete the preparation tasks in Chapter 2, "Preparing for New Installation or Upgrades" before you begin the tasks in this chapter.

➤ ***Note***

Use the information you recorded under "information for VMSINSTAL" on the worksheet to complete this VMSINSTAL session.

2. Review the sample **VMSINSTAL** session in Appendix C, "Sample Sessions," before you begin to load your software. If you are

unsure of a selection at any time during your session, you can check this sample for information.

3. As you execute the loading process, refer to:
   - The *SQL Server Release Bulletin*
   - Operating system documentation for your platform
   - The information recorded on the worksheet

About the Distribution Media

The Sybase products you have purchased are shipped to you on global tape or CD-ROM. The Customer Authorization String that you recorded in the Chapter 2, "Preparing for New Installation or Upgrades" allows you to access the software you have purchased as you run the **VMSINSTAL** utility.

About *VMSINSTAL*

The **VMSINSTAL** utility loads your products from distribution media into the installation directory. Answer the prompts with the information you prepared in Chapter 2, "Preparing for New Installation or Upgrades."

If a prompt has a default answer, it appears in brackets after the question. To enter the default answer, press Return.

Loading the Software

To load your Sybase software onto your system, perform the following steps:

1. Place the distribution media in the tape drive or CD-ROM device.
2. Log in as "SYSTEM".
3. (CD-ROM only) Mount the CD-ROM with the following command:

   $ MOUNT *DEVICE_NAME* SYBASE CDROM

   where *DEVICE_NAME* is the name of the CD-ROM device you recorded on your worksheet.

4. Start **VMSINSTAL** with one of the following commands:

Tape	**`@sys$update:vmsinstal sybase020 device_name`** *(device_name* is the tape drive you noted in Chapter 2, "Preparing for New Installation or Upgrades.")
CD-ROM	**`@sys$update:vmsinstal sybase020 cdrom:[sybase]`**

5. Refer to the information you recorded on the worksheet to answer the following **VMSINSTAL** prompts:
   - *SYBASE_SYSTEM*. Enter the definition of *SYBASE_SYSTEM*.

◆ ***WARNING!***

If you are upgrading SQL Server, ensure that you enter the new definition of SYBASE_SYSTEM at the VMSINSTAL prompt, and not the SYBASE_SYSTEM definition for the existing installation.

   - **Secondary installation** – Answer "yes" or "no" when asked if this is a secondary installation, according to the option circled on your worksheet. (See "Determining Installation Type" on page 2-2 for information.)
   - **Customer Authorization String –** Enter the string from the software packaging that allows you to access your products, as recorded on your worksheet.
   - **Sybase products –** The **VMSINSTAL** utility displays a menu listing the available products. Enter the number that corresponds to each product you want to install, pressing Return after each number. When you have entered the number of the last product you want to install on this machine, press Return twice to enter a blank line.
   - **Product confirmation** – The **VMSINSTAL** utility lists the products you have chosen, and prompts for confirmation. Answer:

     "y" or "Y" if the list is correct

     "q" or "Q" to quit

     Any other character to see the menu and choose again.

6. Wait while **VMSINSTAL** installs your products**. VMSINSTAL** displays a series of screen messages while it installs your software. Installation may take 15 minutes or more. Do not interrupt the

installation. When **VMSINSTAL** is finished, it repeats the list of products you selected for installation.

7. (CD-ROM only) When the products have finished loading, dismount the CD-ROM with the following command:

   ```
   $ dismount cdrom
   ```

8. Remove the distribution media from the drive.

Reviewing the Locales File

If you plan to run **sybinit** in languages other than U.S. English or character sets other than iso_1, you may need to edit the *locales.dat* file so that the entries for the languages and character sets contain definitions that match the definitions used by your operating system.

1. Follow the instructions in "Editing the locales.dat file" on page 8-5 to review and edit the file.
2. Check off the locales file entry on your worksheet.

◆ ***WARNING!***

If locales definitions in your system configuration files do not match SYBASE locales definitions, your applications will not run properly.

What's Next?

If you are configuring a new SQL Server, proceed to Chapter 4, "Installing SQL Server," which provides instructions for a new configuration.

If you are upgrading a SQL Server to release 11.0.x, see Chapter 5, "Upgrading SQL Server," which provides instructions for an upgrade.

If you have installed Language Modules only, see Chapter 7 to reconfigure languages, character sets, or sort order for SQL Server, or to reconfigure languages or character sets for Backup Server.

If you are migrating from VAX to Alpha, see Chapter 10, "Migrating SQL Server from VAX to Digital OpenVMS Alpha."

4 Installing SQL Server

This chapter provides instructions for using an interactive **sybinit** session to configure a new SQL Server. Topics are:

Notes About New Installation

1. If you are unsure of a selection at any time during your installation session, you can refer to the sample **sybinit** session in *SYBASE_SYSTEM:[SYBASE.SAMPLE_SESSIONS]INSTALL.SPL*

for information. See Appendix C, "Sample Sessions," for more information.

2. Perform the preparation tasks in Chapter 2, "Preparing for New Installation or Upgrades," including setting protections for the files you plan to use for the *master*, *sybsystemprocs*, and *sybsecurity* (auditing) devices.
3. Load Sybase files from media, as explained in Chapter 3, "Loading Software from Media."
4. Follow the instructions in this chapter to install a new SQL Server.

   As you complete the installation tasks, fill out the section for Chapter 4 on the "SQL Server Configuration / Upgrade Worksheet" on page 1-13.

Locating the Sybase Installation Structure

Once you perform an installation, you should leave the Sybase installation structure components in place. If you move any of these components, Sybase does not guarantee the proper operation of Sybase software.

The command files and programs necessary for installation-related system management are located in the directory.

Using *sybinit* Interactively

For most new installations, upgrades, or reconfigurations of a SQL Server, the **sybinit** interactive mode is easier to use than a resource file. In the interactive mode, you make selections for your SQL Server configuration from a menu interface. **sybinit** rejects invalid selections or provides warnings or error messages when you make improper selections. In most cases, if you make an error, you can change your selection and continue with the installation session. The interactive mode provides command keys and screen prompts to assist you in

making selections and moving from one menu to another. This and the following chapters focus on the interactive use of **sybinit**.

Table 4-1: sybinit interactive command keys

Key	Command	Action
Ctrl-a	Accept	Accept the values currently listed in the menu and move to the next menu.
Ctrl-b	Backward	Scroll backward one page. Use this command key only when a menu or list does not fit on a single screen.
Ctrl-c	Quit	Quit **sybinit** at any point and return to the DCL prompt.
Ctrl-f	Forward	Scroll forward one page. Use this command key only when a menu or list does not fit on a single screen.
Ctrl-r	Refresh	Refresh the screen.
Ctrl-x or Esc	Exit	Exit the current menu or prompt.
Ctrl-w	Write to resource file	Generate a resource file that duplicates the values specified during the current interactive **sybinit** session. Default file name is *SYBASE_SYSTEM:[SYBASE.INIT.LOGS] RESOURCE.DMP*; **sybinit** prompts for alternate.
?	Help	Display an online help screen.

sybinit Command Line Qualifiers

You can modify the behavior of **sybinit** in interactive mode with several command line qualifiers, which you can type when you start **sybinit** at the prompt. For example, the **/language** or **/charset** qualifiers run **sybinit** in a language other than U.S. English or a character set other than iso_1. Additional command line qualifiers are appropriate for a **sybinit** resource file session. See Table B-1 on page B-3 for information about **sybinit** command line options.

Using *sybinit* with a Resource File

A **sybinit** resource file is more efficient than running several interactive **sybinit** sessions if you are installing or upgrading several SQL Servers at once, and you want each server to be configured in the same manner. However, a **sybinit** resource file session runs non-interactively, so you cannot correct errors during the session.

Resource file templates designed for installing SQL Servers, upgrading SQL Servers, and installing Backup Servers are included with **sybinit**.

See Appendix B, "sybinit Resource Files," for more information.

sybinit and Interfaces Files

At the start of any interactive **sybinit** session, **sybinit** reads the entire interfaces file in the *Sybase* release directory and displays a warning for each invalid entry in the interfaces file. During an interactive upgrade session, when you enter the release directory of the SQL Server to be upgraded at the Upgrade Existing SQL Server menu, **sybinit** reads the interfaces file in that release directory and displays warnings for all invalid entries. These messages are also written to the **sybinit** log file during interactive and resource file sessions.

Invalid entries include duplicates, entries with syntax errors, entries listing hosts that no longer exist, entries listing unsupported or unknown protocols, and entries listing unknown service types. Warning messages do not prevent you from completing the **sybinit** session. Press Return after each warning message to continue.

Some sample warning messages follow:

```
Could not determine hostname for address '130.214.130.22';
discarding this service.
Syntax error while parsing service ' query decnet dec-ether
130.214.130.22'
A duplicate entry 'TESTER' has been detected. The first entry
will be retained and all others discarded.
Parse error on service line ' query tcp
```

When **sybinit** writes to an interfaces file after you have created or edited an entry, it discards the invalid entries that caused the warnings. To avoid delays during your **sybinit** session, you can review the interfaces file before you begin the session and manually edit the file to remove any invalid entries.

Verifying *SYBASE_SYSTEM* Definition

Before you begin a **sybinit** session, verify that *SYBASE_SYSTEM* was defined correctly when the Sybase software was installed from media.

1. Log in using the "Sybase" account you created in Chapter 2, "Preparing for New Installation or Upgrades." At the prompt, type:

   $ SHOW LOGICAL SYBASE_SYSTEM

   The value returned should match the logical name recorded on your worksheet.

2. If the values do not match, type the appropriate command at the prompt:

   For a primary installation:

   $ DEFINE /SYS/EXEC SYBASE_SYSTEM *ROOTED_LOGICAL*

   For a secondary installation:

   $ DEFINE SYBASE_SYSTEM *ROOTED_LOGICAL*

   where *rooted_logical* is the new *SYBASE_SYSTEM* definition recorded on your worksheet.

Starting *sybinit*

1. Make sure the **sybinit** symbol is defined:

   $ SHOW SYMBOL SYBINIT

2. If the symbol is not defined, execute the following:

   $ @SYBASE_SYSTEM:[SYBASE.INSTALL]SYBASE_MANAGER

3. Start **sybinit** by typing the following command at the DCL prompt:

   $ SYBINIT

4. When the first **sybinit** menu appears, verify that the Release Directory entry is correct:

```
The log file for this session is
'SYBASE_SYSTEM:[SYBASE.INIT.LOGS]log1014.1'.

SYBINIT

1. Release Directory: SYBASE_SYSTEM:[SYBASE]
   Type of Installation:  secondary (process-level or
                                     or group-level access)
2. Edit / View Interfaces File
3. Configure a Server product
4. Configure an Open Client/Server product

Ctrl-a Accept and Continue, Ctrl-x Exit Screen, ? Help.

Enter the number of your choice and press return:
```

The displayed release directory must point to the logical for your new 11.0.x *Sybase* release directory. For example, if you recorded SYBASE_110 on your worksheet as the definition for the SYBASE_SYSTEM logical, SYBASE_SYSTEM must point to SYBASE_110 before you continue the **sybinit** session.

If the directory is incorrect, select Release Directory and type the correct path at the prompt. **sybinit** displays the new path.

➤ ***Note***

The Type of Installation item is for display only. See "Determining Installation Type" on page 2-2 for an explanation of the difference between primary and secondary installations.

Configuring SQL Server

1. Choose Configure a Server Product from the Sybinit menu. **sybinit** displays the Configure Server Products menu:

   ```
   CONFIGURE SERVER PRODUCTS

       Product          Date Installed  Date Configured
   1. SQL Server        Sept 14 96 14:55
   2. Backup Server     Sept 14 96 14:55
   ```

   The "Date Installed" column indicates when the product files were copied onto disk. The "Date Configured" column indicates when you configured the product with **sybinit**. **sybinit** records the current date and time in the "Date Configured" column when you complete the SQL Server installation.

2. Select SQL Server. **sybinit** displays the following menu:

   ```
   NEW OR EXISTING SQL SERVER

   1. Configure a new SQL Server
   2. Configure an existing SQL Server
   3. Upgrade an existing SQL Server
   ```

3. Select Configure a New SQL Server.

➤ ***Note***

Enter the SQL Server name that you select in capital letters. **sybinit** is case sensitive. If you do not capitalize the server name, you will have to enclose it in quotes when you enter it on your operating system.

4. **sybinit** displays the Add New SQL Server menu:

```
ADD NEW SQL SERVER

1. SQL Server name: SYBASE
```

5. To accept the default SQL Server name, SYBASE, press Ctrl-a.

   To change the default server name, select SQL Server Name, and enter the name at the prompt. Follow these naming conventions:

   - Restrict the name length to 11 characters
   - Make the initial character a letter (ASCII a–z, A–Z)
   - Make sure the characters that follow the initial character are letters, numbers, or underscores
   - Use only 7-bit ASCII characters
   - Use a name that is unique to the network to avoid name conflicts with other applications
   - If you do not capitalize the server name, you will have to enclose it in quotes when you enter it as a value for a logical name or as a value for a command line qualifier.

Press Ctrl-a when "SQL Server name" shows the correct name, and record it on your worksheet.

SQL Server Configuration Menu

sybinit displays the SQL Server Configuration menu:

```
SQL SERVER CONFIGURATION

1. CONFIGURE SERVER'S INTERFACES FILE ENTRY        Incomplete
2. MASTER DEVICE CONFIGURATION                     Incomplete
3. SYBSYSTEMPROCS DATABASE CONFIGURATION           Incomplete
4. SET ERRORLOG LOCATION                           Incomplete
5. CONFIGURE DEFAULT BACKUP SERVER                 Incomplete
6. CONFIGURE LANGUAGES                             Incomplete
7. CONFIGURE CHARACTER SETS                        Incomplete
8. CONFIGURE SORT ORDER                            Incomplete
9. ACTIVATE AUDITING                               Incomplete
10.SET QUOTA FILE LOCATION                         Incomplete
```

Follow the steps in the remaining sections in this chapter to complete the configuration tasks listed in the SQL Server Configuration menu. If you lose your place during the installation process, press Ctrl-x to cancel the current operation and return to this menu.

To complete the **sybinit** configuration session, complete all the items on the SQL Server Configuration menu, even if you want to accept the defaults for some items. As you complete each task, the task is marked "Complete" on the menu.

Creating SQL Server Interfaces File Entry

➤ ***Note***

See Chapter 9, "Administrative Tasks and Performance and Tuning" for information about the format of interfaces file entries.

Select Configure Server's Interfaces File Entry from the SQL Server Configuration menu. **sybinit** displays the Server Interfaces File Entry Screen:

```
SERVER INTERFACES FILE ENTRY SCREEN

    Server name: SYBASE

1. Retry Count: 0
2. Retry Delay: 0
3. Add a new listener service
```

Entering Connection Information (Optional)

1. Select Retry Count. The retry count specifies the number of times a client attempts a connection after an initial failure to connect to the server.
2. At the prompt, enter a retry count value. Enter the value you recorded on your worksheet.
3. Select Retry Delay. The retry delay specifies how many seconds a client waits between retries to connect to the server.
4. At the prompt, enter the retry delay value. Enter the value you recorded on your worksheet.

The Server Interfaces File Entry Screen now displays the retry count and retry delay values that you entered (or 0 if you did not enter any values). You can change either of the displayed values by reselecting the appropriate menu item and entering a new value at the prompt.

Selecting Network Protocol

Select Add a New Listener Service. **sybinit** displays the Select Network Protocol menu:

```
SELECT NETWORK PROTOCOL

1. TCP
2. DECnet
```

Follow the instructions in whichever of the following sections apply to the network protocol(s) you select.

➤ ***Note***

The SPX protocol is not supported for Digital OpenVMS Alpha.

Adding a TCP Entry

To choose TCP, perform the following steps:

1. Select TCP from the Select Network Protocol menu. **sybinit** displays the Edit TCP Service menu:

```
EDIT TCP SERVICE

1. Hostname/Address:
2. Port:
3. Name Alias:
4. Server Type: master
5. Delete this service from the interfaces entry
```

2. To change the default SQL Server host name, select Hostname / Address and enter the hostname or IP address of the machine on which SQL Server is installed.
3. Select Port and enter the 4- or 5-digit port number, as recorded on your worksheet. The range of valid port numbers is from 5000 to 65535.
4. (Optional) Select Name Alias and enter the alias recorded on your worksheet.
5. Server Type refers to whether the entry is for a Master or Companion Server. The default is “master”.

   If you are creating an entry for a Companion Server, select Server Type. At the prompt that asks if the SQL Server is a Companion Server, type “y”.

➤ ***Note***

If you use Companion Servers you must specify listeners for the Companion Servers.

6. Skip to "Accepting the New Entry" on page 4-10.

Adding a DECnet Entry

To choose DECnet, perform the following steps:

1. Select DECnet from the Select Network Protocol menu. **sybinit** displays the Edit DECnet Service menu:

   ```
   EDIT DECNET SERVICE

   1. Nodename:
   2. Object:
   3. Name Alias:
   4. Server Type: master
   5. Delete this service from the interfaces entry
   ```

2. To change the default SQL Server node name, select Nodename and enter the node name recorded on your worksheet.
3. Select Object and enter the object number recorded on your worksheet. The range of valid object numbers is 128–253. Object names are not recommended.
4. (Optional) Select Name Alias and enter the alias recorded on your worksheet.
5. Server Type refers to whether the entry is for a Master or Companion Server. The default is "master".

   If you are creating an entry for a Companion Server, select Server Type. At the prompt that asks if the SQL Server is a Companion Server, type "y".

Accepting the New Entry

1. Verify that all entries in the Edit [Protocol] Service menu (for the protocol you selected) are correct. If necessary, repeat the preceding instructions to correct an entry.
2. Press Ctrl-a to accept the displayed information. **sybinit** asks you if the information is correct:

   ```
   Is this information correct? n
   ```

3. Type “y” to proceed with the installation. **sybinit** returns to the Server Interfaces File Entry Screen:

```
SERVER INTERFACES FILE ENTRY SCREEN

   Server name: SYBASE

1. Retry Count: 5
2. Retry Delay: 5

3. Add a new listener service

Modify or delete a service

Listener services available:

   Protocol Address        Port    Name Alias
4. tcp      harpo          5500    HQ

4. tcp      123.214.100.59 5500    HQ
```

sybinit displays the listener information you entered under the heading Listener Services Available. You can modify or delete this information at any time by selecting the item number of the listener service (item 4 in the example above).

You can also create additional listener entries for the server by repeating the instructions under “Selecting Network Protocol” on page 4-9. This ability to create multiple entries is especially useful for SQL Servers that operate on networks with multiple protocols.

4. Press Ctrl-a to record the new information in the interfaces file. **sybinit** prompts you to write the changes to the file:

```
Write the changes to the interfaces file now? n
```

5. Type “y” to write the information to the interfaces file. **sybinit** returns to the SQL Server Configuration menu and marks the first task “Complete”.

Configuring the Master Device

1. Select Master Device Configuration from the SQL Server Configuration menu. **sybinit** displays the Master Device Configuration menu:

```
MASTER DEVICE CONFIGURATION

1. Master Device: SYBASE_SYSTEM:[SYBASE]master.dat
2. Size (meg): 24
3: Contiguous: yes
```

2. Select Master Device. At the prompt, enter the full file specification, as recorded on your worksheet. **sybinit** returns to the Master Device Configuration menu.

◆ ***WARNING!***

Do not accept the default file specification for the master device, which places it on *SYBASE_SYSTEM*. If you place the master device on *SYBASE_SYSTEM,* you will encounter problems when you attempt to upgrade your SQL Server.

3. To change the displayed default size of 24MB, select Size. **sybinit** prompts you for the size of the master device. Enter the size, as recorded on your worksheet.
4. The default for Contiguous is "yes". If you entered "no" on your worksheet, select this item and type "no" at the prompt. See "Master Device Contiguity" on page 1-8 for an explanation of the two options for this menu item.
5. Verify that all entries in the Master Device Configuration menu are correct. If necessary, repeat the instructions in this section to correct an entry.
6. Press Ctrl-a to accept the information. **sybinit** returns to the SQL Server Configuration menu and marks the Master Device Configuration task "Complete".

➤ ***Note***

sybinit does not create the master device or execute any other installation options until you complete the installation instructions on page 4-23.

Configuring *sybsystemprocs*

Select Sybsystemprocs Database Configuration from the SQL Server Configuration menu. **sybinit** displays the Sybsystemprocs Database Configuration menu:

```
SYBSYSTEMPROCS DATABASE CONFIGURATION
1. sybsystemprocs database size (Meg): 21
2. sybsystemprocs logical device name: sysprocsdev
3. create new device for the sybsystemprocs database: yes
4. physical name of new device:
SYBASE_SYSTEM:[SYBASE]sybprocs.dat
5. size of the new device (Meg): 21
```

Entering Database Size

1. To change the displayed default size of 21MB, select Sybsystemprocs Database Size.
2. At the prompt, enter the size recorded on your worksheet. The size must be at least 21MB.

Creating a New Device

For a first-time installation of SQL Server, you should create a new device for the *sybsystemprocs* database.

1. Select Sybsystemprocs Logical Device Name to change the default logical name displayed for the *sybsystemprocs* device.
2. At the prompt, type the logical device name recorded on your worksheet.

◆ ***WARNING!***

Sybase recommends that you do not put the *sybsystemprocs* database on the master device. If you plan to put the *sybsystemprocs* database on the master device, see "Using an Existing Device" on page 5-20 for instructions.

3. If **sybinit** does not display "yes" at Create New Device for the Sybsystemprocs Database, select this option to change the displayed value. **sybinit** toggles this item's setting to "yes".
4. Select Physical Name of New Device.

◆ ***WARNING!***

Do not accept the default file specification for the *sybsystemprocs* device, which places it on *SYBASE_SYSTEM*. If you place the *sybsystemprocs* device on *SYBASE_SYSTEM,* you will encounter problems when you attempt to upgrade your SQL Server.

5. At the prompt, type the file specification to use for the *sybsystemprocs* device, as recorded on your worksheet.
6. To change the displayed default size of 21MB, select Size of the New Device. At the prompt, type the device size, as recorded on your worksheet.
7. Press Ctrl-a to accept the settings displayed on the Sybsystemprocs Database Configuration menu. **sybinit** returns to the SQL Server Configuration menu and marks the Sybsystemprocs Database Configuration task "Complete".

Setting the Error Log Location

1. Select Set Errorlog Location from the SQL Server Configuration menu. **sybinit** displays the Set Errorlog Location menu:

```
SET ERRORLOG LOCATION
1. SQL Server errorlog: SYBASE_SYSTEM:[SYBASE.INSTALL]ERRORLOG.
```

➤ ***Note***

If you have multiple SQL Servers, you may want to enter a different error log specification for each SQL Server you install. If you accept the default specification for each SQL Server, all the servers will share the same error log.

2. To change the displayed default location, select SQL Server Errorlog. At the prompt, enter the file specification of the error log.
3. Record the file specification of the error log file on your worksheet.
4. Press Ctrl-a to accept the displayed error log file. **sybinit** returns to the SQL Server Configuration menu and marks the Set Errorlog Location task "Complete".

Naming a Default Backup Server

1. Select Configure Default Backup Server from the SQL Server Configuration menu. **sybinit** displays the menu:

```
SET THE SQL SERVER'S BACKUP SERVER
1. SQL Server Backup Server name: SYB_BACKUP
```

2. To change the displayed name, select SQL Server Backup Server Name and enter the Backup Server name at the prompt. Record this name on your worksheet.
3. Press Ctrl-a to accept the displayed Backup Server name. **sybinit** returns to the SQL Server Configuration menu and marks the Configure Default Backup Server task "Complete".

At this point, you have only defined the name of the default Backup Server for SQL Server to use. You have not yet configured the Backup Server itself.

To dump and load databases, you must configure a new Backup Server (with the name specified in step 2 above) after successfully configuring SQL Server. See Chapter 6, "Installing Backup Server" for more details.

Configuring Languages

➤ ***Note***

The Japanese Language Module is not supported forDigital OpenVMS Alpha.

1. Select Configure Languages from the SQL Server Configuration menu. **sybinit** displays the Configure Languages menu:

```
CONFIGURE LANGUAGES

     Current default language: us_english

     Current default character Set: ISO 8859-1 (Latin-1) -
     Western European 8-bit character set.

     Current sort order: Binary ordering for the ISO 8859/1
     character set (iso_1).

     Select the language you want to install, remove, or
     designate as the default language

     Language      Installed?  Remove  Install Make Default

1. us_english      yes         no      no      yes
2. french          no          no      no      no
3. german          no          no      no      no
4. spanish         no          no      no      no
```

The "Installed?" column indicates whether or not SQL Server can currently use the language. For first-time installations, only the default language, U.S. English, is marked as installed.

The "Remove" column indicates whether or not **sybinit** removes the language when you complete the SQL Server installation.

The "Install" column indicates whether or not **sybinit** installs the language when you complete the installation. By default, **sybinit** installs no additional languages during a first-time installation.

The "Make Default" column indicates which language SQL Server uses as the default language. Only one language can have a "yes" in the "Make Default" column. For first-time installations, **sybinit** initially marks U.S. English as the default language.

2. To change the displayed language configuration, select the number of the language you want to install, remove, or make the default language.
3. If the language is not currently installed, **sybinit** asks if you would like to install it. If the language has been installed, **sybinit** asks if you would like to remove it. Type "y" or "n" to answer the question.
4. If you are installing a new language or changing the default language, **sybinit** prompts you to indicate whether to make the language the new default. Type "y" or "n" to answer the question.
5. Repeat steps 2 through 4 to configure an additional language.

6. Record the default language and any additional languages on your worksheet.
7. Press Ctrl-a when you have finished configuring SQL Server languages. **sybinit** returns to the SQL Server Configuration menu and marks the Configure Languages task "Complete".

Configuring Character Sets

1. Select Configure Character Sets from the SQL Server Configuration menu. **sybinit** displays the Configure Character Sets menu.

```
CONFIGURE CHARACTER SETS

     Current default language: us_english

     Current default character set: ISO 8859-1 (Latin-1) -
     Western European 8-bit character set.

     Current sort order: binary ordering for the ISO 8859/1
     Latin_1 character set (iso_1).
     Select the character set you want to install, remove, or
     designate as the default character set.

Character Set          Installed?  Remove  Install Make Default

1. ASCII               yes         no      no      no
2. Code Page 437       no          no      no      no
3. Code Page 850       no          no      no      no
4. ISO 8859-1          yes         no      no      yes
5. Macintosh           no          no      no      no
6. Hewlett-Packard     no          no      no      no
```

The "Installed?" column indicates whether or not SQL Server can currently use the character set.

The "Remove" column indicates whether or not **sybinit** removes the character set when you complete the SQL Server installation.

The "Install" column indicates whether or not **sybinit** installs the character set when you complete the SQL Server installation. By default, **sybinit** installs no additional character sets during a first-time installation.

The "Make Default" column indicates which character set SQL Server uses as the default. Only one character set can have a "yes" in the "Make Default" column. For first-time installations, **sybinit** initially marks iso_1 as the default character set.

➤ ***Note***

If you install French or German as SQL Server's default character set, you must select one of the following single-byte character sets as the default character set: iso_1, roman8, cp850, cp437, or mac.

2. To change the displayed character set configuration, select the number of the character set you want to install, remove, or make the default character set.
3. If the character set is not currently installed, **sybinit** asks if you would like to install it. If the character set has been installed, **sybinit** asks if you would like to remove it. Type "y" or "n" to answer the question.
4. If you are installing a new character set or changing the default character set, **sybinit** prompts you to indicate whether to make the character set the new default. Type "y" or "n" to answer the question.
5. Repeat steps 2 through 4 to configure an additional character set.
6. Record the default character set and any additional character sets on your worksheet.
7. Press Ctrl-a when you have finished configuring SQL Server character sets. **sybinit** returns to the SQL Server Configuration menu and marks the Configure Character Sets task "Complete."

Configuring Sort Order

1. Select Configure Sort Order from the SQL Server Configuration menu. **sybinit** displays the Configure Sort Order menu. The options displayed in the menu vary according to the character sets installed for SQL Server.

```
CONFIGURE SORT ORDER
     Current default language: us_english
     Current default character set: ISO 8859-1 (Latin-1) -
     Western European 8-bit character set.
     Current sort order: Binary ordering for the ISO 8859/1 or
     latin_1 character set (iso_1).
     Select a sort order>
     Sort Order                                          Chosen
1. Binary ordering for the ISO 8859-1 or Latin-1 character set.
                                                         yes
2. General purpose dictionary ordering.                  no
3. Dictionary order, case insensitive, accent insensitiveno
4. Dictionary order, case insensitive.                   no
5. Dictionary order, case insensitive with preference.   no
6. Spanish dictionary ordering.                          no
7. Spanish case and accent insensitive dictionary order.no
8. Spanish case insensitive dictionary order.            no
```

The "Chosen" column indicates which sort order **sybinit** assigns as the SQL Server default sort order. Only one sort order can have a "yes" in the "Chosen" column. For first-time installations, **sybinit** initially marks binary ordering as the default sort order.

2. To change the displayed default sort order, enter the number of the sort order you want to install.
3. Record the selected sort order on your worksheet.
4. Press Ctrl-a to accept the designated sort order. **sybinit** returns to the SQL Server Configuration menu and marks the Configure Sort Order task "Complete".

Activating Auditing

Select Activate Auditing from the SQL Server Configuration menu. **sybinit** displays the Activate Auditing menu:

```
ACTIVATE AUDITING
1. Install auditing: no
2. sybsecurity database size (Meg): 5
3. sybsecurity logical device name: sybsecurity
4. create new device for the sybsecurity database: no
```

➤ ***Note***

By default, **sybinit** does not install auditing. If you do not want to install auditing, press Ctrl-a and skip to "Setting SQL Server Quota File Location" on page 4-22.

Selecting the Install Auditing Option

Select Install Auditing to toggle that option to "yes." This selection indicates that **sybinit** should install the auditing database and create the auditing device (if specified) when you execute the SQL Server installation as a whole. If you do not toggle Install Auditing to "yes," **sybinit** does not install the auditing feature, even if you set the remaining options on the menu.

➤ ***Note***

Even after auditing is installed, no auditing occurs until a System Security Officer enables auditing with the auditing system procedures. See the *System Administration Guide* for information.

Entering Database Size

By default, **sybinit** creates a 5MB *sybsecurity* database. Depending on the amount and type of auditing you do, you may need to create a larger *sybsecurity* database.

To change the database size, select Sybsecurity Database Size (Meg). At the prompt, enter the database size you recorded on your worksheet.

◆ ***WARNING!***

Sybase recommends that you do not place the *sybsecurity* database on your master device. Doing so uses up valuable space that is best used for the system tables.

Creating a New Device

If you are configuring a new SQL Server, you should create a new device for the *sybsecurity* database. If you are reconfiguring an existing SQL Server you have the option of placing the *sybsecurity* database on a new database device or on an existing device. Skip to "Using an Existing Device" on page 4-22 if you want to use an existing device.

To create a new device:

1. Select Sybsecurity Logical Device Name. **sybinit** prompts you to enter the new logical device name:

```
Enter the logical device name for the sybsecurity database:
```

2. Enter the new logical device name that you recorded on your worksheet.
3. Select Create New Device for the Sybsecurity Database. **sybinit** toggles that item's indicator to "yes" and adds two new options to the menu. The Install Auditing menu should now look like this:

```
ACTIVATE AUDITING
1. Install auditing: yes
2. sybsecurity database size (Meg): 5
3. sybsecurity logical device name: sybsecurity
4. create new device for the sybsecurity database: yes
5. sybsecurity physical device name:
SYBASE_SYSTEM:[SYBASE]SYBSECUR.DAT
6. size of the new device (Meg): 5
```

4. Select Sybsecurity Physical Device Name. **sybinit** prompts you to enter the physical device name:

```
Enter the physical name of the device to use for the sybsecurity
database:
```

5. Enter the full file specification for the device, as recorded on your worksheet.

◆ ***WARNING!***

Do not accept the default file specification for the auditing device, which places it on *SYBASE_SYSTEM*. If you place the auditing device on *SYBASE_SYSTEM*, you will encounter problems when you attempt to upgrade your SQL Server.

6. If you want to change the default device size of 5MB, select Size of the New Device (Meg) and enter the auditing device size.
7. Press Ctrl-a to accept the settings displayed on the Activate Auditing menu. **sybinit** returns to the SQL Server Configuration menu and marks the Activate Auditing task "Complete".

Using an Existing Device

Use these steps to place the *sybsecurity* database on an existing database device:

1. Select Sybsecurity Logical Device Name from the configuration menu. **sybinit** prompts you to enter the new logical device name:

```
Enter the logical device name for the sybsecurity database:
```

2. Enter the logical device name of an existing database device on which to place the *sybsecurity* database, as recorded on your worksheet.
3. Press Ctrl-a to accept the settings displayed on the Activate Auditing menu. **sybinit** returns to the SQL Server Configuration menu and marks the Activate Auditing task "Complete".

Setting SQL Server Quota File Location

Use these steps to use a quota file to set system parameters for SQL Server:

➤ ***Note***

The quota file must exist before you start the server.

1. Select Set Quota File Location from the SQL Server Configuration menu. **sybinit** displays the following menu:

```
SET QUOTA FILE LOCATION
1. SQL Server quota file:
```

➤ ***Note***

If you do not want to use a quota file, press Ctrl-a to accept the null specification and return to the SQL Server Configuration menu.

2. Select SQL Server quota file.

   The quota file must exist before you start the server.

3. At the prompt, enter the file specification of the quota file. There is no default file specification.

   To remove a quota file, enter a space at the prompt.

4. Record the specification of the quota file on your worksheet.

5. Type Control-a to accept the displayed file specification. **sybinit** returns to the SQL Server Configuration menu.

Completing the Installation

1. After you have completed the required installation tasks, press Ctrl-a at the main SQL Server Configuration menu.

   If you skipped any tasks in this chapter, **sybinit** displays the following warning:

```
Some SQL Server configuration screens were never entered.
```

   You cannot proceed with the installation until you complete the remaining items on the SQL Server Configuration menu. Select the options marked “Incomplete” and press Ctrl-a at the submenus to accept the default values. When all of the items on the SQL Server Configuration menu are marked “Complete,” press Ctrl-a again to proceed with the installation.

   sybinit asks if you want to execute the installation:

```
Execute the SQL Server Configuration now?
```

   At this point you can:

   - Type “n” for “no” to return to the SQL Server Configuration menu.
   - Type “y” for “yes” to continue with the installation.

If you continue the installation, **sybinit** performs a series of tasks, such as:

- Creating the master device
- Creating system tables
- Assigning the default Backup Server name
- Installing the language information
- Starting SQL Server in the background
- Installing the auditing feature

2. If any error messages appear, or the installation fails, see Appendix A, "Recovery from Failure."

   When the installation is successful, **sybinit** displays the following message:

   ```
   Configuration completed successfully.
   Press <return> to continue.
   ```

   Press Return to return to the New or Existing SQL Server menu.

   If you plan to configure a Backup Server, press Ctrl-x once to go to the Configure Server Products menu. To exit **sybinit**, press Ctrl-x repeatedly until the operating system prompt appears.

Setting the System Administrator Password

A user account called "sa" is created for the System Administrator when the Sybase software is installed. A user logged in as "sa" can use any database on a SQL Server, including *master*, with full privileges. "sa" automatically gets the same privileges of the Database Owner ("dbo") when "sa" uses a database.

Immediately after a new installation, there is no password on this user account.The initial and default value for a password is NULL. In a production environment, System Administrators should always have passwords.

The System Administrator should log in to the new SQL Server as "sa" and set a password with the **sp_password** system procedure:

```
$ ISQL /USER="SA" /PASSWORD /SERVER="SERVER_NAME"
1> sp_password null, new_password
2> go
```

See the *System Administration Guide* for more information.

Backing Up New Software

Make a backup copy of your new SQL Server installation. You must configure Backup Server before you can dump and load databases.

What's Next?

To configure Backup Server, see Chapter 6, "Installing Backup Server."

To reconfigure your new SQL Server, see Chapter 8, "Reconfiguring SQL Server and Backup Server."

5 Upgrading SQL Server

This chapter provides instructions for using **sybinit** interactively to upgrade SQL Server to release 11.0.x. Topics covered are:

Notes for Upgrade

1. Read all of the instructions in this chapter before you begin your upgrade.
2. If you are unsure of a selection at any time during your installation session, you can refer to the sample **sybinit** session in *SYBASE_SYSTEM:[SYBASE.SAMPLE_SESSIONS]UPGRADE.SPL* for information. See Appendix C, "Sample Sessions," for more information.
3. Complete the preparation tasks in Chapter 2, "Preparing for New Installation or Upgrades," including setting ownership and permissions for the file you plan to use for the *sybsystemprocs* device.

4. Load the new software from media, as explained in Chapter 3, "Loading Software from Media," into a different release directory from your current SQL Server installation. You need access to both release directories to successfully upgrade SQL Server.
5. Follow the instructions in this chapter to upgrade SQL Server, recording information on the worksheet under the "Chapter 5" heading, and referring to information recorded under the "Chapter 2" heading.

   You can press Ctrl-w at the end of your interactive **sybinit** session to generate a resource file which duplicates the configuration. See Appendix B, "sybinit Resource Files," for information about Ctrl-w and other **sybinit** interactive command keys.

   You can use the **/language** and **/charset** command line qualifiers to execute **sybinit** in a language other than U.S. English. See Appendix B, "sybinit Resource Files" for information on **sybinit** command line options.

Preparing for the Upgrade

Log into the server to be upgraded as System Administrator, using the **isql** utility:

```
isql/username="sa" /password="sa_password" /server="server_name"
```

Logging Off Users

After giving users adequate warning to log off SQL Server, make sure everyone is logged off by doing the following:

1. Enter the **sp_who** system procedure with no parameters:

   ```
   1> sp_who
   2> go
   ```

   sp_who returns a table showing users and processes on SQL Server.
2. Make sure all the users are logged off the system.
3. Leave your server running.

Turning Certain Database Options Off Before Upgrading

Verify the following database options are turned off before beginning the upgrade; otherwise, the upgrade will fail.

- **trunc log on chkpt**
- **no chkpt on recovery**

Ensuring Database Integrity

1. Use the **checkpoint** command in each database to write all modified pages in memory cache that are to be written to database devices.

   Synchronizing databases and transaction logs shortens the time necessary to recover databases in the event of a system crash. See the *System Administration Guide* for information about **checkpoint**.

2. Use the **dbcc** commands **checkalloc**, **checkdb**, and **checkcatalog** to check the logical and physical consistency of all databases, including *master* and *model*.

   Correct all errors reported by **dbcc** before proceeding with your upgrade. (Row and database corrections from **checkdb** are not errors.)

   Make sure no databases are marked “suspect” or “read only.”

   See the *System Administration Guide* for information about **dbcc** commands.

Checking for Reserved Word Conflicts

Sybase uses some words as database identifiers If **sybinit** detects any reserved words in your databases, it will not continue the upgrade until these conflicts are resolved. Before you begin upgrading SQL Server, first make sure that there are no reserved word conflicts in your databases. You can either use **sybinit** to run **sp_checkreswords** to make sure there are no reserved word conflicts, or you can run it through **isql**.

Installing and Running *sp_checkreswords* with *sybinit*

sybinit installs **sp_checkreswords** and performs a reserved word check during the upgrade process. The **sybinit** reserved word check makes sure that none of your existing databases use reserved words as

identifiers. **sybinit** then displays the number of reserved word conflicts on the screen and lists the identifiers that conflict with reserved words and their owners in the file *SYBASE_SYSTEM:[SYBASE.INIT.LOGS]CHECKRES.DMP*.

You can use **sybinit** to check for reserved words either before or during an upgrade session. Because **sybinit** always checks for reserved word conflicts during the upgrade session, the first option may take less time than installing **sp_checkreswords** before the upgrade session, because you run **sp_checkreswords** only once.

To use **sybinit** to run **sp_checkreswords**, follow the instructions in this chapter, beginning with "Starting sybinit" on page 5-14, and continuing through "Reserved Word Check" on page 5-18. Skip "Pre-Upgrade Eligibility Test" on page 5-17.

Installing and Running *sp_checkreswords* Manually

You can simplify the upgrade process if you find and eliminate any identifiers in existing databases that conflict with new reserved words before you begin the **sybinit** upgrade session.

To install **sp_checkreswords** manually:

1. After you have loaded the SQL Server release 11.0.x software from media, install **sp_checkreswords** by running the **installupgrade** script:

```
isql/user="sa" /password="PASS" /server="servername"
/input=SYBASE_SYSTEM:[SYBASE.SCRIPTS]INSTALLUPGRADE
```

2. Run **sp_checkreswords** on each database to find any reserved word conflicts:

```
1> use database_name
2> go
1> sp_checkreswords
2> go
```

   sp_checkreswords displays any conflicting identifiers and their owners on the screen. You can redirect this output to a file.

3. See "Eliminating Reserved Word Conflicts" on page 5-5, or the **sp_checkreswords** entry in the *SQL Server Reference Manual* to change identifiers that conflict with reserved words.

Eliminating Reserved Word Conflicts

Reserved word conflicts involving database names cause your upgrade to fail. Conflicts with other object names do not prevent a successful upgrade.

To avoid problems, change the identifiers that conflict with reserved words as soon as possible. Also, change any procedures, triggers, views, or applications that include reserved words. The following sections describe two methods for avoiding reserved word conflicts. Refer to the *SQL Server Reference Manual* for more information.

Changing the Names of Identifiers That Are Reserved Words

You can use **sp_rename** or **sp_renamedb** to change the names of identifiers that are reserved words. In a few cases, you can perform updates to the system tables to change the names of identifiers.

Using Quoted Identifiers

For tables, views, and columns, enclose the identifiers that are reserved words in double quotes. Then use the **quoted_identifier** option of the **set** command in procedures and queries that include the reserved words. The **set quoted_identifier** option tells SQL Server to treat any character string enclosed in double quotes as an identifier. To avoid reserved word conflicts, invoke this option in all stored procedures and queries that include reserved words.

Do not use quoted identifiers for objects other than tables, views, or columns. Change the names of those objects instead of using quoted identifiers.

Backing Up Existing Databases

To speed recovery following an upgrade failure, use the **disk mirror** command to mirror your existing database devices. See the *System Administration Guide* for information on disk mirroring. Disable mirroring before beginning the upgrade.

Use the **dump database** command to make backup copies of all of your databases. If you do not back up your databases, and the upgrade procedure fails, you have no way of restoring them. If you need to restore your databases, use the database dumps you completed before beginning the upgrade procedure.

See the *SQL Server Reference Manual* for information on backing up databases.

Re-creating Database Objects

The upgrade of SQL Server to release 11.0.x involves the remapping of stored procedures, triggers, rules, defaults and views to fit the new format for query trees. If remapping fails for any database objects, you may need to drop and re-create them after the upgrade.

1. Retain your source information so you have the ability to re-create these database objects, if necessary.
2. Check off "Able to re-create database objects if upgrade fails" on the checklist.

Checking the Location of the Runserver File

sybinit copies the runserver file from the old *SYBASE_SYSTEM:[SYBASE.INSTALL]* directory to the new *SYBASE_SYSTEM:[SYBASE.INSTALL]* directory (described in the following figure):

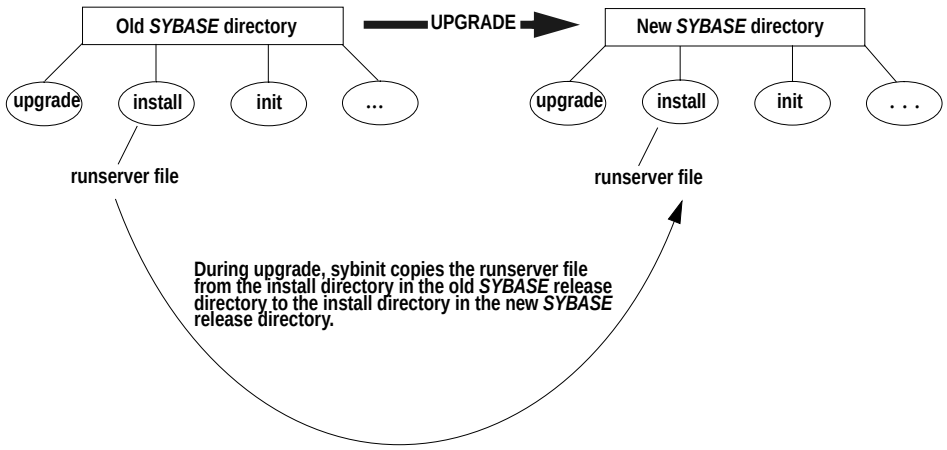


Figure 5-1: Copying the runserver file during upgrade

Make sure that the SQL Server runserver file resides in the *install* subdirectory of the release directory for your current SQL Server installation (recorded on your worksheet as the "old" *Sybase* directory.

➤ ***Note***

In pre-System 10 SQL Server, the runserver file for a server named Sybase is named *RUNSERVER*. If your runserver file is named *RUNSERVER*, **sybinit** renames it to *RUN_SYBASE* during the upgrade.

If you selected another name for the SQL Server, the file is named *RUN_servername*, where *servername* is the name of the SQL Server. This name does not change during the upgrade.

If the runserver file is not located in the old release directory, place a copy of it there now. If **sybinit** is unable to locate your SQL Server runserver file during the upgrade session, it prompts you for the information normally found in that file. This information includes the physical path of the SQL Server device and the SQL Server error log file.

Adding *SYBASE_SYSTEM* Definition to Runserver File

In order for the upgrade to succeed, the runserver file for your existing SQL Server must contain the *SYBASE_SYSTEM* definition for your existing installation. The runserver file may not contain this definition, particularly if the SQL Server is a primary installation.

The definition in the runserver file looks like the following example:

```
$DEFINE SYBASE_SYSTEM ETF2
$DEFINE SYBASE SYBASE_SYSTEM:[SYBASE]
$DEFINE DSLISTEN “MUDHEN”
$SERVER :==$SYBASE_SYSTEM:[SYBASE.BIN]DATASERVER.EXE

$SERVER /DEVICE=[E11:[TS1.DB492]MASTER.DAT]
/ERRORFILE=E11:[TSI.DB492]ERRRORLOG /INTERFACES=SYBASE_SYSTEM:[SYBASE]
```

1. Check your runserver file for the *SYBASE_SYSTEM* definition. Add the definition, if necessary.
2. Check off “Runserver file contains *SYBASE_SYSTEM* definition” on your worksheet.

Making Sure *sybsystemprocs* Is 21MB or Higher

A new *sybsystemprocs* database is created during the upgrade process if you are upgrading from SQL Server release 4.9.2. If you are upgrading from SQL Server release 10.0 or later, you must verify that the *sybsystemprocs* database is at least 21MB. Your own stored procedures will increase in size by 20 percent after the upgrade.

To see if *sybsystemprocs* and *sysprocsdev* are large enough to upgrade:

1. Start the earlier release of SQL Server.
2. Use **sp_helpdb** to determine the size of the *sybsystemprocs* database, in megabytes:

   ```
   1> sp_helpdb sybsystemprocs
   2> go
   ```

   If the size of *sybsystemprocs* is less than 21MB, or if your own stored procedures are included in *sybsystemprocs*, you need to increase the size of the database using the **alter database** command. See "Increasing Space on sybsystemprocs for Your Stored Procedures" on page 5-8 to determine the amount of space your own stored procedures will occupy after the upgrade.

Increasing Space on *sybsystemprocs* for Your Stored Procedures

The stored procedures for release 11.0.x are 20 percent larger than the stored procedures for release 10.0. Any stored procedures that you wrote for release 10.0 will also increase by 20 percent after the upgrade. You will have to take this extra space into account when you determine the size of *sybsystemprocs* for the upgrade. To determine the size of *sybsystemprocs*, including your stored procedures:

Run **sp_spaceused** to determine the amount of space your stored procedures currently occupy. The "reserved" column specifies the amount of space allocated to database objects, including the stored procedures. For example:

```
1> use sybsystemprocs
2> go
1> sp_spaceused
2> go
database_name                  database_size
------------------------------ -------------
sybsystemprocs                  15.0 MB
reserved        data            index_size      unused
--------------- --------------- --------------- ---------------
10240KB         6880 KB         156 KB          566 KB
(return status = 0)
```

To find the amount of space your stored procedures occupy on this system:

1. Subtract 8MB (the size of the default stored procedures in System 10) from the amount of "reserved."

   10240KB = 10MB

   10MB – 8MB = 2MB

   The stored procedures in this example occupy 2MB of space.

2. Multiply the size of your stored procedures by 1.2 to determine how much space your stored procedures will occupy in System 11:

   1.2 x 2MB = 2.4MB

3. Add this amount to the 16MB *sybsystemprocs* requires for System 11:

   21MB + 2.4MB = 23.4MB

   The *sybsystemprocs* database will occupy approximately 23.4MB of space after the upgrade. See "Increasing the Size of sybsystemprocs" on page 5-9 for information about increasing the size of *sybsystemprocs*.

   You may need to first increase the size of *sysprocsdev* before you increase the size of *sybsystemprocs*.

4. Use the **sp_helpdevice** stored procedure to determine the size of the *sysprocsdev* device, in megabytes:

   ```
   1> sp_helpdevice sysprocsdev
   2> go
   ```

   If the size is less than the amount you determined in the above equation, or if the size is not large enough to accommodate an increase of the *sybsystemprocs* database to provide more free space, you need to find space on an existing device or initialize an additional one.

Increasing the Size of *sybsystemprocs*

Use this procedure if the *sybsystemprocs* database is less than 21MB, but the *sysprocsdev* device does not need to be larger. The *sysprocsdev* device size should be at least 21MB, and large enough to accommodate increasing the *sybsystemprocs* database to 21MB.

1. Use an **alter database** command to increase the size of the *sybsystemprocs* database, for example:

```
1> use master
2> go
1> alter database sybsystemprocs on sysprocsdev = 6
2> go
```

   In this example, "sysprocsdev" is the logical name of the existing *sybsystemprocs* device, and "6" is the number of megabytes of increased space needed.

2. Shut down the SQL Server and exit **isql**.

Increasing the Size of Both *sysprocsdev* and *sybsystemprocs*

If the *sysprocsdev* device is too small, you may receive a message similar to the following when you increase the size of the *sybsystemprocs* database:

```
Couldn't find enough space on disks to extend database
sybsystemprocs
```

To increase the size of *sysprocsdev*:

1. Find another existing device that has enough additional free space (for example, 6MB, if your *sysprocsdev* device is 15MB) or use a **disk init** command like the following to create an additional device for *sybsystemprocs*:

```
1> use master
2> go
1> disk init
2> name = "sysprocsdev2",
3> physname = "SYBASE_DEVICES:[SYBASE.DEVICES]SYSPROCS.DAT",
4> vdevno = 9,
5> size = 3072
6> go
```

   where *SYBASE_DEVICES:[SYBASE.DEVICES]* is the path to your *sybsystemprocs* database. The number for *vdevno* must be available. For information about determining whether the *vdevno* is available, see the *System Administration Guide*.

   The size you provide should be the amount of additional megabytes of space needed for the device multiplied by 512, because **disk init** requires the size to be specified in 2K Sybase blocks. In this example, the size is 6MB (6 x 512 = 3072 2K blocks). See the *SQL Server Reference Manual* for more information on **disk init**.

2. Use the **alter database** command to increase the database size, placing the additional database space on the additional device:

```
1> alter database sybsystemprocs on
2> sysprocsdev2 = 6
3> go
```

   In this example, "sysprocsdev2" is the logical name of the additional device, and "6" is the number of megabytes of increased space needed.

3. Shut down SQL Server.

Re-Creating *sysprocsdev* and *sybsystemprocs*

Follow the steps below to drop the existing *sybsystemprocs* database and device and create a new device and database. Use this approach only as a last resort, because it will force you to re-create all user-defined stored procedures. The database you create is empty until you install the new version of SQL Server, which repopulates *sybsystemprocs* with the updated Sybase stored procedures.

All user-defined stored procedures in *sybsystemprocs* are deleted when you drop the *sybsystemprocs* database. You can re-create these stored procedures using your backup scripts after you re-create the database and upgrade SQL Server.

1. Use a **drop database** command to drop the existing *sybsystemprocs* database:

```
1> sp_configure "allow update", 1
2> go
1> reconfigure with override
2> go
1> use master
2> go
1> drop database sybsystemprocs
2> go
```

   If the *sysprocsdev* device file is on a file system device, delete the *sysprocsdev* device file. Do not delete the *sysprocsdev* device file if it is on a raw device:

```
1> begin transaction
2> delete from sysdevices
3> where name = "sybprocsdev"
4> go
```

If the *sysprocsdev* device file is on a file system device, delete the *sysprocsdev* device file; if the *sysprocsdev* device file is on a raw device, you do not need to delete it.

2. Shut down and reboot the SQL Server.
3. Use a **disk init** command like the following to create a new *sysprocsdev* device:

```
1> disk init
2> name = "sysprocsdev",
3> physname = "SYBASE_DEVICES:[SYBASE.DEVICES]SYSPROCS.DAT”,
4> vdevno = 1,
5> size = 10752
6> go
```

where *SYBASE_DEVICES:[SYBASE.DEVICES]* is the path to the physical device. The number for *vdevno* must be available. For information about determining whether the *vdevno* is available, see the *System Administration Guide*.

Make sure the size you provide is the amount of additional megabytes of space needed for the device multiplied by 512, because **disk init** requires the size to be specified in 2K Sybase blocks. In this example, the size is 10MB (21 x 512 = 10752 2K blocks). See the *SQL Server Reference Manual* for more information on **disk init**.

4. Use the **create database** command to create a new *sybsystemprocs* database. For example:

```
1> create database sybsystemprocs on
2> sysprocsdev = 21
3> go
```

In this example, the logical name for the new *sysprocsdev* device is *sysprocsdev* and its size is 21MB.

5. Shut down SQL Server and exit **isql**.

Although you have created the *sybsystemprocs* database, it will not contain any stored procedures until **sybinit** finishes the upgrade.

You can also manually load the stored procedures by running the **installmaster** script. For more information about the **installmaster** script see Appendix A, “Manually Completing Upgrade” and “The Scripts Directory” on page 7-5.

Removing Database Devices from *SYBASE_SYSTEM*

Your SQL Server upgrade will fail if any database devices are located on *SYBASE_SYSTEM*.

1. If the master device or master's mirror device is located on *SYBASE_SYSTEM*, change the master or mirror device specification in the runserver file to be the actual physical path rather than the logical.
2. If any other database devices are located on *SYBASE_SYSTEM*, contact Sybase Technical Support for assistance.
3. After you have ensured that no database devices are on *SYBASE_SYSTEM*, check off "Removed database devices from *SYBASE_SYSTEM*" on the worksheet.

Verifying *SYBASE_SYSTEM* Definition

Before you begin a **sybinit** session, verify that *SYBASE_SYSTEM* was defined correctly when the SYBASE software was installed from media.

1. Log in using the account you created in Chapter 2, "Preparing for New Installation or Upgrades." At the prompt, type:

   ```
   $ SHOW LOGICAL SYBASE_SYSTEM
   ```

   The value returned should match the SYBASE_SYSTEM definition for the new SQL Server installation, as recorded on your worksheet.
2. If the values do not match, type the appropriate command at the prompt:

   For a primary installation:

   ```
   $ DEFINE/SYS/EXEC SYBASE_SYSTEM ROOTED_LOGICAL
   ```

   For a secondary installation:

   ```
   $ DEFINE SYBASE_SYSTEM ROOTED_LOGICAL
   ```

   where *rooted_logical* is the *SYBASE_SYSTEM* definition for the new SQL Server installation, as recorded on your worksheet.

➤ ***Note***

You need to use two different *SYBASE_SYSTEM* logicals during an upgrade of SQL Server. You must define *SYBASE_SYSTEM* to represent the new release location before starting **sybinit.** During your **sybinit** upgrade session, the "Upgrade Existing SQL Server" menu requires you to specify the *SYBASE_SYSTEM* logical that represents the existing release location of the SQL Server to be upgraded.

Starting *sybinit*

1. Make sure the **sybinit** symbol is defined:

   $ SHOW SYMBOL SYBINIT

2. If the symbol is not defined, execute the following:

   $ @*SYBASE_SYSTEM*:[SYBASE.INSTALL]SYBASE_MANAGER

3. Start **sybinit** by typing at the operating system prompt:

   $ SYBINIT

   sybinit displays the following menu:

```
The log file for this session is
'SYBASE_SYSTEM:[SYBASE.INIT.LOGS]LOG1014.001'.

SYBINIT

1. Release Directory: SYBASE_SYSTEM:[SYBASE]
   Type of Installation:  secondary (process-level
                                  or group-level access)
2. Edit / View Interfaces File
3. Configure a Server product
4. Configure an Open Client/Server product

Ctrl-a Accept and Continue, Ctrl-x Exit Screen, ? Help.

Enter the number of your choice and press return:
```

➤ ***Note***

The "Type of Installation" item is for display only. See "Determining Installation Type" on page 2-2 for an explanation of the difference between primary and secondary installations.

4. Select Configure a Server Product from the main Sybinit menu. **sybinit** displays the Configure Server Products menu:

```
CONFIGURE SERVER PRODUCTS

    Product          Date Installed   Date Configured
1. SQL Server        Oct 16 96 16:33
2. Backup Server     Oct 16 96 16:34
```

The "Date Installed" column indicates when **VMSINSTAL** loaded the product files onto disk. The "Date Configured" column indicates when you configured the product with **sybinit**. When you complete the SQL Server upgrade, **sybinit** records the current date and time for the SQL Server "Date Configured" column.

5. Select SQL Server. **sybinit** displays the following menu:

```
NEW OR EXISTING SQL SERVER

1. Configure a new SQL Server
2. Configure an existing SQL Server
3. Upgrade an existing SQL Server
```

6. Select Upgrade an Existing SQL Server.

Setting the Upgrade Release Directory

sybinit displays the following menu:

```
UPGRADE EXISTING SQL SERVER

1. Release directory for previous release SYBASE_SYSTEM:[SYBASE110]
```

The displayed release directory is the SQL Server directory where you loaded your new software (listed on your worksheet as the **new** SYBASE environment variable). Change this setting to display the path of the current release directory (listed on your worksheet as the **old** SYBASE logical name):

1. Select Release Directory for Previous Release.
2. At the prompt, enter the release directory for your current SQL Server installation (listed on your worksheet as the old SYBASE logical name).
3. Press Ctrl-a when the displayed path is correct.
4. Record the release directory for the previous release in the worksheet.

Selecting a SQL Server to Upgrade

sybinit displays the names of existing SQL Servers as options. This list includes Backup Servers.

```
CONFIGURE EXISTING SQL SERVER

1. PUB
2. MARKET
3. SYBASE_491
```

1. If the entire list of servers does not fit on one screen, press Ctrl-f to scroll to the end of the list. **sybinit** does not allow you to select a server until you have reached the end of the list.
2. Select the SQL Server to upgrade and record the name after "Name of existing SQL Server to upgrade" in the worksheet.

Entering "SA" Information

sybinit displays the next menu:

```
ENTER SA ACCOUNT NAME AND PASSWORD

1. SA Account: "sa"
2. SA Password:
```

1. Select SA Account and type the name of the account as recorded on your worksheet. **sybinit** displays the name in the Enter SA Account Name and Password menu.
2. Select SA Password and type the password for the account as recorded on your worksheet.

   For security reasons, **sybinit** substitutes pound signs (#) for characters when it displays the password in the menu.
3. Press Ctrl-a to accept the account name and password.
4. Record the System Administrator account name and password in the worksheet.
5. If SQL Server is not running, **sybinit** displays the message:

```
The SQL Server is down. Would you like to boot it now? y
```

   Press Return to boot SQL Server or "n" to return to the New or Existing SQL Server menu.

Pre-Upgrade Eligibility Test

sybinit displays the SQL Server Upgrade menu:

```
SQL SERVER UPGRADE

1. Test SQL Server upgrade eligibility now
2. Check for reserved word conflicts
3. sybsystemprocs database configuration
4. Upgrade SQL Server now
```

Select Test SQL Server Upgrade Eligibility Now. **sybinit** examines your existing SQL Server to determine if your databases:

- Contain enough free space to complete the upgrade
- Have any options set that would prevent a successful upgrade
- Contain any inconsistencies that would prevent a successful upgrade

Passing the Pre-Upgrade Test

If SQL Server passes the pre-upgrade test, **sybinit** displays the following message:

```
Server SERVER_NAME passed preupgrade eligibility
test.
```

Press Return to return to the "SQL Server Upgrade" menu.

Failing the Pre-Upgrade Test

If SQL Server fails the pre-upgrade test, **sybinit** displays the message:

```
Server SERVER_NAME failed preupgrade eligibility
test. See log for more information.
```

If you receive this message, follow this procedure:

1. Examine the log file created in the *SYBASE_SYSTEM*:[*SYBASE.INIT.LOGS*] directory to obtain information about why SQL Server failed the pre-upgrade eligibility test. The name of this log is displayed on the screen when you enter and exit **sybinit**. See Appendix B, "sybinit Resource Files" for more information on log files.
2. Correct the problem based on the messages below:

- If the log contains messages about insufficient space, use the Transact-SQL **alter database** command to increase the space in your databases:

  ```
  alter database database_name on device_name = x
  ```

  where *database_name* is the name of your database, *device_name* is the name of the database device, and *x* is the number of megabytes of increased space.

- If the log contains messages about database options and includes the phrase "You must reset this," use the Transact-SQL system procedure **sp_dboption** to change the settings for the database options noted. See Volume 2 of the *SQL Server Reference Manual* for information about **sp_dboption**.

- If the log contains messages about database inconsistencies and includes the phrase "You must resolve this," you must resolve the problems for all databases in the SQL Server before any databases can be upgraded.

  You may be able to resolve some problems by running **dbcc** commands which have the **fix** option, or by referring to the *SYBASE Troubleshooting Guide*. Sometimes the only solution is to drop the problem database. If necessary, contact Sybase Technical Support for assistance.

3. After you have resolved any problems, return to your **sybinit** upgrade session and re-run the pre-upgrade eligibility test.

Reserved Word Check

Select Check for Reserved Word Conflicts from the SQL Server Upgrade menu to see if any of your existing databases use 11.0.x Transact-SQL keywords as identifiers.

Reserved Word Conflicts

If your existing databases use any Transact-SQL keywords as identifiers, **sybinit** displays the following message, including the number of conflicts it found:

```
Warning: x conflicts with 11.0 reserved words were
found. Sybase suggests that you resolve these
conflicts before upgrading the SQL Server. Run
'sp_checkreswords' on each database for more info.
```

sybinit writes a list of the identifiers that conflict with reserved words and their owners to the file *SYBASE_SYSTEM:[SYBASE.INIT.LOGS]CHECKRES.DMP*. Review the contents of this file and follow the instructions in Volume 2 of the *SQL Server Reference Manual* to change these identifiers.

Reserved word conflicts involving database names prevent the upgrade process from completing. You must eliminate all reserved word conflicts with database names before you upgrade SQL Server.

Conflicts with other object names do not prevent the upgrade process from completing. However, applications that reference these conflicting object names may no longer work after the upgrade. To ensure that all of your applications work, it is important to eliminate the conflicts with object names as soon as possible.

No Reserved Word Conflicts

If **sybinit** does not find any reserved word conflicts, the following message appears on the screen:

```
Reserved word check passed, there are no conflicts
with 11.0 SQL Server reserved words.
```

Press Return to return to the SQL Server Upgrade menu.

Configuring *sybsystemprocs*

Select Sybsystemprocs Database Configuration from the SQL Server Upgrade menu to enter configuration information for this database and its device. **sybinit** moves to the following menu:

```
SYBSYSTEMPROCS DATABASE CONFIGURATION

1. sybsystemprocs database size (Meg): 21

2. sybsystemprocs logical device name: sysprocsdev

3. create new device for the sybsystemprocs database: yes

4. physical name of new device:
SYBASE_SYSTEM:[SYBASE]sybprocs.dat

5. size of the new device (Meg): 21
```

Entering Database Size

To change the size of the *sybsystemprocs* database, follow these steps:

1. Select Sybsystemprocs Database Size from the Sybsystemprocs Database Configuration menu.
2. At the prompt, enter the database size and record it on your worksheet. The size must be at least 21MB. **sybinit** redisplays the Sybsystemprocs Database Configuration menu with the new database size.

Creating a New Device

To place the *sybsystemprocs* database on an existing device, see "Using an Existing Device" on page 5-20. To create a new device, follow these steps:

1. To change the default logical name, select Sybsystemprocs Logical Device Name. At the prompt, type the logical device name as recorded on the worksheet.
2. If **sybinit** does not display "yes" at Create New Device for the Sybsystemprocs Database, select this option to change the displayed value. **sybinit** toggles the setting to "yes."
3. To use a physical device other than the displayed default, select Physical Name of New Device.

   At the prompt, type the full path to use for the *sybsystemprocs* device, as recorded in *sybsystemprocs* logical device name on the worksheet.
4. If you typed the path of an operating system file for the *sybsystemprocs* device, select Size of the New Device to change the displayed default. At the prompt, type the device size, as recorded on your worksheet.
5. Press Ctrl-a to accept the settings displayed on the Sybsystemprocs Database Configuration menu. **sybinit** redisplays the SQL Server Upgrade menu and marks the Sybsystemprocs Database Configuration task "Complete."

Using an Existing Device

If you created a new device for *sybsystemprocs*, go to "Executing the Upgrade" on page 5-21. To place the *sybsystemprocs* database on an existing device, follow these steps:

◆ ***WARNING!***

Do not put the *sybsystemprocs* database on the master device.

1. Select Sybsystemprocs Logical Device Name from the Sybsystemprocs Database Configuration menu.
2. At the prompt, type the name of the device to use, as recorded under *sybsystemprocs* logical device name on the worksheet.
3. If **sybinit** does not display "no" for Create New Device for the Sybsystemprocs Database, select the option to change the displayed value. **sybinit** toggles the setting to "no" and removes the remaining options from the menu.
4. Press Ctrl-a to accept the values displayed in the Sybsystemprocs Database Configuration menu. **sybinit** returns to the SQL Server Upgrade menu.

Executing the Upgrade

When you are ready to run the upgrade:

1. Select Upgrade SQL Server Now.

   sybinit prompts you to specify a path for the SQL Server error log file. The default path is *SYBASE_SYSTEM:[SYBASE.INSTALL]ERRORLOG*, where *SYBASE_SYSTEM* is the logical definition for the upgraded SQL Server.
2. Type in a different path at the prompt or press Return to accept the default.

◆ ***WARNING!***

Do not interrupt your upgrade while it is in progress. In particular, do not run any stored procedures or attempt to connect to your server. If you want to check the progress of your upgrade, check the sybinit log.

sybinit displays a series of messages as it executes a number of tasks, including the following:

- Saves a log of the upgrade process to the file *SYBASE_SYSTEM:[SYBASE.UPGRADE]SERVER_NAME.LOG*
- Saves previous configuration settings (for pre-10.0 SQL Server) to the file

*SYBASE_SYSTEM:[SYBASE.UPGRADE]SERVER_NAME*_CSAVE.*OUT*

- Checks current stack size and memory and if less than default values, sets sizes to default and logs message
- Shuts down pre-11.0.x SQL Server
- Boots new SQL Server
- Issues SQL statements that modify SQL Server so that it conforms to the rules for SQL Server release 11.0.x.
- Saves a log of configuration changes to the file *SYBASE_SYSTEM:[SYBASE.UPGRADE]SERVER_NAME*_CONFIG.*OUT*
- Updates version number of SQL Server
- Remaps query trees
- Runs **installmaster** and **installmodel** scripts
- Saves details on upgrade status to the file *SYBASE_SYSTEM:[SYBASE.UPGRADE]SERVER_NAME*_UPGRADE.*OUT*

3. See Appendix B, "sybinit Resource Files," if any error messages appear or the upgrade fails

Quitting *sybinit*

sybinit displays the following message after completing an upgrade:

```
Configuration completed successfully.
Press <return> to continue.
```

Press Return to return to the SQL Server Upgrade menu. To exit **sybinit**, press Esc until you have backed out of the submenus and returned to the Sybinit menu.

Remapping Query Trees

When **sybinit** returns to the SQL Server Upgrade menu after the upgrade has completed, a new option, Remap the Query Trees in All Databases, appears:

```
SQL SERVER UPGRADE
1. Test SQL Server upgrade eligibility now
2. Check for reserved word conflicts
3. sybsystemprocs database configuration
4. Upgrade SQL Server now
5. Remap the query trees in all databases
```

Select this option to remap any query trees that the upgrade did not successfully remap. Remapping results are saved to the file *SYBASE_SYSTEM:[SYBASE.INIT.LOGS]REMAP.DMP*. Review the contents of this file to see if remapping failed again for any database objects.

◆ ***WARNING!***

If remapping failure occurs, you must complete certain upgrade tasks manually. You may need to drop and re-create some database objects. See Appendix A, "Recovery from Failure" for procedures to follow after remapping failures.

Ending Upgrade Session

If you plan to configure Backup Server, or to reconfigure SQL Server, press Ctrl-x once to go to the Configure Server Products menu. To exit **sybinit**, press Ctrl-x repeatedly until to the operating system prompt returns.

Moving Stored Procedures to *sybsystemprocs*

After the upgrade, the *sybsystemprocs* database contains most of the system procedures that were stored in the *master* database for pre-11.0.x SQL Server. The system procedures needed during difficult recovery situations remain in *master*. Any stored procedures created locally in *master* also remain there after the upgrade.

If you want to move your locally created stored procedures to *sybsystemprocs*, check them carefully for references to system tables in *master*. You must fully qualify any table names that appear in your locally created system procedures because these procedures will reside in a different database than the system tables.

A stored procedure in *sybsystemprocs* should never modify the tables in *master* inside a transaction, as this could create problems with recovery. The *master* database must be fully recovered before any other databases can be recovered. Open transactions in *sybsystemprocs* that involve *master* could prevent full recovery of *master*.

To move your procedures to *sybsystemprocs*, create the procedures in the new database and drop them from *master*. Refer to Volume 1 of the *SQL Server Reference Manual* for more information on creating and dropping procedures.

Checking Data Cache Memory

SQL Server release 11.0.x uses more memory for code and internal structures than SQL Server release 10.0. The exact amount varies depending on your configuration. For example, the amount of memory required per user is slightly larger in this release, so if you have configured SQL Server for a large number of users, you may see a greater increase in the amount of memory required.

If you have sized your data cache to provide optimum performance for your release 10.0 applications, you may need to increase the amount of memory available to SQL Server release 11.0.x. For example, the decrease in cache size may mean that tables that fit comfortably into the cache in release 10.0 will no longer fit completely into the cache in release 11.0.x. You will see a substantial increase in disk I/O if this is the case.

Increasing Data Cache Space

Use the following steps to increase the amount of memory available to your server:

1. Check the error log for your SQL Server release 10.0. The number of 2K pages is listed in the message:

```
Number of buffers in buffer cache: 20514
```

This server has 20514 pages, or 41028K, of memory available for the data cache.

2. Log into your SQL Server release 11.0.x using **isql** and run the **sp_configure** command to find the total data cache size for the 11.0.x server:

```
1> sp_configure 'total data cache size'
2> go
Parameter Name       Default  Memory Used  Config Value  Run Value
-------------------- -------- -----------  -----------   -----------
total data cache size       0        38580            0          38580
```

3. To get approximately the same amount of data cache on your upgrade SQL Server, you need to add 2448K of memory (41028K – 38580K = 2448K, or 1224 2K pages) to the data cache.
4. Check the current amount of memory allocated (described by the "Run Value"). The value is reported in 2K pages:

```
1> sp_configure 'total memory'
2> go
Parameter Name       Default  Memory Used  Config Value  Run Value
-------------------- -------- -----------  -----------   -----------
total memory             7500        60000        30000          30000
```

5. Check the procedure percentage on your SQL Server:

```
1> sp_configure 'procedure cache percent'
2> go
Parameter Name        Default   Memory Used  Config Value  Run Value
--------------------  --------- -----------  -----------   ----------
procedure cache percent      20       10010           20            20
```

This SQL Server allocates 20 percent of its memory to the data cache.

6. Multiply the amount of memory you want to add to the data cache by 20 percent ($\frac{5}{4}$):

   1224 x $\frac{5}{4}$ = 1530

   Your SQL Server may require a different percentage to achieve optimal performance. Replace the $\frac{5}{4}$ with the correct multiple for your system configuration.

7. Add the additional value to the amount reported:

   30000 + 1530 = 31530 pages

8. Execute **sp_configure** with the new value:

   ```
   sp_configure 'total memory', 31530
   ```

You must restart SQL Server for this change to take effect.

You may need to reconfigure your VMS system quotas and parameters. See "Summary of Reconfiguration Steps" on page 8-6 for more information about reconfiguring VMS system quotas and parameters.

Increasing the Procedure Cache

The size of query plans increased about 20 percent between release 10.0.x and release 11.0.x.

If you have sized your procedure cache to match your procedure workload, you may need to increase its size. You may want to increase both the total memory and the percent of memory allocated to the procedure cache.

If a user executes a procedure and all space in the procedure cache is in use by other active procedures, users receive error message 701:

```
There is not enough procedure cache to run this procedure,
trigger, or SQL batch. Retry later, or ask your SA to
reconfigure SQL Server with more procedure cache.
```

If there is not enough procedure cache space for an individual procedure, users receive error message 703:

```
You cannot run this procedure, trigger, or SQL batch because it
requires more than %ld pages of memory. Break it up into shorter
queries, if possible.
```

See the *System Administration Guide* for more information on configuring memory.

Backing Up New Software

When the upgrade is successful, make a backup copy of your upgraded databases. You must configure a Backup Server before you can dump or load databases on your upgraded SQL Server.

New Backup Server

You must install a new Backup Server before you can dump and load databases on your upgraded SQL Server. Upgrading SQL Server does not automatically upgrade Backup Server, and there is no specific process for upgrading Backup Server. You must install a **new**

Backup Server after you upgrade SQL Server from System 10 to release 11.0.x

What's Next?

Chapter 6, "Installing Backup Server" provides instructions for configuring a new Backup Server.

Chapter 8, "Reconfiguring SQL Server and Backup Server" provides instructions for reconfiguring your upgraded SQL Server and for reconfiguring an existing Backup Server.

Chapter 7, "Getting Started with SQL Server and Backup Server," discusses starting and monitoring SQL Server and Backup Server.

6 Installing Backup Server

This chapter provides instructions for using **sybinit** interactively to configure a new Backup Server. Topics covered are:

See Appendix B, "sybinit Resource Files" for information on using a **sybinit** resource file to configure a Backup Server.

➤ ***Note***

The Backup Server for a SQL Server must be the same release as the SQL Server. After upgrading SQL Server to release 11.0.x, shut down your old Backup Server and install a new Backup Server release 11.0.x before making a backup copy of your upgraded databases. Upgrading the SQL Server does not automatically upgrade the Backup Server.

Before you configure Backup Server, see the *Release Bulletin* for information about compatibility between different releases of SQL Server and Backup Server.

Understanding the Naming of Backup Server

The system default name for Backup Server is SYB_BACKUP. When installing SQL Server and Backup Server, you can choose to use the default name or specify a different name (for example, FLUFFY_BACKUP).

The default name, SYB_BACKUP, is always listed in the SQL Server system tables (*srvname* column of *sysservers*). The *srvname* does not change, even if you specify a different name. SQL Server always communicates with the Backup Server using the name SYB_BACKUP because many commands for Backup Server (such as **shutdown**) require the SYB_BACKUP name.

The Backup Server name you specify, FLUFFY_BACKUP, is best thought of as an alias to SYB_BACKUP. FLUFFY_BACKUP is stored in the interfaces file and the *srvnetname* listed in *sysservers*. The *servnetname* does change if you specify a different name.

Table 6-1: Naming of Backup Server

If You Use This Name	*Srvname*	*Srvnetname*
System default name	SYB_BACKUP	SYB_BACKUP
A different name	SYB_BACKUP	FLUFFY_BACKUP

Starting *sybinit*

1. If you did not exit **sybinit** after a previous session, press Ctrl-x until the Configure Server Products menu appears.

   If you exited **sybinit**, follow the instructions in "Starting sybinit" on page 4-5 and select Configure a Server Product from the main menu.

2. At the Configure Server Products menu, select Backup Server. **sybinit** displays the following menu:

```
NEW OR EXISTING BACKUP SERVER

1. Configure a new Backup Server
2. Configure an existing Backup Server
```

3. Select Configure a New Backup Server. **sybinit** displays the Add New Backup Server menu:

```
ADD NEW BACKUP SERVER

1. Backup Server name: SYB_BACKUP
```

4. To accept the Backup Server name, SYB_BACKUP, press Ctrl-a.

   To change the name, select Backup Server Name. At the prompt, enter the new Backup Server name, as configured for SQL Server and recorded on your worksheet:

```
Enter the name of the new Backup Server (default is
'SYB_BACKUP'):
```

sybinit returns to the Add New Backup Server menu and displays the new name.

➤ ***Note***

The Backup Server name must match the default Backup Server name you enter during the SQL Server installation or reconfiguration process. Refer to your worksheet for the correct name.

5. Type Ctrl-a when Backup Server Name shows the correct name.

Selecting Error Log Location

sybinit displays the Backup Server Configuration menu:

```
BACKUP SERVER CONFIGURATION

1. Backup Server errorlog:
SYBASE_SYSTEM:[SYBASE.INSTALL]BACKUP.LOG
2. Backup Server quota file:
3. Enter / Modify Backup Server interfaces file information
4. Backup Server language: us_english
5. Backup Server character set: iso_1
```

1. To change the displayed default location of the Backup Server error log file, select "Backup Server Errorlog. "
2. At the prompt, enter the file specification of the error log file.
3. Record the file specification of the error log file on the worksheet.
4. **sybinit** returns to the Backup Server Configuration menu.

Setting Quota File Location (Optional)

1. To use a special quota file to set system parameters for Backup Server, select Backup Server Quota File.
2. At the prompt, enter the file specification of the quota file. There is no default file specification.

   To remove a quota file, enter a space at the prompt.
3. Record the file specification of the quota file on your worksheet.
4. **sybinit** returns to the Backup Server Configuration menu.

Creating Backup Server Interfaces File Entry

➤ ***Note***

For information on the format of interfaces file entries, see "Interfaces File Format" on page 9-2.

Select Enter / Modify Backup Server Interfaces File Information from the Backup Server Configuration menu. **sybinit** displays the Server Interfaces File Entry Screen:

```
SERVER INTERFACES FILE ENTRY SCREEN
   Server name: SYB_BACKUP
1. Retry Count: 0
2. Retry Delay: 0
3. Add a new listener service
```

Entering Connection Information (Optional)

1. Select Retry Count.
2. At the prompt, enter the retry count value for the Backup Server that is recorded on your worksheet.
3. Select Retry Delay.
4. At the prompt, enter the Backup Server retry delay value that is recorded on your worksheet.

The Server Interfaces File Entry Screen now displays the retry count and retry delay values that you entered (or 0 if you did not enter any values). You can change either of the displayed values by reselecting the appropriate menu item and entering a new value at the prompt.

Selecting Network Protocol

Select Add a New Listener Service. **sybinit** displays the Select Network Protocol menu:

```
SELECT NETWORK PROTOCOL
1. TCP
2. DECnet
```

Follow the instructions in whichever of the following sections apply to the network protocol(s) you select.

➤ ***Note***

The SPX protocol is not supported for Digital OpenVMS Alpha.

Adding a TCP Entry

To choose TCP, perform the following steps:

1. Select TCP from the Select Network Protocol menu. **sybinit** moves to the Edit TCP Service menu:

   ```
   EDIT TCP SERVICE

   1. Hostname/Address:
   2. Port:
   3. Name Alias:
   4. Server Type: master
   5. Delete this service from the interfaces entry
   ```

2. To change the default Backup Server host name, select Hostname / Address and enter the recorded IP address of the machine on which Backup Server is installed.
3. Select Port and enter the 4- or 5-digit port number, as recorded on your worksheet. The range of valid port numbers is from 5000 to 65535.
4. (Optional) Select Name Alias and enter the alias recorded on your worksheet.

➤ ***Note***

"Server type" refers to whether the entry is for a Master or Companion Server. This option does not apply to Backup Server.

5. Skip to "Accepting the New Entry" on page 6-6.

Adding a DECnet Entry

To choose DECnet, perform the following steps:

1. Select DECnet from the Select Network Protocol menu. The Edit DECnet Service menu appears:

   ```
   EDIT DECNET SERVICE

   1. Nodename:HARPO
   2. Object:
   3. Name Alias:
   4. Server Type: master
   5. Delete this service from the interfaces entry
   ```

2. To change the default Backup Server node name, select Nodename and enter the node name recorded on your worksheet.
3. Select Object and enter the object name recorded on your worksheet. Use a named object rather than an object number.
4. (Optional) Select Name Alias and enter the alias recorded on your worksheet.

➤ ***Note***

"Server type" refers to whether the entry is for a Master or Companion Server. This option does not apply to Backup Server.

Accepting the New Entry

1. Verify that all entries in the Edit [Protocol] Service menu (for the protocol you selected) are correct. If necessary, repeat the preceding instructions to correct an entry.
2. Press Ctrl-a to accept the displayed information. **sybinit** asks if the information is correct:

   ```
   Is this information correct? n
   ```

3. Type "y" to proceed with the installation. **sybinit** returns to the Server Interfaces File Entry Screen:

```
SERVER INTERFACES FILE ENTRY SCREEN

   Server name: SYBASE

1. Retry Count: 5
2. Retry Delay: 5

3. Add a new listener service

Modify or delete a service

Listener services available:

   Protocol Address  Port    Name Alias
4. tcp   harpo      5600    131.214.100.59
```

sybinit displays the listener information you entered under the heading "Listener services available." You can modify or delete this information at any time by selecting the item number of the listener service (item 4 in the example above).

You can also create additional listener entries for the server by repeating the instructions in "Selecting Network Protocol" on page 6-4. This ability to create multiple entries is especially useful for Backup Servers that operate on networks with multiple protocols.

4. Press Ctrl-a to record the new information in the interfaces file. **sybinit** prompts you to write the changes to the file:

   ```
   Write the changes to the interfaces file now? n
   ```

5. Type "y" to write the information to the interfaces file. **sybinit** returns to the Backup Server Configuration menu.

Configuring Backup Server Language

In the Backup Server Configuration menu, **sybinit** displays the default language for new Backup Server installations, U.S. English.

1. To change Backup Server's language, select this option. **sybinit** displays the Backup Server Language menu:

```
BACKUP SERVER LANGUAGE

   Current Backup Server language:  us_english

1. us_english
2. french
3. german
4. spanish
```

(The displayed language options may vary, according to which Language Modules you have installed.)

2. Select the language option for Backup Server to use. **sybinit** displays this option in the menu as the Current Backup Server Language. Record the Backup Server language on the worksheet.
3. Press Ctrl-a to accept the displayed Backup Server language. **sybinit** returns to the Backup Server Configuration menu.

Configuring Backup Server Character Set

In the Backup Server Configuration menu, **sybinit** displays the default character set for new Backup Server installations, iso_1.

1. To change Backup Server's character set, select this option. **sybinit** displays the Backup Server Character Set menu:

```
BACKUP SERVER CHARACTER SET

   Current Backup Server character set:  iso_1

1. ascii_8: ASCII, for use with unspecified 8-bit data.
2. iso_1: ISO 8859-1 (Latin-1_ - Western European 8-bit character
3. roman8: Hewlett_Packard proprietary character set for European
4. cp850: Code Page 850 (Multilingual) character set.
5. cp437: Code Page 437, (United States) character set.
6. mac: Macintosh default character set for Western European loca
```

2. Select the character set option for Backup Server to use. **sybinit** displays this option in the menu as the Current Backup Server Character Set. Record the Backup Server character set on the worksheet.
3. Press Ctrl-a to accept the displayed Backup Server character set. **sybinit** returns to the Backup Server Configuration menu.

Completing Backup Server Configuration

Press Ctrl-a at the Backup Server Configuration menu after you have completed the required configuration tasks. **sybinit** asks if you want to execute the configuration:

```
Execute the Backup Server Configuration now? n
```

At this point you can:

- Type “n” for “no” to return to the Backup Server Configuration menu.
- Type “y” for “yes” to continue with the configuration.

If you continue the configuration, **sybinit** performs a series of tasks, including:

- Creating the Backup Server’s runserver file
- Booting the Backup Server

Quitting *sybinit*

When the configuration is successful, **sybinit** displays the following message:

```
Configuration completed successfully.
Press <return> to continue.
```

Press Return to return to the New or Existing Backup Server menu after completing these tasks. To exit **sybinit**, press Ctrl-x repeatedly until the operating system prompt reappears.

Backing Up New Software

After you have successfully configured a new Backup Server, use the **dump database** command to back up your new or upgraded SQL Server installation. See the *System Administration Guide* for more information on dumping databases.

Shutting Down Backup Server

To shut down a Backup Server, specify the Backup Server name with the Transact-SQL **shutdown** command:

1. Use **isql** to log in to a SQL Server account with System Administrator privileges:

```
$ ISQL /USER="SA" /PASSWORD="PASSWORD" /SERVER="SERVER_NAME"
```

2. Enter the following command to shut down the specified Backup Server:

```
1> shutdown SYB_BACKUP
2> go
```

When you use the **shutdown** command to stop a Backup Server, you must specify the *srvname* listed in *sysservers* for Backup Server. A SQL Server's default Backup Server's *srvname* is SYB_BACKUP.

Refer to the *SQL Server Reference Manual* for more information on the **shutdown** command.

➤ ***Note***

After you shut down a Backup Server, you must wait at least 30 seconds before rebooting it.

What's Next

To reconfigure a SQL Server or Backup Server, see Chapter 8, "Reconfiguring SQL Server and Backup Server" for instructions.

Chapter 7, "Getting Started with SQL Server and Backup Server," discusses starting and monitoring SQL Server and Backup Server.

7 Getting Started with SQL Server and Backup Server

This chapter provides instructions for using SQL and Backup Servers. Topics covered include:

Case Sensitivity

Command line parameters in the Sybase utilities are case-sensitive. You must enclose lowercase character string parameters in double quotes. If you do not, they are automatically converted to uppercase by Digital OpenVMS Alpha. For example, to log in to **isql** as "bob," type:

```
$ isql /user="bob"
```

Starting, Monitoring, and Shutting Down Servers

You can start (or boot), shut down, and restart SQL Server or Backup Server using operating system commands. See the *SQL Server Utility Programs for Digital OpenVMS Alpha* manual for more information on the Sybase utilities that perform these functions.

Starting Servers at the Command Line

This section discusses how to use runserver files to start SQL Server or Backup Server at the command line. It also describes how to move SQL Server's shared memory files by adding a flag to the runserver file.

◆ ***WARNING!***

Never delete the runserver file that is created in *SYBASE_SYSTEM:[SYBASE.INSTALL]*. sybinit uses this file to boot servers when you customize your installation. If you need the runserver file in another location, make a copy of the original.

Using *startserver* to Start Servers

The following command starts SQL Server from the command line:

```
startserver {/server=SERVER_NAME]
```

where *SERVER_NAME* is the name of the server you are starting. The **startserver** utility is in the *SYBASE_SYSTEM:[SYBASE.INSTALL]* directory. **startserver** runs SQL Server as a detached process by executing *RUN_SERVER_NAME.COM* in *SYBASE_SYSTEM:[SYBASE.INSTALL]*

Each time a new SQL Server or Backup Server is installed, the **sybinit** program creates a *runserver* file that contains the information required to start that particular server. The *runserver* file is named *RUN_SERVERNAME.COM*, where *servername* is the name of server.

For example, the runserver file name for a SQL Server named TEST is *RUN_TEST.COM*, and the *runserver* file name for a SQL Server named PRACTICE is *RUN_PRACTICE.COM*. *runserver* files are created in the *SYBASE_SYSTEM:[SYBASE.INSTALL]* directory.

You must be the owner of the master device for a SQL Server before you can start it with **startserver.** See *Utility Programs for OpenVMS* for more information on **startserver.**

Runserver File and Upgrades

During an upgrade, the runserver file is copied from the old Sybase release directory to the new Sybase release directory. The new runserver file is reconfigured to start SQL Server from the new release directory. The runserver file in the old Sybase release directory is left intact.

Starting Servers During System Restart

If you want SQL Server, Backup Server to restart automatically when your system restarts, you can edit the *SYS$MANAGER: SYSTARTUP_VMS.COM* file for the particular node, add definitions for the *SYBASE* logicals and the **startserver** symbol, and invoke the **startserver** command. For example:

```
$ DEFINE/SYSTEM/TRANSLATION=CONCEALED -
_$ SYBASE_DEVICE DISK$PROD_1:[USER.]
$ DEFINE/SYSTEM SYBASE_SYSTEM SYBASE_DEVICE
$ DEFINE/SYSTEM SYBASE SYBASE_SYSTEM:[SYBASE]
$!
$ STARTSERVER :== $SYBASE_SYSTEM:[SYBASE.INSTALL]STARTSERVER
$!
$ STARTSERVER/SERVER="SERVER_NAME"
```

For Backup Server, substitute a command like the following for the last line in the preceding example:

```
$ STARTSERVER/BACKUPSERVER="BACKUP_SERVER_NAME"
```

Alternatively, you can use the Digital OpenVMS Alpha **SYSMAN** utility. See the *OpenVMS SYSMAN Utility Manual*, part of the Digital OpenVMS Alpha System Management documentation set, for more information.

Shutting Down Servers

Use a command like the following to stop a SQL Server:

```
$ STOPSERVER /SERVER="SERVER_NAME"
```

Use a command like the following to stop a Backup Server:

```
$ STOPSERVER /BACKUPSERVER /SERVER="BACKUP_SERVER_NAME"
/SERVER="SERVER_NAME" /ISER=SA" /PASSWORD="SA_PASSWORD"
```

stopserver prompts for the password for a user account with System Administrator privileges. If for some reason you cannot connect to the server, you can use the **/abort** qualifier with **stopserver**. Use this qualifier only in case of emergency. **stopserver** is designed to take the server down gracefully.

Monitoring Servers

The simplest way to tell whether a SQL Server is running is by logging in to it using **isql** or Data Workbench.

You can also run the **showserver** command to determine whether any SQL Server processes are running on the system:

```
$ SHOWSERVER /SERVER=SERVER_NAME
```

The **SHOW PROCESS** DCL command can also be used:

```
$ SHOW PROCESS SERVER_NAME_SQL
```

Note that you can use a similar command to determine whether any Backup Server processes are running on the system:

```
$ SHOW PROCESS SERVER_NAME_BSV
```

For example, if you type the following command with the server Sybase running:

```
$ SHOW PROCESS SYBASE_SQL
```

the output looks like the following:

```
29-OCT-1991 11:25:37.73  User: SYBASE    Process ID:   40801CC2
                    Node: STAR      Process name: "SYBASE_SQL"

Terminal:
User Identifier:    [230,17]
Base priority:      4
Default file spec:  Not available

Devices allocated:  NET5217:
                    NET5218:
```

To find out whether a server has multiple engines running, use the following command:

```
$ SHOW PROCESS SERVER_NAME_SQL /SUBPROCESSES
```

For example, if you type the following command with the server Sybase running:

```
$ SHOW PROCESS SYBASE_SQL/SUBPROCESSES
```

the output looks like the following:

```
29-OCT-1991 11:29:08.82  User: SYBASE  Process ID:   40801CC2
                        Node: STAR    Process name: "SYBASE_SQL"

    There are 3 processes in this job:

      SYBASE_SQL (*)
      SYBASE1
      SYBASE2
```

In the above example, the server Sybase has three engines running, with the process names SYBASE_SQL, SYBASE1, and SYBASE2.

You can also use the **SHOW SYSTEM** DCL command to see what SQL Server processes are running. See Chapter 9, "Administrative Tasks and Performance and Tuning," for an example of **SHOW SYSTEM** output.

Logging Errors

Most error messages from Sybase SQL Server and Sybase SQL Toolset™ go to the user's terminal only. Backtraces from SQL Server, fatal error messages (severity levels 19 and above), kernel error messages, and informational messages from SQL Server are sent to an error log file.

You specify the paths for SQL Server's and Backup Server's error log files during the sybinit installation session. By default, sybinit creates SQL Server's error log file as *SYBASE_SYSTEM:[SYBASE.INSTALL]ERRORLOG* and Backup Server's error log file as *SYBASE_SYSTEM:[SYBASE.INSTALL]BACKUP.LOG*. The location of a server's error log file is set in its runserver file.

The error log files are owned by the System Administrator (or the user who ran the **sybinit** installation session). You must have write access to a server's error log file to successfully start a server. Each time a server is started with ***startserver***, information on the success (or failure) of the start is appended to its error log file.

SQL Server and Backup Server keep their error log files open until you stop the server process. To reduce the size of the error log file, stop the SQL Server or Backup Server process and delete the old messages.

startserver creates an output file for the server process in the *SYBASE_SYSTEM:[SYBASE.INSTALL]* directory. This file, called *START_SERVERNAME.OUT*, may contain Digital OpenVMS Alpha messages in addition to SQL Server error log messages. A new version of this file is created every time a server is restarted. If this file cannot be created at process creation, the server cannot start.

The Scripts Directory

Various SQL scripts are supplied in the *SYBASE_SYSTEM:[SYBASE.SCRIPTS]* directory of the SQL Server directory structure. **installmaster** and **installmodel** are used by the SQL Server installation program to load the *master* and *model* databases into a running SQL Server. These scripts contain the text of the

system procedures and grant necessary permissions for using SQL Server. Be sure to make backup copies of these scripts and store them in a safe place.

◆ ***WARNING!***

Do not edit or otherwise tamper with the Sybase system procedures. You should never need to edit installmaster and installmodel. If you encounter any problems (such as permissions), create a separate file rather than altering the original scripts. Sybase Technical Support can help diagnose and treat permissions problems.

Online Syntax Help: *sp_syntax*

The *SYBASE_SYSTEM:[SYBASE.SCRIPTS]* directory contains scripts for installing the "syntax help" database, *sybsyntax*. Users can retrieve this data with the **sp_syntax** system procedure.The *SCRIPTS* directory may contain the following **sp_syntax** scripts (depending on which Sybase products are included with your server):

Table 7-1: sp_syntax Installation Scripts

Script	Product
ins_syn_sql	Transact-SQL
ins_syn_cl	Open Client Client-Library
ins_syn_dblib	Open Client DB-Library
ins_syn_esql	Embedded SQL
ins_syn_os	Open Server

All SQL Server installations receive the **ins_syn_sql** script. This script includes syntax information for Transact-SQL, the System Procedures, and the Sybase Utilities. When you execute this script, you install the SQL portion of the *sybsyntax* database.

You can install any of these scripts, depending on the need for Sybase information on your server. The first script that you execute creates the *sybsyntax* database and needed tables and indexes. Any scripts that you execute after the first one add to the existing information in the database. If you execute a script that has been executed

previously, the previously installed rows of information are deleted from the table in the database and then reinstalled.

Default Device

The *sybsyntax* database requires 2MB on your default database device. By default, the *sybsyntax* installation scripts install the database on the device that is designated as the default.

If you have not used **sp_diskdefault** to change the status of master (installed as the default disk) and specify another default device, the scripts install *sybsyntax* on your master device. This configuration is not recommended because *sybsyntax* takes up valuable space that is best used for the system tables.

To avoid installing *sybsyntax* on the master device, do one of the following:

- Use **sp_diskdefault** to specify a default device other than master. See Volume 2 of the *SQL Server Reference Manual* for information on **sp_diskdefault**.
- Modify each *sybsyntax* installation script that you plan to execute to specify a different device, as explained in step 2 in the following section.

Installing *sybsyntax*

Perform the following steps for each *sybsyntax* installation script that you want to execute:

1. Make a copy of the original script. Be sure you can access this copy, in case you have problems with the edited script.
2. Use a text editor to edit the script, if necessary. You can add lines, delete lines, or comment out lines.

For example, the following text is the portion of the installation scripts that tests the size of the default devices and installs the database.

```
/* create the database, if it does not exist */
if not exists (select name from sysdatabases where name =
“sybsyntax”)
begin
/* create the sybsyntax table if it doesn’t exist */
/* is the space left on the default database devices > size of
model? */
```

```
if (select sum (high-low +1) from sysdevices where status & 1 =
1) - (select sum(size) from sysusages, sysdevices
      where vstart >= sysdevices.low
      and vstart <= sysdevices.high
      and sysdevices.status &1 = 1) >
(select sum(sysusages.size) from sysusages where dbid = 3)
        begin
create database sybsyntax
end
else
begin
print "There is not enough room on the default devices to create
the sybsyntax database."
return
end
end
```

If you want to install the database on a device other than master, you should comment out the portion of the script shown above, and replace it with the correct **create database** command. For example:

create database sybsyntax on *user_device*

where *user_device* is the name of the device where you want to install *sybsyntax*.

3. Execute the script with a command like the following:

```
$ ISQL /USER="SA" /PASSWORD="PASSWORD" -
_$ /SERVER="SERVER_NAME" -
_$ /INPUT=SYBASE_SYSTEM:[SYBASE.SCRIPTS]INS_SYN_SQL
```

where *sa* is the user ID of a System Administrator, *password* is the System Administrator's password and *servername* is the name of the SQL Server on which you wish to install the database.

4. To ensure that you have installed the *sybsyntax* database and that it is working correctly, you can use **isql** to log in to the server on which you installed the database, and execute **sp_syntax**. Use input like the following:

```
$ ISQL /USER="SA" /PASSWORD="PASSWORD" -
$_ /SERVER="SERVERNAME"
1> sp_syntax "select"
2> go
```

This input should return a list of commands containing the word or word fragment "select".

◆ ***WARNING!***

The ins_syn_cl and ins_syn_os scripts conflict. If you execute both of these scripts, errors relating to multiple entries to a unique index occur.

Sample Databases

The *SCRIPTS* directory also contains scripts for installing the U.S. English sample database, foreign language sample databases, and the *image* data associated with the U.S. English sample database.

◆ ***WARNING!***

Sybase does not recommend installing any of the sample databases on the master device.

pubs2 Database

The U.S. English sample database is called *pubs2*. The *System Administration Guide* details the contents of *pubs2* and provides suggestions for its use.

Perform the following steps to install *pubs2*:

1. Make a copy of the original **installpubs2** script. Be sure you can access the copy, in case you have problems with the edited script.
2. Use a text editor to edit the script, adding lines, deleting lines, or commenting out sections as necessary. Be sure to edit the script to specify a default device other than master, if necessary.
3. Execute the script with a command like the following:

```
$ ISQL /USER="SA" /PASSWORD="PASSWORD" -
_$ /SERVER="SERVERNAME" /INPUT= -
_$ $SYBASE_SYSTEM:[SYBASE.SCRIPTS]INSTALLPUBS2.
```

4. To install the *image* data associated with *pubs2*, repeat steps 1—3 with the script **installpix2** after you have installed *pubs2*.

➤ ***Note***

The image data requires a lot of space—there are six pictures, two each in the PICT, TIF, and Sun master file formats. Run **installpix2** only if you want to use or test the *image* datatype. Sybase does not supply any tools for displaying image data. You must use appropriate screen graphics tools to display the images once you have extracted them from the database.

interpubs Database

interpubs is a database similar to *pubs2* that contains French and German data. This data contains 8-bit characters and is for use at SQL Server installations using the ISO 8859-1 character set (iso_1). To display the French and German data correctly, you must set up your iso_1-compatible terminal to display 8-bit characters.

Perform the following steps to install *interpubs*:

1. Ensure that iso_1 is installed as the default character set or as an additional character set. See the *System Administration Guide* and Chapter 4, "Configuring Character Sets," for information on configuring character sets.
2. Make a copy of the original **installintpubs** script. Be sure you can access this copy, in case you have problems with the edited script.
3. Use a text editor to edit the script, adding lines, deleting lines, or commenting out sections as necessary. Be sure to edit the script to specify a default device other than master, if necessary.

```
$ ISQL /USER="SA" /PASSWORD="PASSWORD" /SERVER="SERVERNAME" -
_$ /DISPCHARSET=ISO_1 -
_$ /INPUT=SYBASE_SYSTEM:[SYBASE.SCRIPTS]INSTALLINTPUBS.
```

See *SQL Server Utility Programs for OpenVMS* for more information on the **/dispcharset** qualifier of **isql**.

➤ ***Note***

Since you may want to refresh or make new copies of the sample databases, back up the original and edited versions of the installation scripts and store them in a safe place.

What's Next?

Appendix D, "Sample SQL Server System Specifications" details requirements and limitations for SQL Server on your operating system.

8 Reconfiguring SQL Server and Backup Server

This chapter provides instructions for using an interactive **sybinit** session to reconfigure an existing SQL Server or Backup Server. Topics covered include:

See Appendix B, “sybinit Resource Files” for information on using **sybinit** resource files.

Reconfiguring SQL Server with *sybinit*

1. If you did not exit **sybinit** after a previous session, press Ctrl-x repeatedly until the Configure Server Products menu appears.

   If **sybinit** is not running, follow the instructions in “Verifying SYBASE_SYSTEM Definition” on page 4-4 and “Starting sybinit” on page 4-5. Select “Configure a Server Product” from the main menu. Select SQL Server from the Configure Server Products menu.

2. Select Configure An Existing SQL Server from the New or Existing SQL Server menu.

3. **sybinit** displays the names of existing SQL Servers as options. This list includes Backup Servers.

   ```
   CONFIGURE EXISTING SQL SERVER

   1. PUB
   2. MARKET
   3. SYBASE_110
   ```

sybinit does not allow you to select a server until you have reached the end of the list. If the entire list of servers does not fit on one screen, press Ctrl-f to scroll to the end of the list. Select the SQL Server to reconfigure.

4. **sybinit** displays the Enter SA Account Name and Password menu. Select SA Account and type the name of System Administrator account on your worksheet. Select SA Password and type the password for the SA Account. Press Ctrl-a to accept the account name and password.

   sybinit displays the SQL Server Configuration menu.

```
SQL SERVER CONFIGURATION

1. CONFIGURE SERVER'S INTERFACES FILE ENTRY          Unchanged
2. MASTER DEVICE CONFIGURATION                       Unchanged
3. SYBSYSTEMPROCS DATABASE CONFIGURATION             Unchanged
4. SET ERRORLOG LOCATION                             Unchanged
5. CONFIGURE DEFAULT BACKUP SERVER                   Unchanged
6. CONFIGURE LANGUAGES                               Unchanged
7. CONFIGURE CHARACTER SETS                          Unchanged
8. CONFIGURE SORT ORDER                              Unchanged
9. ACTIVATE AUDITING                                 Unchanged
10. Remap the query trees in all databases
11.SET QUOTA FILE LOCATION                           Unchanged
```

5. The SQL Server Configuration menu displays the Remap the Query Trees in All Databases option. This option applies to upgrades only. See Chapter 5, “Upgrading SQL Server” and Appendix A, “Recovery from Failure” for information on query tree remapping.
6. To complete other reconfiguration tasks, turn to Chapter 4, “Installing SQL Server.” Follow the instructions in the sections relating to the reconfiguration tasks you want to complete, as listed in the following table:

Table 8-1: SQL Server reconfiguration tasks

Reconfiguration Task	Related Section	Page
Rename default Backup Server	“Naming a Default Backup Server”	4-15
Reconfigure languages	“Configuring Languages”	4-15
Reconfigure character sets	“Configuring Character Sets”	4-17
Change sort order	“Configuring Sort Order”	4-18
Activate auditing	“Activating Auditing”	4-19

See the *System Administration Guide* for further details on configuring languages, character sets, and sort order.

Reconfiguring Backup Server with *sybinit*

1. If you did not exit **sybinit** after a previous session, press Ctrl-x repeatedly until the Configure Server Products menu appears.

   If **sybinit** is not running, follow the instructions in "Verifying SYBASE_SYSTEM Definition" on page 4-4 and "Starting sybinit" on page 4-5. Select Configure a Server Product from the main menu. Select Backup Server from the Configure Server Products menu.

2. Select Configure an Existing Backup Server from the New or Existing Backup Server menu.

3. **sybinit** displays the names of existing Backup Servers as options. This list includes SQL Servers.

   ```
   CONFIGURE EXISTING SQL SERVER

   1. SYB_BACKUP
   2. MARKET_BK
   3. PUB_BK
   ```

   sybinit does not allow you to select a server until you have reached the end of the list. If the entire list of servers does not fit on one screen, press Ctrl-f to scroll to the end of the list. Select the Backup Server to reconfigure.

   sybinit displays the Backup Server Configuration menu.

```
BACKUP SERVER CONFIGURATION

1. Backup Server errorlog:
SYBASE_SYSTEM:[SYBASE.INSTALL]BACKUP.LOG
2. Enter / Modify Backup Server interfaces file information
3. Backup Server quota file:
4. Backup Server language: us_english
5. Backup Server character set: iso_1To complete other
reconfiguration tasks now, follow the instructions in the
sections relating to the reconfiguration tasks you want to
complete, as listed in the following table:
```

Table 8-2: Backup Server reconfiguration tasks

Reconfiguration Task	Related Section	Page
Reconfigure language	"Configuring Backup Server Language"	6-7
Reconfigure character set	"Configuring Backup Server Character Set"	6-8

The *locales.dat* File

You may want to edit the *locales.dat* file later, after you have installed a new Language Module or when you reconfigure language or character set options for SQL Server or Backup Server.

Client applications use the *locales.dat* file to identify the language and the character set it is using. When a client application starts, it checks the operating system language setting (for example, french), then checks the *locales.dat* file for an entry for that locale. A locale entry for French would look like the following:

```
[vms]
      ; from "Digital Guide to Developing International Software"
      ; use environment variable SYS$LANGUAGE
      locale = FRENCH, french, iso_1
      locale = CANADIAN, french, iso_1
      locale = AUSTRIAN, german, iso_1
      locale = GERMAN, german, iso_1
      locale = SPANISH, spanish, iso_1
      locale = default, us_english, iso_1
```

When the client connects to SQL Server, the language and character set information is passed to SQL Server. SQL Server then:

- Uses the character set information (for example, iso_1) to identify the client's character set and verifies whether it can convert data to this character set.
- Uses the language (for example, french) and character set information (iso_1) to see if it has messages in the client's language and character set.

  For example, using the locale entry above, SQL Server first checks the *locales* directory for a French language subdirectory, then it verifies that the messages file exists in the *iso_1* character set subdirectory. See Figure 1-3 on page 1-11 for an illustration of the *locales* directory structure.

On SQL Server, the *locales.dat* file is used by applications such as **sybinit** or **isql**.

Format of *locales.dat* File Entries

Each entry in the *locales.dat* file links a platform-specific locale definition to a Sybase language and character set combination. Each entry has the following format:

```
locale = platform_locale, syb_language, syb_charset
```

When the locale being defined is the default for the site, the *platform_locale* is "default".

For example, the following entry defines the default locale as U.S. English with the iso_1 character set:

```
locale = default, us_english, iso_1
```

See your operating system documentation to determine valid *platform_locale* values for your platform.

See the *SYBASE_SYSTEM[SYBASE.LOCALES.LANGUAGE_NAME]* directories to determine valid values for *syb_language*.

See the *SYBASE_SYSTEM[SYBASE.LOCALES.LANGUAGE_NAME.CHARSET_NAME]* directories to determine valid values for *syb_charset*.

Sybase-Defined Locale Entries

If the Sybase locale entries defined in the locales file don't meet your needs, you can either modify them or add new locales entries. For example, the following lines show the locales entries from the *locales.dat* file:

```
; from "Digital Guide to Developing International Software"
; use environment variable SYS$LANGUAGE
; locale = default, us_english, iso_1
```

Editing the *locales.dat* file

Verify that the *LOCALES.DAT* file on SQL Server has a valid entry if:

- You install a new Language Module, and
- You want your messages on the SQL Server to appear in the new language.

To edit the *LOCALES.DAT* file:

1. Make a copy of the original, unedited *LOCALES.DAT* file in case you have problems with the edited version.
2. Make sure the file contains an entry for the language (*syb_language*) and character set (*syb_charset)* combination you want to use. The value for *platform_locale* must match the value required by your operating system.

   For example, if you want your SQL Server messages to appear in French, and your server is using the iso_1 character set, check the *LOCALES.DAT* entries for the OpenVMS operating system and look for the following entry:

```
[vms]
      ; from “Digital Guide to Developing International Software”
      ; use environment variable SYS$LANGUAGE
      locale = FRENCH, french, iso_1
      locale = CANADIAN, french, iso_1
      locale = AUSTRIAN, german, iso_1
      locale = GERMAN, german, iso_1
      locale = SPANISH, spanish, iso_1
      locale = default, us_english, iso_1
```

3. If the entry for your locale does not exist, add the required entry or modify an existing entry.
4. Save the file.

Summary of Reconfiguration Steps

Following is a summary of the steps for reconfiguring SQL Server, Backup Server, and OpenVMS parameters. The rest of this chapter explains these steps in detail.

1. Determine SQL Servers requirements.
2. Determine Backup Server requirements.
3. Estimate Digital OpenVMS Alpha SYSGEN parameters.
4. Estimate DECnet logical links.
5. Modify the SQL Server configuration.
6. Create a SQL Server quota specification file.
7. Modify the Backup Server configuration.
8. Create a Backup Server quota specification file.
9. Reconfigure SYSGEN and DECnet parameters.
10. Modify *SYS$MANAGER:SYSTARTUP_VMS.COM*.

11. Restart the operating system.
12. Restart SQL Server.
13. Restart Backup Server.

➤ ***Note***

Make copies of the worksheets that follow and use them to estimate and record the values for the system parameters as you reconfigure your system.

About Reconfiguring SQL Server

Use the SQL Server OpenVMS Resources Worksheet to see if your existing configuration meets current resource requirements. You should reconfigure SQL Server when:

- You need to change the number of user connections to database device or engines

  The **startserver** utility starts SQL Server as a detached process. **startserver** provides the values for the applicable Digital OpenVMS Alpha quotas and privileges necessary to support a default configuration. If you add any connections or devices to SQL Server, the SQL Server may use more resources than the default **startserver** provides.

- You need more memory for data and procedure buffers within SQL Server

Refer to the **sp_configure** in Volume 2 of the *SQL Server Reference Manual* for descriptions of all user-configurable parameters. Configuration and memory usage are also discussed in the *System Administration Guide*.

About Reconfiguring Backup Server

Refer to the *System Administration Guide* for information about using Backup Server, executing dump and load commands, and creating devices for dumping and loading. This information is necessary for planning the configuration for your dump and load operations.

Use the Backup Server OpenVMS Resources Worksheet to estimate the amounts of the following resources required for your dump and load operations:

- Server connections to Backup Server
- Network connections (DBPROCESSes) that Backup Server originates

You should reconfigure Backup Server if your dump or load operations require more than the following default values:

- 10 server connections (***/connections***)
- 25 network connections (**/network_connections**)

Use the **backupserver** executable with the appropriate command line options to modify your Backup Server configuration, as described later in this chapter.

SQL Server OpenVMS Resources Worksheet

➤ ***Note***

The following quotas are suggested values that Sybase believes will be sufficient. Customers knowledgeable in VMS tuning may choose different values, but should be aware that insufficient process quotas may seriously affect server performance.

Complete this worksheet as you follow the instructions in "Step 1: Determine SQL Server Requirements" on page 8-14.

	Parameter	How to Calculate	Total
I. User Connections Required			
1.	Current SQL Server user connections	**sp_configure** ***"user connections"***	_____
2.	Additional user connections needed		
	a. New **isql** users		_____
	b. New Data Workbench / APT-Library users	____ * 3 =	_____
	c. New **dbopen()** calls	____* no. of programs running =	_____
	d. New RPC sites		_____
	e. New network protocols	Add lines 2a-e	_____
	f. Total additional user connections needed		

	Parameter	How to Calculate	Total
3.	Total SQL Server user connections required	Add lines and 2f	_____
II. Devices Required			
4.	Current database/log devices a. Primary devices b. Mirrors	**select count(*) from master..sysdevices where cntrltype = 0**	_____ _____
5.	Additional database/log devices needed a. Primary devices b. Mirrors		_____ _____
6.	Total SQL Server database/log devices needed	Add lines 4a, 4b, 5a and 5b	_____
III. Engines			
7.	Total no. of SQL Server engines	Minimum value is 1; maximum is number of available CPUs	_____
IV. Sybase Memory Requirements			
8.	Current memory *run_value* for SQL Server (in Sybase 2K pages)	**sp_configure** "memory"	_____
9.	Additional user connection memory required for SQL Server (in Sybase 2K pages)	No. of new user connections (line 2f) * 25	_____
10.	Additional amt. of buffer/procedure cache required for SQL Server	No. of additional data buffers + No. of additional procedure buffers	_____
11.	Total memory required for SQL Server in Sybase 2K pages.	Add lines 8, 9, and 10	_____
V. OpenVMS Memory Requirements			
12.	OpenVMS memory pagelets required for SQL Server (in 512-byte OpenVMS pagelets)	Total Sybase memory (line 11) * 4 Note: 4 OpenVMS pagelets = 1 Sybase page	_____
VI. OpenVMS Process Quotas			
13.	BIOLM for SQL Server	7 * total SQL Server user connections (line 3)	_____
14.	DIOLM for SQL Server	(63 * total database and log devices (line 6)) + 16	_____
15.	TQELM for SQL Server	Total SQL Server user connections (line 3)	_____
16.	ASTLM for SQL Server	BIOLM + DIOLM + TQELM (Add lines 13, 14, and 15	_____

	Parameter	How to Calculate	Total
17.	FILLM for SQL Server	120 or [(Total SQL Server user connections (line 15)+ 6) + (total database/log devices (line 6) * engines (line 7))], whichever is the higher value	_____
18.	BYTLM for SQL Server	(ASTLM * 512) + 20000 + (FILLM * 96); Double the final total if you are using UCX; if you choose an insufficient value for BYTLM, your server may stall in a resource wait state (RWAST)	_____
19.	JTQUOTA for SQL Server	((Total user connections * 100) + 100) * Engines	_____
20.	ENQLM for SQL Server	= Total database and log devices + 8	_____
21.	PRCLM for SQL Server	(Engines * 2) - 1; or 0 (no limit)	_____
22.	PGFLQUOTA for SQL Server	OpenVMS pagelets (line 12) + (80 * Total user connections) + 2400	_____
23.	WSDEFAULT for SQL Server	OpenVMS pagelets] line 12	_____
24.	WSQUOTA for SQL Server	= PGFLQUOTA + 2000	_____
25.	WSEXTENT for SQL Server	= PGFLQUOTA + 2000	_____

Backup Server OpenVMS Resources Worksheet

Complete this worksheet as you follow the instructions in "Step 2: Determine Backup Server Requirements" on page 8-19.

	Parameter	How to Calculate	Total
VII. Connections Required			
26.	Backup Server connections	(2 * max. no. of simultaneous dumps/loads) + 1	_____
27.	Backup Server network connections		
	a. For local dumps/loads	2 * max. no. of SQL Servers dumping /loading simultaneously	_____
	b. For remote dumps/loads	1 * max. no. of stripes used for simultaneous remote dumps/loads	_____
	c. Total network connections	Line 27a + line 27b + 1	_____
VIII. Devices			
28.	Total Backup Server devices needed	Max. no. of devices simultaneously used for dumping or loading	_____

	Parameter	How to Calculate	Total
IX. Sybase Memory Requirements			
29.	Total memory required for Backup Server	(32K * max. no. of simultaneous dumps or loads) + 6656K	_____
X. OpenVMS Memory Requirements			
30	OpenVMS memory pagelets required for Backup Server (in 512-byte OpenVMS pagelets)	Total memory (line 29) / 512 Note: 2 OpenVMS pagelets = 1K	_____
XI. OpenVMS Process Quotas			
31.	BIOLM for Backup Server	4 * total Backup Server network connections (line 27c)	_____
32.	DIOLM for Backup Server	4 * max. no. of devices simultaneously used for dumping or loading (line 28)	_____
33.	ASTLM for Backup Server	BIOLM + DIOLM (Line 31 + line 32)	_____
34.	FILLM for Backup Server	(Backup Server network connections + 6) + max. no. of devices simultaneously used for dumping or loading	_____
35.	BYTLM for Backup Server	(32768 * max. no. of simultaneous dumps or loads) + (FILLM * 96); Double the final total if you are using UCX	_____
36.	ENQLM for Backup Server	= Total Backup Server devices (line 28)	_____
37.	PGFLQUOTA for Backup Server	OpenVMS pagelets (line 30)	_____
38.	WSDEFAULT for Backup Server	OpenVMS pagelets (line 30)	_____
39.	WSQUOTA for Backup Server	= PGFLQUOTA + 2000	_____
40.	WSEXTENT for Backup Server	= PGFLQUOTA + 2000	_____

SYSGEN and DECnet Parameters Worksheet

Complete this worksheet as you follow the instructions in "Step 3: Estimate OpenVMS SYSGEN Parameters" and "Step 4: Estimate DECnet Logical Links" on page 8-21.

	Parameter	How to Calculate	Total
XII. OpenVMS additional SYSGEN Parameters to current configuration			
41.	CHANNELCNT	Current value + (FILLM for SQL Server + FILLM for Backup Server) * 2	_____

	Parameter	How to Calculate	Total
42.	WSMAX	≥ (Larger of WSEXTENT for SQL Server or Backup Server) or current value	_____
43.	GBLPAGFIL	Current value + (Larger of PGFLQUOTA for SQL Server or Backup Server) / 16	_____
44.	GBLPAGES	Current value + (Larger of PGFLQUOTA for SQL Server or Backup Server)	_____
45.	VIRTUALPAGECNT	≥[(Larger of PGFLQUOTA for SQL Server or Backup Server) + 4400] or current value	_____
46.	NPAGEDYN	≥ [(Current SYSGEN value) + (BYTLM for SQL Server) + (BYTLM for Backup Server)] or current value	_____
XIII. DECnet Parameters			
47.	LOGICAL LINKS	(Non-local SQL Server node connections) + (2 * local SQL Server node connections) + (total Backup Server network connections) + (non-Sybase links)	_____

OpenVMS Resource Definitions

Following is a partial summary of the OpenVMS resources that SQL Server and Backup Server use.

ASTLM

Asynchronous Trap Limit. Servers use this resource primarily for I/O completion processing and network connection/interrupt servicing. SQL Server also uses the ASTLM for internal and user-initiated timing functions.

Because a server cannot start an I/O until an AST is available, I/O performance may be sluggish if you don't provide enough of this resource.

BIOLM and *DIOLM*

Buffered I/O limit and direct I/O limit. BIOLM and DIOLM determine the maximum number of buffered (DECnet I/O, *mailboxes*, *terminals*, and so on) and direct (TCP/IP, disk) I/O that a process can have outstanding at any time. If SQL Server or Backup Server exceeds these limits, I/O is deferred and attempted later. This message appears:

```
Insufficient DIOLM quota. Increase by number.
```

BYTLM

The maximum number of bytes of non-paged memory pool that a server can use.

ENQLM

The maximum number of locks queued at one time. A server uses an exclusive lock on each virtual device file it opens to prevent other servers from accessing the same virtual device file.

FILLM

The limit on the number of I/O channels that can be open at the same time. Servers use one open channel per connection and additional open channels for devices and other internal needs.

If you don't provide enough of this resource for SQL Server, it may run out of connections when a request is made by a client. SQL Server places a message in the error log file indicating that its open file limit quota has been exceeded, and the connection request is refused.

If you don't provide enough of this resource for Backup Server, it may run out of connections during the process of dumping or loading.

JTQUOTA

The byte quota for the job-wide logical name table. For each connection, SQL Server creates a temporary mailbox, whose logical name is placed in SQL Server's JOB level logical name table. A good rule of thumb is to provide 100 bytes of the JTQUOTA for each connection, plus an additional 100 bytes for logical names that SQL Server uses only once. For example, for a SQL Server that supports 36 connections, set up a JTQUOTA of 3700 bytes.

If you do not provide enough of this quota, you may see this error message:

PGFLQUOTA

Determines the number of 512-byte pagelets that may be allocated to each process in the paging file. If you do not provide enough of this quota, SQL Server or Backup Server fails to start, or the server may start but cannot create the subprocesses for all requested engines or **disk init**.

PRCLM

The limit of the number of subprocesses that a process can start. Each engine, except Engine 0, runs in a subprocess. Also, activities such as two-phase commit run in subprocesses.

TQELM

The maximum number of entries in the OpenVMS timer queue for the SQL Server process.

WSDEFAULT* and *WSQUOTA

The number of physical OpenVMS pagelets to be allocated to the SQL Server or Backup Server process.

➤ ***Note***

If SQL Server or Backup Server seems to be paging, increase the value of these two process quotas. Make sure that the value of WSEXTENT remains greater than or equal to the value of WSDEFAULT and WSQUOTA.

WSEXTENT

The maximum number of pagelets allocated to the SQL Server or Backup Server process, if available.

Step 1: Determine SQL Server Requirements

Use the “SQL Server OpenVMS Resources Worksheet” as you perform the tasks in this section.

Estimating SQL Server User Connections

A SQL Server connection is equivalent to a SQL Server task. That task may service a client network connection, a server-to-server RPC, or a network protocol listener. The default number of user connections is 25.

Current User Connections

1. Determine the currently configured number of user connections by executing this command in **isql**:

   ```
   sp_configure "user connections"
   ```

2. Record the number of current user connections on the worksheet.

Additional User Connections

You can make a reasonable estimate for the number of additional connections that SQL Server may need using the following guidelines:

- Each user running **isql** needs one connection.
- Data Workbench and APT-Library™ may each require up to three connections at a time for each user.
- In DB-Library™ applications, each call to **dbopen** creates one connection and each call to **dbclose** drops one connection.
- Each RPC site needs one connection for the duration of the network connection to that site.
- A SQL Server task is created for each network protocol with an active listener. SQL Server memory overhead for each user connection is 50K.
  - Record the number of additional **isql** connections on your worksheet.
  - Record the number of additional Data Workbench and APT-Library connections on your worksheet.
  - Record the number of additional **dbopen** calls on your worksheet.
  - Record the number of additional RPC sites on your worksheet.
  - Record the number of additional network protocols you need on your worksheet.

Total User Connections

1. Estimate the total number of additional user connections needed by totaling all of the connections estimated in the section "Additional User Connections" on page 8-15.
2. Record the total number of additional user connections needed on the worksheet.
3. Determine the total number of user connections needed by adding together the number of current connections and the total number of additional connections needed.
4. Record the total number of user connections needed on the worksheet.

Estimating SQL Server Devices

Devices are used either to contain database or transaction logs, or to store dumps of transaction logs or databases. SQL Server does not open dump devices, so you do not need to include them in your calculations.

Database and log devices may also have mirror devices on another disk.

Determine the number of current database and log devices and mirrors by executing this command in **isql**:

```
select * from master..sysdevices where cntrltype= 0
```

- Record the current number of primary database and log devices (not dump devices) on the worksheet.
- Record the current number of mirror devices on the worksheet.
- Record the number of additional primary database and log devices (not dump devices) needed on the worksheet.
- Record the number of these database or log devices that will be mirrored.
- Total the above four lines and record the value on the worksheet.

Specifying the Number of Engines Needed

If your machine supports symmetric multiprocessing (SMP) and is configured with more than one CPU, additional SQL Server engines

can be brought on-line. See the *System Administration Guide* for more information about SQL Server engines.

1. Determine the total number of current engines and additional engines needed for SQL Server. This value must be at least one. Do not enter a value that is greater than the number of available CPUs.
2. Record the total number of $_{SQL}$ Server engines on the worksheet.

Estimating SQL Server Memory Requirements

An OpenVMS memory pagelet is 512 bytes. A SQL Server memory page is 2048 bytes (four OpenVMS pagelets).

Current Memory Value

1. Determine the current memory value for your SQL Server by executing the following command in **isql**. The value shown in the *run_value* column is the current configuration in 2K SQL Server memory pages.

   ```
   sp_configure "memory"
   ```

2. Record the current memory run_value on the worksheet.

Memory for User Connections

1. Estimate the server memory required for additional user connections by multiplying the number of additional user connections (as recorded on the worksheet) by 25. Each connection uses approximately 25 2K server pages of memory.
2. Record the additional memory required on the worksheet.

Memory for Buffers and Procedure Cache

To estimate how many additional 2K SQL Server pages you need for buffers and procedure cache, first determine the percentage of available memory that your server allocates for procedure cache. Execute this command:

```
sp_configure "procedure cache"
```

and note the *run_value*. This is the percentage of available memory currently allocated for procedure cache. The rest is data buffer cache (*run_value* + buffer cache = 100). Calculate the number of pages you

need to add in order to increase the data buffer or procedure buffers by the appropriate amount. The formula for calculating the number of SQL Server pages to add for a given number of additional data buffers is as follows:

```
No. of additional data buffers needed / ((100 - run_value) / 100)
= no. of additional 2K memory pages needed
```

The formula for calculating the number of procedure buffers to add is as follows:

```
No. of additional procedure buffers / (run_value / 100)
= no. of additional 2K memory pages needed
```

For example, if you wish to add 100 data buffers, you need to perform the following steps.

1. Determine what percentage of available memory is procedure cache and what percentage is buffer cache. (For this example, procedure cache is 20 percent and buffer cache is 80 percent.)
2. Determine the total number of 2K server pages you need to add in order to add 100 data buffers, by dividing the number of buffers desired by the percentage of memory they represent (For this example, 100 / .80).
3. Add the appropriate number of memory pages. (For this example, 125 pages).
4. Record the number of additional buffers required on the worksheet.

You can also increase the procedure or buffer caches by changing the procedure cache configuration variable. In this case, the total pool of cache does not change, but when you increase the value of procedure cache, the procedure cache increases. When you decrease the value of procedure cache, the data buffer cache increases. See the *System Administration Guide* for more information.

Total Memory

1. Add the current memory to the memory needed for additional user connections needed and the additional cache memory needed.
2. Record the total memory required on the worksheet.

➤ ***Note***

Never exceed your system's physical memory.

Estimating SQL Server OpenVMS Memory Requirements

1. Estimate the OpenVMS memory requirement for SQL Server, multiply the number of Sybase memory pages required (as recorded on the worksheet) by 4.

   One SQL Server memory page (2K) equals 4 OpenVMS memory pagelets (512 bytes each).

2. Record the "OpenVMS memory required for SQL Server" on the worksheet.

Estimating SQL Server OpenVMS Resource Quotas

Complete the "OpenVMS Process Quotas" section of the SQL Server worksheet.

Step 2: Determine Backup Server Requirements

Use the "Backup Server OpenVMS Resources Worksheet" as you perform the tasks in this section.

Estimating Backup Server Connections

Estimate the maximum number of server connections and network connections Backup Server may need, according to the following guidelines.

Connections

Backup Server requirements for server connections are the same for local and remote dumps and loads, as follows:

- Maximum of two connections per simultaneous dump or load
- One connection for possible required volume changes
- Record the number of connections needed for Backup Server on the worksheet

Network Connections

Backup Server requirements for network connections vary for local and remote dumps and loads, as follows:

- For local dumps or loads:

  Two connections for each SQL Server simultaneously executing local dumps or loads

- For remote dumps or loads:

  One connection for each stripe used for simultaneous remote dumps or loads

In addition, Backup Server may require one network connection for possible required volume changes (if different SQL Servers are executing volume changes and dumps or loads).

- Record the number of network connections needed for local dumps or loads on the worksheet.
- Record the number of network connections needed for remote dumps or loads on the worksheet.
- Total the above, add one (for volume changes), and record the total number of network connections needed for Backup Server on the worksheet.

Estimating Backup Server Devices

1. Estimate the number of devices needed for Backup Server according to the maximum number of database devices and archive devices that you plan to use simultaneously during dumping or loading.

   For example, if you plan to dump five databases located on three separate database devices to five tapes (archive devices), then the number of devices needed is eight (three database devices plus five archive devices).

   See the *System Administration Guide* for more information about database dumping and loading.

2. Record the Backup Server devices needed on the worksheet.

Estimating Backup Server Memory Requirements

1. Estimate memory requirements for Backup Server based on the following guidelines:

   - Backup Server requires a maximum of 32K of memory for each simultaneous dump or load.

- The actual running program uses approximately 6656K (6.5 Mb) of virtual memory.

2. Record the total memory required for Backup Server on the worksheet.

Estimating Backup Server OpenVMS Memory Requirements

One kilobyte of memory for Backup Server equals two OpenVMS memory pagelets.

1. Estimate the OpenVMS memory requirement for Backup Server by dividing the number of kilobytes of Backup Server memory by 512.
2. Record the OpenVMS memory required for SQL Server on the worksheet.

Estimating Backup Server OpenVMS Resource Quotas

Complete the “OpenVMS Process Quotas” section of the Backup Server worksheet “Online Syntax Help: sp_syntax” on page 7-6

Step 3: Estimate OpenVMS SYSGEN Parameters

Complete the “SYSGEN Parameters” section of the “SYSGEN and DECnet Parameters Worksheet” on page 8-11.

Step 4: Estimate DECnet Logical Links

A logical link can be thought of as an end point of DECnet network communications. The total number of logical links available is a DECnet configuration parameter. The total number needed is the sum of the logical links used by Sybase software and those used by other processes on the system. For example, a **set host** command executed in a user process to connect to another node uses one logical link.

To calculate the required number of logical links:

- Determine the number of connections to the SQL Server from another node (non-local connections). Each takes one link.
- Determine the number of local connections to the SQL Server, that is, those from clients executing on the same node. Each takes

two links, since both communication endpoints originate on the same node.

- This calculation may vary slightly because of some exceptions. Each call to isql or **dbopen** takes one connection, and each Data Workbench or APT-Library application, in general, takes three connections.
- Add in the number of Backup Server network connections you calculated previously. Each takes one link.
- Add in the number of logical links used by non-Sybase applications. If you do not know this number, 32 is a good default.
- Record the sum of the above four logical link values on the worksheet.

Step 5: Modify SQL Server Configuration

➤ ***Note***

You must shut down and restart SQL Server for the following configuration changes to take effect. Be sure to follow steps 6 and 9, to create a SQL Server quota specification file and to reconfigure the DECnet and SYSGEN parameters, before restarting SQL Server. Where the **/password qualifier** appears, use the System Administrator password for the SQL Server you are reconfiguring.

User Connections

To configure SQL Server for more than the 25 default connections, execute the following:

```
$ ISQL /USER="SA" /PASSWORD="PASSWORD"
1> sp_configure "user connections", total user connections
2> go
1> reconfigure with override
2> go
```

where *sa* is the user name of an account with System Administrator privileges, *password* is the password for that account, and *total user connections* is the value you recorded on the worksheet (line 3) for the total number of SQL Server user connections needed.

Database and Log Devices

Change the number of database and log devices by executing the following commands:

```
$ ISQL /USER="SA" /PASSWORD="PASSWORD"
1> sp_configure "devices", total devices needed
1fs1fc
2> go
1> reconfigure with override
2> go
```

where *total devices needed* is the value you recorded on the worksheet (line 6) for the total SQL Server database and log devices required.

Engines

Change the number of engines by executing the following commands:

```
$ ISQL /USER="SA" /PASSWORD="PASSWORD"
1> sp_configure "max online engines",
2> total no. of engines
3> go
1> reconfigure with override
2> go
```

where *total no. of engines* is the total SQL Server engines required, as recorded on "SQL Server OpenVMS Resources Worksheet" on page 8-8, the worksheet line 7.

Memory

Change the default memory configuration by executing the following:

```
$ ISQL /USER="SA" /PASSWORD="PASSWORD"
1> sp_configure "memory", total memory required
2> go
1> reconfigure with override
2> go
```

where *total memory required* is the value you recorded (line 11) for the total SQL Server memory required, in Sybase 2K pages.

Step 6: Create a SQL Server Quota Specification File

Whether you reconfigure SQL Server for additional user connections, for additional memory, or for both at once, you must create a new quota specification file with the new OpenVMS resource quotas.

Use any editor to create the quota specification file in the directory *SYBASE_SYSTEM:[SYBASE.INSTALL]*.

Give the file the same name as the SQL Server using it. For example, if you have a SQL Server called SYBASE, name the file *QUOTA_SYBASE.DAT*. Use the values that you have recorded on your SQL Server reconfiguration worksheet to create the new file.

The quota file looks like this:

```
PROCESS_QUOTA_1=VALUE_1
PROCESS_QUOTA_2=VALUE_2
```

where PROCESS_QUOTA_*n* is a valid process quota.

Following is an example quota file that may be used by the **startserver** utility. These values are sufficient for a SQL Server that uses 40MB of memory, and has 100 connections, 25 devices and 2 engines. You can find a copy of this file in *SYBASE_SYSTEM:[SYBASE.INSTALL]PRODUCT 11.QUOTA*.

The contents of the file are:

```
astlm = 1452
biolm = 700
diolm = 652
bytlm = 778460
fillm = 156
enqlm = 156
jtquota = 20,200
prclm = 3
wsdefault = 81920
wsquota = 83920
wsextent = 83920
tqelm = 100
pgflquota = 92320
```

To use the *PRODUCT11.QUOTA* file, you must start SQL Server each time with:

```
/quota=SYBASE_SYSTEM:[SYBASE.INSTALL]SYBASE_Q.DAT
```

Step 7: Modify Backup Server Configuration

➤ ***Note***

You must shut down and restart Backup Server for the following configuration changes to take effect. Be sure to follow steps 8 and 9, to create a quota specification file and to reconfigure the DECnet and SYSGEN parameters, before restarting Backup Server.

Connections

To increase the number of server connections Backup Server can accept from the default of 10, execute the following:

```
$ BACKUPSERVER /CONNECTIONS=VALUE
```

where *value* is the maximum number of Backup Server connections, as recorded on "Backup Server OpenVMS Resources Worksheet" on page 8-10, on the worksheet line 26.

Network Connections

To increase the number of network connections Backup Server can originate from the default of 25, execute the following:

```
$ BACKUPSERVER /NETWORK_CONNECTIONS=VALUE
```

where *value* is the maximum number of Backup Server network connections required, as recorded on "Backup Server OpenVMS Resources Worksheet" on page 8-10, on the worksheet line 27c.

Step 8: Create a Backup Server Quota Specification File

If you reconfigure Backup Server for additional connections, you must create a new quota specification file with the new OpenVMS resource quotas.

Use any editor to create the quota specification file in the directory *SYBASE_SYSTEM:[SYBASE.INSTALL]*.

See the previous section, "Step 6: Create a SQL Server Quota Specification File" on page 8-24, for more information on creating a quota file.

Step 9: Reconfigure *DECnet* and *SYSGEN* Parameters

DECnet

The *LOGICAL LINKS* value you calculated on the worksheet refers to the maximum number of logical links for which DECnet is configured. To determine the current configuration, use the Network Control Program (NCP) by typing:

```
$ RUN SYS$SYSTEM:NCP
```

When the NCP prompt appears, type:

```
NCP> show executor characteristics
```

Information similar to the following appears:

```
Node Volatile Characteristics as of 30-Sept-96
08:57:59

Executor node = 1.34 (MUDGE)

Identification      = DECnet-VAX V5.3, OpenVMS V6.2
Management version  = V4.0.0
Incoming timer      = 45
Outgoing timer      = 60
Incoming Proxy      = Enabled
Outgoing Proxy      = Enabled
NSP version         = V4.1.0
Maximum links       = 32

    .

    .

    .
```

The value for Maximum Links should be greater than or equal to the value that you calculated for *LOGICAL LINKS*. If it is not, you must reconfigure DECnet. The following example shows how to reconfigure DECnet for 85 links:

```
NCP> define executor maximum links 85
NCP> set executor maximum links 85
NCP> exit
```

The first of these commands puts the new value for Maximum Links, “85”, into the permanent database for DECnet. You enter the value into DECnet’s permanent database so that, after an OpenVMS restart, the number of links remains set. The second command puts the new value into the volatile database for DECnet and takes effect immediately.

➤ ***Note***

Depending on the value of Maximum Links on your system, you may need to change other DECnet parameters. Consult your OpenVMS system manager and the *OpenVMS Guide to Networking* for more information on changing DECnet parameters and on resources used by DECnet.

SYSGEN

The following SYSGEN parameters may need to be changed for your installation after you reconfigure SQL Server or Backup Server.

- CHANNELCNT—the maximum number of channels available system-wide.
- GBLPAGFIL—the maximum number of global pages that can be created with page file backing store.
- GBLPAGES—the maximum number of global pages available on your system.
- NPAGEDYN—the maximum size of non-paged dynamic memory pool available system-wide.
- VIRTUALPAGECNT—the maximum number of virtual pages that can be allocated per process.
- WSMAX—the maximum working set size per process. Compare the values on your worksheet with the current configuration of your OpenVMS system to see if these must be changed to accommodate the reconfigured SQL Server or Backup Server. Failure to do so could result in unpredictable SQL Server or Backup Server behavior.

➤ ***Note***

The values calculated on the worksheet are minimum values for running SQL Server and Backup Server. These values must also reflect other activity on the system. Consult your OpenVMS system manager for help in reconfiguring and rebooting OpenVMS.

➤ ***Note***

You should always use the **AUTOGEN** procedure described in the *Guide to Setting Up an OpenVMS System* in the OpenVMS documentation set. Failure to do so could result in unpredictable problems.

Paging File

The OpenVMS system's total paging space must be large enough to accommodate the larger of SQL Server's or Backup Server's PGFLQUOTA. Consult your OpenVMS system manager about this.

Step 10: Modify *SYS$MANAGER:SYSTARTUP_VMS.COM*

If you want to integrate SQL Server or Backup Server startup with OpenVMS System startup, you must modify the file *SYSTARTUP_VMS.COM* so that the command lines for starting SQL Server and Backup Server reference the new quota specification files. Insert these command lines **after** the commands that start your network software (DECnet and / or TCP / IP).

The new command file should look like this:

```
$ ! Define the Sybase  system level logicals.
$ ! Replace the reference to SYBASE_LOGIN_DEVICE on the next line
$ ! with the actual login device for the Sybase  user account.
$ ! This line should appear after the @STARTNET command so that
$ ! DECnet will have been started.
$ @SYBASE_LOGIN_DEVICE:[SYBASE.INSTALL]SYLOGICALS.COM
$ !
$ STARTSERVER :== $SYBASE_SYSTEM:[SYBASE.INSTALL]STARTSERVER.EXE
$ !
$ ! If the startserver command is used with no other arguments,
$ ! the SQL Server will be started by using the file that
$ ! is located in SYBASE_SYSTEM:[SYBASE.INSTALL]RUN_SYBASE.COM.
$ !
$ STARTSERVER
/QUOTA=SYBASE_SYSTEM:[SYBASE.INSTALL]SYBASE_QUOTA.DAT
$ STARTSERVER
/QUOTA=SYBASE_SYSTEM:[SYBASE.INSTALL]BSRV1_QUOTA.DAT
/BACKUPSERVER=BSRV1
```

This command file starts SQL Server with the default name SYBASE, using the file called *QUOTA_SYBASE.DAT* to determine the correct resource quotas and starts Backup Server with the name *BSRV1*,

using the file called *BSRV1_QUOTA.DAT* to determine the correct resource quotas. Use the **/server** qualifier with the **startserver** command if your SQL Server has a name other than the default.

See the *SQL Server Utility Programs for OpenVMS* for more information about **startserver.**

Step 11: Restart Operating System

If you modified the *SYS$MANAGER:SYSTARTUP_VMS.COM* file to integrate SQL Server or Backup Server restart with operating system restart, restart your operating system now.

Step 12: Restart SQL Server

If you did not integrate SQL Server restart with operating system restart, use the following information to restart SQL Server.

The **startserver** utility uses the quota specification file you created to provide SQL Server with the new resource quotas. Reference the quota specification file on the **startserver** command line like this:

```
$ STARTSERVER /QUOTA=QUOTA_SPEC_FILENAME /SERVER=SERVER_NAME
```

Unless you specify a different directory, **startserver** looks for the quota file in the current default directory. **startserver** starts SQL Server as an OpenVMS detached process and passes the new resource quotas to this process. If you specify **startserver** with the **/wait** or **/companion** options, **startserver** waits for SQL Server to start and notifies you that SQL Server has started successfully. If an error occurs during SQL Server startup, **startserver** reports the error in a file called *START_servername.OUT* located in *SYBASE_SYSTEM:[SYBASE.INSTALL]* or in the SQL Server error log file.

➤ ***Note***

The most common problem when starting SQL Server with the new quota specification is that the system appears to hang.

Usually this problem indicates that you did not set PGFLQUOTA high enough to accommodate WSQUOTA, or there is not enough space in the OpenVMS system paging file to accommodate WSQUOTA. If you suspect that SQL Server is hanging, use the

stopserver command with the **/abort** qualifier to stop SQL Server and check PGFLQUOTA and WSQUOTA.

➤ ***Note***

Once you have reconfigured SQL Server, use the quota specification file whenever you restart SQL Server. If you do not, SQL Server may hang, or you may experience other system problems.

Step 13: Restart Backup Server

If you did not integrate Backup Server restart with operating system restart, use the following command to restart Backup Server.

```
$ STARTSERVER /QUOTA=QUOTA_FILE_NAME /BACKUPSERVER=SERVER_NAME
```

Installing Language Modules Only

If you are installing Language Modules only, you can use **sybinit** to reconfigure languages, character sets, or sort order. Perform the following steps to install Language Modules:

1. Prepare your operating system according to the instructions in Chapter 2, “Preparing for New Installation or Upgrades.” (These preparations are not necessary if you completed them when installing SQL Server.)
2. Load Language Module software from media according to the instructions in Chapter 3, “Loading Software from Media.” You can load this software into the same directory where you loaded SQL Server.
3. Edit the *locales.dat* file to include the proper definitions for your system.
4. Follow the instructions in this chapter to start a **sybinit** session and reconfigure languages, character sets, or sort orders for SQL Server or Backup Server.

See Chapter 1, “Introduction” for information on localization files and available languages, character sets, and sort orders.

What's Next?

If you have completed your installation, configuration and upgrade tasks, read the *System Administration Guide* and *System Administration Guide Supplement* for information on getting started with SQL Server.

9 Administrative Tasks and Performance and Tuning

The administration of SQL Server databases includes routine tasks and performance and tuning considerations. Topics include:

The Interfaces File

Sybase SQL Server can communicate with a variety of other software products on the same network. This communication occurs through the interfaces file, usually called *INTERFACES.*, which resides in the *SYBASE_SYSTEM:[SYBASE]* directory. The interfaces file is like an address book. This file contains a name and address entry for each SQL Server or Backup Server to which a client might connect.

➤ ***Note***

This chapter discusses interfaces file entries for and client communication with SQL Servers. Backup Servers also have interfaces file entries and can communicate with clients. The material in this chapter also applies to Backup Servers, although it does not always refer to them.

You can use the **sybinit** utility to create, verify or modify interfaces file entries for SQL Server or Backup Server. This chapter provides instructions for using **sybinit** to create an interfaces file entry.

You can also edit the interfaces file manually, but do so only if you are an experienced user, and always make a copy of the file first. See "Creating a Master Interfaces File Manually" on page 9-14 for some other precautions you should take if you manually edit the interfaces file.

Topics covered in this chapter include:

- Communication between Sybase clients and SQL Server
- Format of interfaces file entries
- Functions of different parts of an interfaces file entry
- Using **sybinit** to create interfaces file entries
- Altering the interfaces file to accommodate multiple SQL Server installations

Interfaces File Format

When you successfully configure a new SQL Server, **sybinit** creates an entry for it in the interfaces file for your system. Examine this entry for an example of the proper syntax of the interfaces file entry elements described in the following sections.

Each SQL Server that you plan to use should have only one entry, although there may be multiple lines in the entry. Each interfaces file entry has the following format:

```
# put comments here<return>
SERVERNAME<tab>RETRY_ATTEMPTS<tab>DELAY_INTERVAL<return>
<tab>CONNECTION_TYPE  PROTOCOL NETWORK NODE OBJECT_NUMBER<return>
<blank lines between entries for different SQL Servers>
```

➤ ***Note***

You **must** place a <tab> or a space before each entry line following the "SERVERNAME" line. The entry does not work without a space or a <tab> at the beginning. Use a single space between each element on an entry line.

Following is a description of each element in an interfaces file entry:

SERVERNAME

SERVERNAME is the name of the SQL Server or Backup Server. Server names must be no more than 11 characters long. The initial character of a server name must be a letter (ASCII a-z, A-Z). The characters that follow must be letters, numbers, or the underscore ("_").

Because server names are case-sensitive in Digital OpenVMS Alpha, your server name should be in all upper-case letters.

retry_attempts and *delay_interval*

The *retry_attempts* and *delay_interval* values determine how many times and how often a client tries to connect to a server after an initial failure to connect.

Clients try to connect to a server using each query line in the interfaces file in the order in which the entries appear. The client connects to the server using the first working query line it finds. If the client can't find any working query lines, it checks the *retry_attempts* and *delay_interval* values in the interfaces file. The *retry_attempts* value specifies how many times the client loops through the file looking for a connection after an initial failure to connect. The *delay_interval* specifies how many seconds it waits between retries.

When you configure a server, **sybinit** prompts you to specify the delay interval and number of retry attempts for that server. You can use **sybinit** to change these values at a later date. You can also manually edit your interfaces file using an operating system editor. If you manually edit the interfaces file, make sure that you insert tabs between these integer values.

connection_type

The *connection_type* is the entry's connection type. A tab precedes the *connection_type* on each line. There are three possible values for *connection_type*: "query", "master", and "debug":

- A line beginning with the word "query" represents a connection from a client, such as Data Workbench, that needs to log in to a server. The "query" entry in an interfaces file corresponds to the DSQUERY logical name.
- A line beginning with the word "master" represents the connection that the server itself uses. This entry corresponds to the DSLISTEN logical name, defined in the process running the server, and it tells the server where to listen for logins from clients.
- A line beginning with the word "debug" represents a connection for use by Sybase Technical Support.

If you install only Sybase clients (Data Workbench®, for example) on a machine, the interfaces file for that installation contains entries with the "query" line only. If no server uses that interfaces file entry, the "master" and "debug" lines are unnecessary.

➤ ***Note***

All elements of the "query" entry in the interfaces file on the node running the client software must match all elements of the "master" entry in the interfaces file on the node running SQL Server.

protocol

protocol is the name of the network protocol. On Digital OpenVMS Alpha, SQL Server and Backup Server support the DECnet and TCP / IP protocols simultaneously. The two possible values for *protocol* are "decnet" and "top". This simultaneous multiprotocol support is managed by additional entries in the interfaces file that route connection requests from a client to a particular server over the appropriate network.

When a server starts, it establishes a connection to the network using the protocol(s) specified in the interfaces file. If the network is later shut down, the server is no longer be accessible even if the network is restarted. The network object is no longer allocated to the server. The server must be restarted to reestablish the network connection.

network

network is the name of the network. Although this column of the entry is not currently used by SQL Server, **sybinit** enters a network name. For OpenVMS, the name **sybinit** enters is "dec-ether".

node

node is the network name or address of the machine on which the server is running. For DECnet node names, the first 6 bytes of the machine name must be unique on the network. The DECnet node name in the interfaces file has a maximum of 6 bytes. A TCP / IP node name is its Internet address.

You can use network utilities to find a network name or Internet address. You may need system privileges. The following paragraphs detail how to determine the network name or address of a machine.

Finding Node Names on DECnet

To find the network address and node name on DECnet, type:

```
$ SHOW NETWORK
```

In the first line of the output from this command, OpenVMS displays the network address and node name. For example:

```
OpenVMS Network status for local node 1.53 WHITNEY on 8-JUN-1992
11:31:13.72
```

object or *port*

sybinit asks for an object name or number, or a port number, for a server. The DECnet and TCP/IP protocols specify these values differently:

- DECnet requires a number between 128 and 253.

  Object names are not recommended. Sybase supports network object names for compatibility with previous releases. If you use an object name when running SQL Server on DECnet, Companion Server may have problems starting. See Chapter 3, "Loading Software from Media" for more information about running Companion Servers.

- TCP/IP requires a number between 5000 and 65535.

An object or port number must be unique to the VMScluster on which you are running SQL Server.

You can verify in DECnet that the number is unique with the following commands:

```
$ RUN SYS$SYSTEM:NCP
NCP> show known objects
NCP> exit
```

➤ ***Note***

These commands display only the current DECnet network objects on the current node. If a product using the same network number starts before SQL Server, you cannot start SQL Server.

A list of the network object numbers currently in use is displayed in the "Number" column. Make sure the number you plan to use does not appear.

illustrates the differences in node names and numbers (object numbers and port numbers) on networks using the different protocols.

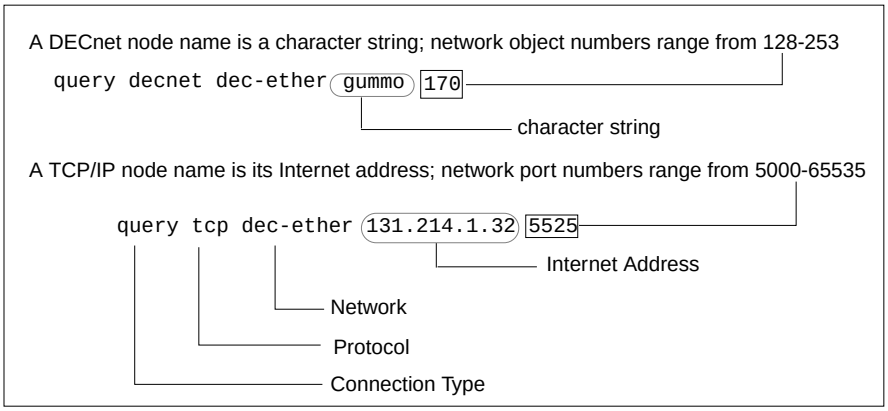


Figure 9-1: Node names and numbers on supported networks

Name Alias

To specify a protocol for your Sybase application to use to connect to your server, create a server name alias for that protocol in the interfaces file. Otherwise, the application connects to the server via the first query entry in the interfaces file. You can specify a name alias in an interfaces file entry during a **sybinit** configuration session, or you can edit the file manually to add an alias.

The server name alias entry is formatted like a regular server name entry, but includes only a query line (no master line is needed). For example:

```
#
TEST
    query tcp dec-ether 131.214.1.32  5525
    master tcp dec-ether  131.214.1.32  5525
    query decnet dec-ether gummo 150
    master decnet dec-ether gummo 150

#
TEST_DEC
    query decnet dec-ether gummo 150
#
TEST_TCP
    query tcp dec-ether 131.214.1.32 5525
```

In this example, if you want a client to connect over either the TCP/IP protocol or the DECnet protocol (with TCP/IP as the first choice, because it appears first in the interfaces file), define your DSQUERY logical name as TEST, or specify TEST on the command line with the **/server** qualifier. If you want a client to connect using only TCP/IP, or DECnet, set your DSQUERY logical name to TEST_TCP or TEST_DEC, respectively.

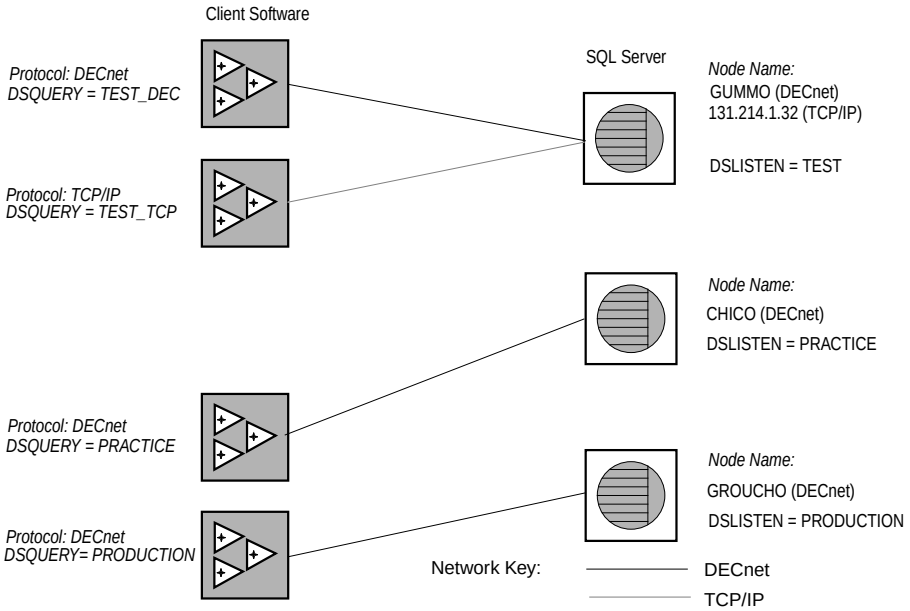
Interfaces File Example

In Figure , the clients using DECnet and TCP/IP to connect with a SQL Server named TEST use server name aliases to ensure that they connect using a specific network protocol. The SQL Servers are:

Table 9-1: SQL Server, networks, and node name for interfaces file example

SQL Server Name	DECnet	TCP/IP	Node
TEST	Yes	Yes	gummo
PRACTICE	Yes	No	chico
PRODUCTION	Yes	No	groucho

Figure illustrates an environment with one SQL Server that supports two networks, and two SQL Servers that support only DECnet:




```
#
TEST    2     10
        query decnet dec-ether gummo 170
        query tcp dec-ether    131.214.1.32 5525
        master decnet dec-ether gummo 170
        master tcp dec-ether   131.214.1.32 5525

        debug decnet dec-ether gummo 172
#
TEST_DEC 2      10
        query decnet dec-ether gummo 170
#
TEST_TCP 2 10
        query tcp dec-ether 131.214.1.32 5525
#
PRACTICE   2    10
        query decnet dec-ether chico 160
        master decnet dec-ether chico 160

#
PRODUCTION 2 10
        query decnet dec-ether groucho 200
        master decnet dec-ether groucho 200
```

Figure 9-2: An interfaces file for three SQL Servers

Media Supported for Backups on OpenVMS

Sybase supports the following types of media for dumping from SQL Servers:

- 4-millimeter/SCSI (primary)
- 8-millimeter/SCSI
- TZ85
- Disk

Before performing a backup, use **sp_helpdevice** to make sure a dump device exists in SQL Server. If no device exists, or if you do not want to use an existing device, add a device with **sp_addumpdevice.** Refer to the *SQL Server Reference Manual* for more information about **sp_addumpdevice**.

For complete details about backing up your SQL Server databases, see the *System Administration Guide*.

Defining the DSQUERY Logical Name

Sybase clients compare the value of the DSQUERY logical name to a SQL Server name listed in the interfaces file.

The DSQUERY logical name must specify the name of a SQL Server running on the network. There must be a corresponding DSLISTEN logical name defined in the process running SQL Server. For example, the following DCL code specifies a connection to SQL Server NEWSERVER:

```
$ ! DSQUERY defined to allow connection to the SQL Server
$ ! named 'NEWSERVER'
$ DEFINE DSQUERY NEWSERVER
```

Alternatively, you can define DSQUERY for all users on the node by adding the following line to *SYS$MANAGER:SYSTARTUP_VMS.COM:*

```
$ DEFINE/SYSTEM DSQUERY NEWSERVER
```

Using *sybinit* to Create or Edit Interfaces File Entries

You can use the **sybinit** utility to create or edit interfaces file entries. This section provides instructions for using **sybinit** to create a new interfaces file entry.

If you have SQL Server installations in different directories (for example, you have a SQL Server release 11.0.x installation and an installation from an earlier release of SQL Server that still has SQL Servers in use), see "Creating One Interfaces File for Multiple Installations" on page 9-14.

Starting *sybinit*

1. Make sure the **sybinit** symbol is defined:

   $ SHOW SYMBOL SYBINIT

2. If the symbol is not defined, execute the following:

   $ @SYBASE_SYSTEM[SYBASE.INSTALL]SYBASE_MANAGER.COM

3. Start **sybinit** by typing this at the DCL prompt:

   $ SYBINIT

4. When the Sybinit main menu appears, check to be sure the Release Directory entry is correct.
5. At the main menu, select Edit / View Interfaces File. **sybinit** displays the Interfaces File Top Screen.
6. Select Add a New Entry. **sybinit** displays the "Create New Interfaces File Entry" menu.
7. Select Server Name. At the prompt, enter the name of the server for which you want to create an entry.
8. If you are creating an interfaces file entry for a Backup Server, enter the name of a Backup Server.
9. When **sybinit** displays the correct server name at the Create New Interfaces File Entry menu, press Ctrl-a to accept it. **sybinit** displays the Server Interfaces File Entry Screen:

```
SERVER INTERFACES FILE ENTRY SCREEN

     Server name: SYBASE

1. Retry Count: 0
2. Retry Delay: 0

3. Add a new listener service


Ctrl-a Accept and Continue, Ctrl-x Exit Screen, ? Help.

Enter the number of your choice and press return:
```

Entering Connection Information (Optional)

1. Select Retry Count and enter a retry count value.
2. Select Retry Delay and enter a retry delay value. **sybinit** returns to the Server Interfaces File Entry Screen.

   The Server Interfaces File Entry Screen now displays the retry count and retry delay values that you entered (or 0 if you did not enter any values). You can change either of the displayed values by reselecting the appropriate menu item and entering a new value at the prompt.

Selecting Network Protocol

Select Add a New Listener Service. **sybinit** moves to the Select Network Protocol menu:

```
SELECT NETWORK PROTOCOL

1. TCP
2. DECnet
```

Follow the instructions in whichever of the following sections apply to the network protocol(s) you select.

Adding a TCP Entry

To add a TCP entry, perform the following steps:

1. Choose TCP from the Select Network Protocol menu. **sybinit** moves to the Edit TCP Service menu:

   ```
   EDIT TCP SERVICE

   1. Hostname/Address:
   2. Port:
   3. Name Alias:
   4. Server Type: master

   5. Delete this service from the interfaces entry
   ```

2. If the Hostname / Address item is correct, skip to the next item. To change the default SQL Server host name, select the Hostname / Address item and enter the IP address of the machine where your SQL Server installation is located.
3. Select Port and enter a 4- or 5-digit port number.
4. (Optional) Select Name Alias and enter a SQL Server alias.

5. Server Type refers to whether the entry is for a Master or Companion Server. The default is "master."

   If you are creating an entry for a Companion Server, select Server Type. At the prompt that asks if the SQL Server is a Companion Server, type "y".

➤ ***Note***

If you use Companion Servers, you must specify listeners for the Companion Servers.

6. Skip to "Accepting the New Entry" on page 9-13.

Adding a DECnet Entry

To add a DECnet entry, perform the following steps:

1. Select DECnet from the Select Network Protocol menu. The Edit DECnet Service menu appears:

   ```
   EDIT DECNET SERVICE

   1. Nodename:
   2. Object:
   3. Name Alias:
   4. Server Type: master

   5. Delete this service from the interfaces entry
   ```

2. If Nodename is correct, skip to the next item.
3. To change the default SQL Server node name, select Nodename and enter a node name.
4. Select Object and enter the object number recorded on your worksheet. Object names are not recommended.
5. (Optional) Select Name Alias and enter a SQL Server alias.

   Server Type refers to whether the entry is for a Master or Companion Server. The default is "master."
6. If you are creating an entry for a Companion Server, select Server type.
7. At the prompt that asks if the SQL Server is a Companion Server, type "y".

Deleting an Interfaces File Entry

1. Select the interfaces file entry you want to delete and press Return. **sybinit** asks you to confirm this selection:

   ```
   Remove this entry from the interfaces file?n
   ```

2. Enter "y" to delete this entry. **sybinit** returns to the Interfaces File Top Screen.

Modifying or Viewing an Interfaces File entry

1. **sybinit** displays the Server Interfaces File Entry Screen:

```
SERVER INTERFACES FILE ENTRY SCREEN

   Server name:  IGNATZ
1.  Retry Count:  5
2.  Retry Delay:  5
3.  Add a new listener service

Modify or delete a service
Listener services available:

Protocol       Address     Port        Name Alias
4.  tcp        IGNATZ      5749        KrazyKat


Ctrl-a Accept and Continue, Ctrl-x Exit Screen, ? Help.
Enter the number of your choice and press return:
```

   If you selected View an Interfaces File Entry, you will not be able to select the items on this screen.

   If you select Modify an Interfaces File Entry, follow the instructions beginning with "Entering Connection Information (Optional)" on page 9-11.

Accepting the New Entry

1. Verify that all entries in the Edit [Protocol] Service (for whichever protocol you selected) are correct. If necessary, repeat the preceding instructions to correct an entry.
2. Press Ctrl-a to accept the information. **sybinit** asks if the information is correct.
3. Type "y" to proceed with the configuration. **sybinit** returns to the "Server Interfaces File Entry Screen."

4. Press Ctrl-a to record the new information in the interfaces file. **sybinit** prompts you to write the changes to the file.
5. Type "y" to write the information to the interfaces file. **sybinit** returns to the SQL Server Configuration menu and marks the Configure Server Interfaces File Entry item "Changed".

Creating One Interfaces File for Multiple Installations

Distributing copies of one interfaces file with entries for all SQL Servers is the easiest way to ensure that all Sybase products on the network can communicate with each other. To maintain consistency in the interfaces files on a network, make all changes to one version of the file; then copy the updated file to the rest of the machines on the network.

For example, the interfaces file in "" on page 9-8 was created by copying the "groucho" version and "chico" version to "gummo", and then using a text editor to merge the three files. The file containing all three definitions was then copied to "groucho" and "chico."

Using *sybinit* to Create a Master Interfaces File

To maintain one consistent interfaces file (a "master" file) in an environment with multiple SQL Server installations, follow these steps:

1. Select the interfaces file that contains the most complete, up-to-date information.
2. Begin a **sybinit** session in your latest Sybase installation to edit this interfaces file. If the file is not in the same installation from which you are running **sybinit**, change the release directory item in the main menu to the file's location.
3. Follow the instructions in the preceding sections to create entries for any SQL Servers or Backup Servers from other installations that are not listed in the interfaces file.
4. Copy this interfaces file to all appropriate Sybase directory trees.

Creating a Master Interfaces File Manually

If you are already familiar with Sybase software, you may find it is faster to create one interfaces file by using a text editor. Be extremely

careful when editing this file. Do so only if you are very familiar with the correct format.

1. Before you create a master interfaces file through manual editing, be sure that the values you use for the different elements in the interfaces file on each machine are correct for that machine. You can check the new values you enter by comparing them with values for existing entries.
2. Create a complete interfaces file by copying the contents of all the interfaces files from all machines with SQL Server installations into one file.
3. Use a text editor to edit this merged file so that there is one entry for each SQL Server and each Backup Server. Do not attempt to use an application program to edit the interfaces file; you must use a text editor.
4. Copy this interfaces file to all appropriate Sybase directory trees.

Performance and Tuning

This chapter discusses how to improve SQL Server performance on OpenVMS systems and gives a brief overview of the operating system commands to use for monitoring resource usage. You can monitor performance and SQL Server usage of operating system resources—disk, memory, and I/O—to see if you need to make any changes to your system. Monitoring system resource usage is particularly important for SQL Servers running in multiprocessor environments.

Multiple Disk Drives and SQL Server Performance

Performance in an I/O-bound application is determined by the number of disk drives and disk controllers on a system, not by the amount of space available. A single disk drive may not be able to deliver the number of I/Os per second that you need for your SQL Server application. To achieve your performance objectives for an application, you must have enough disk drives. Your disk drive requirements may not be directly related to the size of your database.

Sybase recommends that you distribute data from heavily used databases across multiple disks. To distribute data effectively, you need to monitor disk usage. If one or more disks are consistently very busy, consider distributing the database objects on those disks to

other devices. This distribution spreads out the work among disks and allows for greater data throughput.

For most databases, on-disk space utilization is best monitored using SQL Server stored procedures, such as **sp_helpdevice** and **sp_helpdb**. See the *SQL Server Reference Manual* for information on these procedures.

See "Monitoring Disk Usage" on page 9-16 for information on monitoring disk usage.

Monitoring SQL Server Use of Operating System Resources

The *System Administration Guide* discusses maintenance of the optimal number of SQL Server engines for your workload and system configuration. To determine the optimal number, you need to monitor system and SQL Server CPU usage.

OpenVMS supplies many tools to help monitor performance, some of which are briefly discussed here. For details, consult your OpenVMS system manager, OpenVMS manuals, and on-line documentation.

- The **MONITOR** utility can be used to monitor disk activity and CPU usage.
- The **SHOW PROCESS** command can be used to monitor CPU usage.

Monitoring Disk Usage

The **MONITOR** utility can be used to monitor disk activity. The following command displays the number of I/O request packets waiting and currently being serviced by the disks mounted on your node:

```
$ MONITOR DISK /ITEM=QUEUE_LENGTH
```

The output looks like the following:

```
                    OPEN VMS Monitor Utility
                      DISK I/O STATISTICS
                         on node STAR
                       4-FEB-1996 13:59:51

I/O Request   Queue     Length            CUR     AVE     MIN     MAX

$40$DUA0:     (HSC000) STAR_SYS          0.33    0.66    0.00    0.66
$40$DUA1:     (HSC000) DEV7              0.00    0.00    0.00    0.33
$40$DUA5:     (HSC000) DEV5              0.00    0.00    0.00    0.66
$40$DUA6:     (HSC000) DEV6              2.33    1.67    0.00    4.00
$10$DUA0:     (SPOT)   FIFI_SYS    (R)   0.00    0.00    0.00    0.33
$10$DUA1:     (SPOT)   FIFI_SYS2   (R)   0.00    0.00    0.00    0.00
ERIS$DKB0:             ERIS_SYS    (R)   0.00    0.00    0.00    0.00
ROVER$DUA0:            ROVER_1     (R)   0.00    0.00    0.00    0.00
ROVER$DUA1:            ROVER_2     (R)   0.00    0.00    0.00    0.00
```

Sampling is done every second. Use the **/interval=***n* qualifier to change the sampling interval. Disks that consistently show outstanding I/O requests are busier than disks that do not. Please consult your system manager when you interpret these numbers. Some judgment is necessary to decide what average number is large enough to require action. See the *System Administration Guide* for more information.

Monitoring CPU Usage

The *System Administration Guide* discusses maintenance of the correct number of SQL Server engines in a multiprocessor environment. You need to monitor CPU usage to determine the correct number of engines. On OpenVMS, you can watch SQL Server CPU usage as it accumulates, monitor its current percentage of total CPU usage, or display it as a static amount accumulated since the last server boot.

Use the following command to watch CPU usage accumulate:

```
$ SHOW PROCESS /CONTINUOUS ENGINE_PROCESS_NAME
```

If you use SYBASE_SQL as an example, the output looks like the following:

```
 Process SYBASE_SQL          11:38:32

State               CUR          Working set            5849
Current PC          7FFEDF8A     CPU time     000:00:00:13.42
Current PSL         03C00000     Direct I/O               390
Current user SP     00217414     Buffered I/O             130
PID                 40801CC6     Page faults             3025
UIC                 [230,17]     Event flags         60000000
                                                     80000000

STAR$DUA0:[SYBASE.BIN]DATASERVER.EXE;1
```

This output is refreshed whenever one of the displayed values changes. The "CPU time" value indicates CPU usage. You need to run the CPU usage command for each engine. For example, if your server name is Sybase and you are running four engines, you need to look at CPU usage for SYBASE_SQL, SYBASE1, SYBASE2, and SYBASE3. Engine 0 is named *server_name*_SQL. Additional engines are named *server_name* appended with an *engine_number*.

Use the MONITOR facility to obtain the percentage of total CPU used by SQL Server:

```
$ MONITOR PROCESS/TOPCPU
```

The command produces a histogram that looks like the one below. The system refreshes this histogram every 3 seconds unless the qualifier **/interval=***n* is used to indicate the number of seconds between refreshes.

➤ ***Note***

MONITOR PROCESS/TOPCPU always reports a higher CPU usage than **sp_sysmon**.

```
                OPENVMS Monitor Utility
                TOP CPU TIME PROCESSES
                     on node STAR
                  1-Oct-1996 11:33:10

                           0       25      50      75     100
                           + - - - + - - - + - - - + - - - -+
4080028B  SYBASE_SQL 93    XXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                           |       |       |       |        |
4080321B  SYBASE1    86    XXXXXXXXXXXXXXXXXXXXXXXXXXX
                           |       |       |       |        |
40802C1D  SYBASE2    30    XXXXXXXXXX
                           |       |       |       |        |
4080028D  _RTA2:      1
                           |       |       |       |        |

                           |       |       |       |        |

                           |       |       |       |        |

                           + - - - + - - - + - - - + - - - -+
```

Use the **SHOW SYSTEM** command to display the amount of CPU accumulated for each engine since the last server boot. The command output looks like the following, where the last three lines show the processes for a SQL Server named Sybase running three engines.

```
$ SHOW SYSTEM

OPENVMS V6.2 on node STAR 20-Sept-1996 12:07:13.37  Uptime 21 02:00:36
 Pid     Process Name    State Pri    I/O      CPU       Pg flts  Ph.Mem
60A00101 SWAPPER          HIB  16       0  0 00:24:00.15       0     0
60A00107 ERRFMT           HIB   8   17558  0 00:02:01.65      82   141
60A00108 CACHE_SERVER     HIB  16    5689  0 00:00:07.71      63   117
60A00109 CLUSTER_SERVER   HIB  10     236  0 00:00:24.02     135   354
60A0010A OPCOM            HIB   6   31569  0 00:04:43.95    5007   244
60A0010B AUDIT_SERVER     HIB  10    1236  0 00:17:11.53    1368   423
60A0010C JOB_CONTROL      HIB  10   26396  0 00:00:44.18     157   389
60A0010D CONFIGURE        HIB  10     212  0 00:00:00.53     110   187
60A0010E SMISERVER        HIB   9     167  0 00:00:20.41     446   689
```

Routine Tasks, Maintenance, and Troubleshooting

The administration of SQL Server databases includes many routine tasks. The *System Administration Guide* covers most of these tasks in detail. This chapter describes some of the tasks that may vary

according to the type of operating system on which the SQL Server is installed. For OpenVMS these tasks can include:

- Starting, monitoring, and stopping SQL Server and Backup Server
- Checking the error logs for SQL Server and Backup Server
- Determining which type of media to use for database dumps from and loads to SQL Server
- Installing on-line syntax help sample databases

Starting, Monitoring, and Shutting Down Servers

You can start (or boot), shut down, and restart SQL Server or Backup Server using operating system commands. See the *SQL Server Utility Programs for OpenVMS* manual for more information on the Sybase utilities that perform these functions.

Monitoring Servers

The simplest way to tell whether a SQL Server is running is by logging in to it using **isql** or Data Workbench.

You can also run the **showserver** command to determine whether any SQL Server processes are running on the system:

```
$ SHOWSERVER /SERVER=SERVER_NAME
```

The **SHOW PROCESS** DCL command can also be used:

```
$ SHOW PROCESS SERVER_NAME_SQL
```

Note that you can use a similar command to determine whether any Backup Server processes are running on the system:

```
$ SHOW PROCESS SERVER_NAME_BSV
```

For example, if you type the following command with the server Sybase running:

```
$ SHOW PROCESS SYBASE_SQL
```

the output looks like the following:

```
29-OCT-1996 11:25:37.73  User: SYBASE    Process ID:   40801CC2
                    Node: STAR      Process name: "SYBASE_SQL"

Terminal:
User Identifier:    [230,17]
Base priority:      4
Default file spec:  Not available

Devices allocated:  NET5217:
                    NET5218:
```

To find out whether a server has multiple engines running, use the following command:

$ SHOW PROCESS *SERVER_NAME*_SQL /SUBPROCESSES

For example, if you type the following command with the server Sybase running:

$ SHOW PROCESS SYBASE_SQL/SUBPROCESSES

the output looks like the following:

```
29-OCT-1996 11:29:08.82  User: SYBASE  Process ID:   40801CC2
                        Node: STAR    Process name: "SYBASE_SQL"

    There are 3 processes in this job:

      SYBASE_SQL (*)
      SYBASE1
      SYBASE2
```

In the above example, the server Sybase has three engines running, with the process names *SYBASE_SQL, SYBASE1,* and *SYBASE2.*

You can also use the **SHOW SYSTEM** DCL command to see what SQL Server processes are running. See “Monitoring CPU Usage” on page 9-17 for an example of **SHOW SYSTEM** output.

Shutting Down Servers

Use a command like the following to stop a SQL Server:

$ STOPSERVER /SERVER="*SERVER_NAME*"

Use a command like the following to stop a Backup Server:

$ STOPSERVER /BACKUPSERVER="BACKUP_SERVER_NAME"
_$ /SERVER=”*SERVER_NAME*”

stopserver prompts for the password for a user account with System Administrator privileges. If for some reason you cannot connect to the server, you can use the **/abort** qualifier with **stopserver.** Use this

qualifier only in case of emergency. **stopserver** is designed to take the server down gracefully.

Logging Errors

Most error messages from SQL Server and Sybase SQL Toolset™ go to the user's terminal only. Backtraces from SQL Server, fatal error messages (severity levels 19 and above), kernel error messages, and informational messages from SQL Server are sent to an error log file.

You specify the paths for SQL Server's and Backup Server's error log files during the **sybinit** installation session. By default, **sybinit** creates SQL Server's error log file as *SYBASE_SYSTEM:[SYBASE.INSTALL] ERRORLOG* and Backup Server's error log file as *SYBASE_SYSTEM: [SYBASE.INSTALL]BACKUP.LOG*. The location of a server's error log file is set in its runserver file.

The error log files are owned by the System Administrator (or the user who ran the **sybinit** installation session). You must have write access to a server's error log file to successfully start a server. Each time a server is started with **startserver**, information on the success (or failure) of the start is appended to its error log file.

SQL Server and Backup Server keep their error log files open until you stop the server process. To reduce the size of the error log file, stop the SQL Server or Backup Server process and delete the old messages.

startserver creates an output file for the server process in the *SYBASE_SYSTEM:[SYBASE.INSTALL]* directory. This file, called *START_SERVERNAME.OUT*, may contain OpenVMS messages in addition to SQL Server error log messages. A new version of this file is created every time a server is restarted. If this file cannot be created at process creation, the server cannot start.

What's Next?

Appendix D, "Sample SQL Server System Specifications" details requirements and limitations for SQL Server on your operating system

10 Migrating SQL Server from VAX to Digital OpenVMS Alpha

This chapter provides instructions for migrating SQL Server release 10.0 on DEC VAX to release 11.0.x on Digital OpenVMS Alpha. Topics include:

See the "Road Map for Migrations" preceding the preface for an outline of migration steps.

Migration Issues

There are four main issues associated with migrating databases from a VAX to an OpenVMS Alpha:

- Datatype formats
- User-created database objects
- Database, log, and dump device locations
- SQL Server memory and stack size

Datatype Formats

The OpenVMS Alpha uses a different internal storage format for the float and real datatypes than the VAX. (The OpenVMS Alpha uses industry-standard IEEE floating point formats.) You must reformat data stored as floats and reals before you use it on an OpenVMS Alpha. System tables on the VAX can be used unaltered on an OpenVMS Alpha. User tables without float or real columns can also be used unaltered on an OpenVMS Alpha.

User-Created Database Objects

Stored procedures, triggers, rules, defaults, and views are stored in *sysprocedures* according to the natural alignment rules of the native computer architecture. Since the VAX has different alignment rules

than the OpenVMS Alpha, you must drop these objects and recreate them on the OpenVMS Alpha.

Database and Dump Device Directory Locations

Database device directory locations may be different on the OpenVMS Alpha than they are on the VAX. The location information is stored in the sysdevices and *syslogs* tables in the master database. You need to correct that information when migrating.

Dump device names may change after the migration. If they do change, you need to update these names in the sysdevices table in the master database.

Server Memory and Stack Size

SQL Server requires more memory and a larger stack size on an OpenVMS Alpha than on a VAX. You need to adjust these sizes, as well as the associated OpenVMS process quotas when migrating.

Loading OpenVMS Alpha Software From Media

You must load the new OpenVMS Alpha SQL Server software from media before you perform the migration.

1. Follow the instructions in Chapter 2, “Preparing for New Installation or Upgrades” to prepare your OpenVMS Alpha system for the software load and to record information on your worksheet that you will need to load your software.
2. Follow the instructions in Chapter 3, “Loading Software from Media” to use the **VMSINSTAL** utility to load your software from media.

Pre-Migration Preparation on the DEC VAX

Before you can migrate your SQL Server to the OpenVMS Alpha, perform the following steps:

1. Disable disk mirroring. Use the following Transact-SQL command:

   ```
   disk unmirror name = "device_name", mode = remove
   ```

   where *device_name* is the name of the device being mirrored.

2. Identify all user-supplied stored procedures, triggers, rules, defaults, and views, and find the text files used to create them. System stored procedures are recreated using **installmaster**.

   The following Transact-SQL commands list all stored procedures, triggers, rules, defaults, and views:

   ```
   select name from database_name..sysobjects where
   type in ("V","P","R","D","TR")
   ```

   The text for these objects is stored in a database's system table *syscomments*. To see this text, use the following command on each database:

   ```
   select syscomments.text from syscomments,
   sysobjects where syscomments.id = sysobjects.id
   and sysobjects.name = "object_name"
   ```

   Some users delete these rows in order to conserve space in a database.

3. In every database, identify all user tables that contain columns defined with the *float* and / or *real* datatypes. The system-supplied tables do not contain *floats* or *reals*. Use the following Transact-SQL command to identify the affected tables:

   ```
   select distinct sysobjects.name from
   sysobjects,syscolumns where sysobjects.id =
   syscolumns.id and syscolumns.type in (59,62,109)
   ```

4. Use **bcp** to copy out data from *all* tables containing *float* and / or *real* columns. Remember that **bcp** cannot copy data into or out of tables whose names are reserved words; rename such tables, if necessary.

➤ ***Note***

If you execute **bcp** from an OpenVMS Alpha client, the **bcp out** format should be in ASCII character format. If you execute **bcp** from a non-OpenVMS Alpha client, the native format of the client can be used. See the *Utility Programs for OpenVMS* for further information on **bcp**.

5. Adjust the **sp_configure** values needed to boot SQL Server on the OpenVMS Alpha. SQL Server may need a quota file, and your system resources may need to be adjusted.
6. Drop *all* stored procedures, triggers, rules, defaults, and views. **Be sure to drop system stored procedures last.** Execute the following Transact-SQL command:

drop *object_type* object_name

where *object_type* is a procedure, trigger, rule, default or view and *object_name* is the name of the object being dropped. You must unbind rules and defaults before dropping them. To drop system stored procedures, drop the *sybsystemprocs* database first and then drop the procedures in *master*.

Performing the Migration

Once you have loaded your new OpenVMS Alpha software and prepared your DEC VAX SQL Server, you are ready to begin the migration. Follow these steps:

1. Copy all of the database device files from the VAX to the OpenVMS Alpha. Make a note of the new directory and filenames for each device.
2. Use **sybinit** to create interfaces file entry(ies) for the SQL Server on the OpenVMS Alpha. See "Creating SQL Server Interfaces File Entry" on page 4-8 for instructions.
3. Copy the RUN_*SERVER_NAME*.COM and *RUN_M_SERVER_NAME*.COM files from the VAX. Edit these files to correct directory specification information for items such as the master device and error log. Change the definition of *SYBASE_SYSTEM*.
4. Issue the following command to mark the type of database so OpenVMS Alpha can read it:

```
$ buildmaster /disk="master_device" /alter="cbp_hw=3"
```

5. Boot the OpenVMS Alpha SQL Server to recover only *master* with the following operating system command:

```
$ startserver /server="server_name" /wait /masterrecover
```

6. Run **installmaster** to install the system stored procedures, as follows:

```
$ isql /user="sa" /password="password" -
_$ /input=sybase_system:[sybase.scripts]installmaster.
```

where "*sa*" is the user name of an account with System Administrator privileges and password is the password for that account.

7. Update the *phyname* column in *sysdevices* to reflect the new directory names for the various database, log, and tape dump devices with Transact-SQL commands like the following:

```
update sysdevices
   set phyname="dka100:[sybasedatabases]d_sybase.dat"
   where name="database_device_name"

update sysdevices
   set phyname = "mka100:"
   where name = "tape_device_name"
```

8. Shut down SQL Server.
9. Boot SQL Server with the following operating system command and review the error log to see if all databases are online. If a file cannot be accessed, check its directory specification in *sysdevices*.

   ```
   $ startserver /server="server_name" /wait
   ```

10. In each database, truncate tables containing *float* and / or *real* data with the following Transact-SQL command:

    ```
    truncate table table_name
    ```

    where table_name is the name of the table listed in Step 3 of "Pre-Migration Preparation on the DEC VAX" on page 10-2.
11. Reload the changed tables using **bcp** and the data saved earlier.
12. Recreate all user stored procedures, triggers, rules, defaults, and views.
13. Re-mirror any mirrored devices with the following Transact-SQL command:

```
disk mirror name="device_name", mirror="physical_name",
writes={serial | noserial }
```

    See Volume 1 of the *SQL Server Reference Manual* for information on the **disk mirror** command's syntax.
14. Dump all databases, including master and model.

◆ ***WARNING!***

Database and transaction dumps taken from a VAX SQL Server cannot be loaded into an OpenVMS Alpha Server.

What's Next?

When you have completed the tasks explained in this guide, read the *System Administration Guide* and *System Administration Guide Supplement* for information on getting started with SQL Server.

A Recovery from Failure

This appendix provides information on error messages and common failure scenarios for **sybinit** sessions.

Log Files

In case of failure during a **sybinit** session, review the messages in the **sybinit** log for information. Also review the SQL Server or Backup Server error logs. In case of failure during an upgrade session, review the logs created by the upgrade process.

sybinit Log Files

When you run a **sybinit** session, either interactively or with a resource file, a dated log file is created. This file contains a record of all the error messages generated during the session (including information not echoed to the screen during an interactive **sybinit** session). The log file also includes information about your user environment, such as the location of the SYBASE release directory, and which user invoked the session.

Log files are located in *SYBASE_SYSTEM:[SYBASE.INIT.LOGS]*, and are named *logmonthdate.version*, where month is a two-digit number (with leading zeros if necessary) from 01 to 12, date is a two-digit number from 01 to 31, and version is a three-digit number from 001 to 999. The month and date identify when the session occurred, and version identifies the particular session for days in which multiple sessions were run. For example:

```
log0323.002
```

identifies the log file for the second **sybinit** session run on March 23.

During an interactive **sybinit** session, the name of the log file for the session appears on the screen above the main "Sybinit" menu when you start **sybinit** and again when you exit **sybinit**.

You can use the **/log** option when you issue the **sybinit** command if you want to specify a different file specification for the **sybinit** log. See *Table B-1: sybinit command line qualifiers* for more information.

SQL Server and Backup Server Error Log Files

During new installation sessions, **sybinit** places the error logs for SQL Server and Backup Server in default locations. You can specify different file specifications during the interactive session or in your resource file.

- The default file specification for the SQL Server error log is *SYBASE_SYSTEM:[SYBASE.INSTALL]errorlog*.
- The default file specification for the Backup Server error log is *SYBASE_SYSTEM:[SYBASE.INSTALL]backup.log*.

During a SQL Server upgrade session, **sybinit** prompts you to specify a path for the SQL Server error log file. The default path is *SYBASE_SYSTEM:[SYBASE.INSTALL]errorlog*, where *SYBASE_SYSTEM* is the logical definition for the upgraded SQL Server. Type in a different path at the prompt, or press Return to accept the default.

Upgrade Log Files

The output from various upgrade steps is saved to log files in the *upgrade* subdirectory of the new *SYBASE* home directory (*SYBASE_SYSTEM:[SYBASE.UPGRADE]*). Examining this output after upgrade failure can help you determine the cause of failure and the upgrade phase when failure occurred.

Table A-1: Files generated during upgrade

File Name	Description
SERVER_NAME.LOG	Log of upgrade process
SERVER_NAME_CSAVE.OUT	Previous configuration settings
SERVER_NAME_CONFIG.OUT	Log of configuration changes
SERVER_NAME_UPGRADE.OUT	Detailed upgrade status log

Common *sybinit* Error Messages

Insufficient Operating System Memory

If you receive an error message similar to this:

```
INSFMEM, insufficient dynamic memory
```

you have not assigned enough shared memory in *SYSGEN*. Follow the instructions in Chapter 2 for adjusting shared memory values, reboot your operating system, and begin this installation again.

Port Already in Use

If you receive an error message similar to this:

```
kernel: ninit: bind, Address already in use
```

you have entered a port number that is already in use. You will have to begin the installation again and use a different port number.

Recovery from Upgrade Failure

The following sections describe common upgrade failure scenarios and suggest procedures to follow after different types of upgrade failures. After any upgrade failure, review the **sybinit** log file, the SQL Server error log file, and the log files generated during upgrade to determine the cause of failure and the phase when it occurred.

In most cases of failed upgrades, try to fix the problem that caused the upgrade failure and then restart the **sybinit** upgrade session. However, if failure occurs during query tree remapping, you cannot rerun **sybinit** to retry the upgrade. You must complete the upgrade manually (see “Manually Completing Upgrade” on page A-8). Be sure to correct remapping problems before beginning these steps.

Restoring from Backup

In most cases of upgrade failure, you should not have to restore your databases from backup. However, you should always first make backups of all your databases before you begin the upgrade process. See“Backing Up Existing Databases” on page 5-6 for more information about backing up your databases. If the upgrade does fail, first attempt to fix the problem that caused the upgrade failure and then rerun the **sybinit** upgrade session.

If you believe the upgrade failure or its cause may have damaged your databases, use the backup copies you made before the upgrade to restore your databases.

To restore from backup:

1. Exit **sybinit.**
2. Follow the instructions in the *System Administration Guide* to recover all user databases and system databases from backup.
3. After you have restored all databases to their condition prior to the upgrade, follow the instructions under “Starting sybinit” on page 5-13.

Recovery from New Installation Failure

Perform the following steps if your new configuration fails:

1. Read the **sybinit** log file, and the SQL Server error log, if necessary.

   Consult the log to determine the reason for the failure. Take any suggested actions to correct the problem.

2. Shut down any SQL Servers (OK to use the **STOP/ID** command if you can’t log on).

   If the configuration fails **after sybinit** boots the SQL Server you are attempting to configure, shut down that server.

3. Manually remove any SQL Server device files.

   If the configuration fails **after sybinit** has created any RMS files (for example, the master device or *sybsystemprocs* device), delete the device files (if the file is locked, you must wait until the server is shut down) and the RUN_SERVER.COM file, if it exists.

   You do **not** need to delete the interfaces file.

4. Restart the entire configuration process from the beginning.

   When you reach the “SQL Server Configuration” menu, if “Configure Server's Interfaces File Entry” is marked “Complete”, you added the interfaces file entry for the new server during the first installation attempt and you do not have to add the entry a second time.

Failure to Boot New SQL Server Release 11.0.x

If the upgrade process returns a message to the screen that it failed to boot the SQL Server release 11.0.x, check the SQL Server error log (in the location that you specified just prior to executing the upgrade) for start-up error messages.

Fix any problems noted in the error log. Shut down the SQL Server 11.0.x, if it is running, and restart your pre-11.0.x SQL Server. Confirm that the SQL Server booted successfully (for example, by executing the **isql** utility). Then return to "Starting sybinit" on page 5-14, and perform the steps necessary to complete the upgrade.

Insufficient Space for Upgrade

You should be able to detect the problem of insufficient space by executing the pre-upgrade eligibility test. If, however, the **sybinit** error log indicates that upgrade failure occurred because your existing databases do not have enough space, follow the instructions in "Failing the Pre-Upgrade Test" on page 5-17 to increase database space according to the requirements listed in the error log. Then go to "Executing the Upgrade" on page 5-21 to complete the upgrade.

Failure to Remap Database Objects

The upgrade of SQL Server to release 11.0.x includes a process called "query tree remapping," which restructures all stored procedures, triggers, rules, defaults and views. This remapping provides improved performance and allows for new features.

Query tree remapping occurs after the version number for SQL Server has been updated. You cannot rerun a **sybinit** upgrade session on a SQL Server after query tree remapping fails, because the updated version number does not allow you to initiate an upgrade session.

If errors occur in the remapping of five consecutive objects, the remapping process terminates. If this type of failure occurs, follow these steps:

1. Complete the query tree remapping following the instructions in "Stack Size and Remapping" below.
2. Manually complete the upgrade according to the instructions in "Manually Completing Upgrade."

Stack Size and Remapping

Insufficient stack space is a common cause of query tree remapping failure. You can avoid this problem by temporarily altering the SQL Server's configuration to increase stack size. Use the following instructions to increase stack size:

1. Use **isql** to log into SQL Server.

```
$ isql /user="sa" /password="password" /server="servername"
```

2. Execute **sp_configure** to display current configuration values and record the values for "user connections" and "stack size". (After you attempt the remapping process again, you should reconfigure SQL Server to these original values.)

```
1> sp_configure
2> go
```

3. Execute **sp_configure** to reduce user connections to 10 and stack size to 327680 (10 times the default).

```
1> sp_configure "user connections", 10
2> go
1> sp_configure "stack size", 327680
2> go
```

➤ ***Note***

If you increase the stack size too much, you must also reduce the number of user connections. If the stack size is too large, SQL Server may not boot because it will not have enough memory allocated for data and procedure cache.

4. Execute the **reconfigure** command to install the changed configuration values.

```
1> reconfigure
2> go
```

5. Shut down SQL Server.

```
1> shutdown
2> go
```

6. Restart SQL Server.

```
$ startserver /server="server_name"
```

Restarting Remapping

1. After you have increased stack size, you can select the **sybinit** option "Remap the query trees in all databases" to restart the remapping process.

   This option appears in the "SQL Server Upgrade" menu after the first upgrade attempt. It also appears in the "SQL Server

Configuration" menu when you access this menu through the "Configure an Existing SQL Server" path.

2. **sybinit** saves output from this option to the file *SYBASE_SYSTEM:[SYBASE.INIT.LOGS]REMAP.DMP*. Review the contents of this file to determine whether remapping was successful.
3. If remapping was successful, execute **sp_configure** to restore "user connections" and "stack size" to their original values.

   If remapping failed for fewer than five consecutive objects, you may want to drop and recreate the specific objects listed in the log file that failed remapping, rather than reattempt the entire remapping.

   Refer to Volume 1 of the *SQL Server Reference Manual* for information on dropping and creating database objects.
4. If the remapping process terminated because of failure to remap five consecutive objects, refer to the following section.

Remapping Workaround

If remapping still terminates after you have increased stack size:

1. See the following files to determine which object's query tree caused the termination.

SYBASE_SYSTEM:[SYBASE.UPGRADE]server_name*_UPGRADE.*out
SYBASE_SYSTEM:[SYBASE.INIT.LOGS]remap.dmp

2. Use the **defncopy** utility to save the text definition of the object to a file. See *SQL Server Utility Programs for OpenVMS* for information on **defncopy.**
3. Drop the object.
4. Reselect the **sybinit** option "Remap the query trees in all databases."
5. After each remapping failure, repeat steps 1-4 as many times as necessary, until the remapping process completes without termination.
6. Recreate the objects you dropped. See the *SQL Server Reference Manual* for information on recreating database objects.

Manually Completing Upgrade

If failure occurs during query tree remapping, you must manually complete the upgrade process after you have successfully completed remapping. Perform the following steps to complete the upgrade:

1. Use the **disk init** command to create a device for the *sybsystemprocs* database.

   The device must be at least 21MB. Use the specification and size that you recorded on your worksheet. See Chapter 2 for information on selecting and preparing devices. See the *SQL Server Reference Manual* for details on **disk init**.

2. Use **isql** to log in to SQL Server and execute the **create database** command to create the *sybsystemprocs* database.

   The database must be at least 21MB. See the *SQL Server Reference Manual* for details on **create database**.

3. Use **isql** to execute the **installmaster** script.

◆ ***WARNING!***

Do not run the installmaster script repeatedly without dropping the *sybsystemprocs* database. Running installmaster repeatedly can change the distribution of index values in such a way that the *sysprocedures* table will require much more disk space to store the same amount of data. To avoid this problem, drop and re-create the *sybsystemprocs* database before running installmaster.

```
$ set default sybase_system:[sybase.scripts]
$ define dsquery server_name
$ isql /user="sa" /password="password" /input=installmaster
```

4. Use **isql** to execute the **installmodel** script.

```
$ set default sybase_system:[sybase.scripts]
$ define dsquery server_name
$ isql /user="sa" /password="password" /input=installmodel
```

5. Copy the *runserver* file from the old *SYBASE_SYSTEM* directory (entered on your worksheet as the "old" *SYBASE_SYSTEM* logical name) to the new release 11.0.x *SYBASE_SYSTEM* directory (entered on your worksheet as the "new" *SYBASE_SYSTEM* logical name). Edit this file so that it points to the new *SYBASE_SYSTEM* directory. For example, a *runserver* file that had the following line for a 10.0.2 SQL Server:

```
$define SYBASE_SYSTEM SYBASE_1002
```

Would be edited as follows to point to the 11.0 *SYBASE_SYSTEM* directory

```
$define SYBASE_SYSTEM SYBASE_110
```

6. Shut down and reboot the new SQL Server.

B *sybinit* Resource Files

This appendix provides instructions for using **sybinit** resource files to configure or upgrade a SQL Server, or to configure a Backup Server. It also provides a list of command line qualifiers available for resource files or interactive **sybinit** sessions. See chapters 4–7 for instructions on using **sybinit** interactively.

Generating Resource Files

To install, upgrade, or reconfigure more than one SQL Server or Backup Server, you can generate a resource file that records the configuration specified during your interactive **sybinit** session. You can then use and reuse the resource file as necessary. You can generate a resource file that duplicates the configuration specified during your interactive **sybinit** session, as follows:

1. Press Ctrl-w after you have made entries for all required menu items.

   sybinit prompts with the phrase "Dump out all current attributes?"

2. Type "yes" to indicate that you want to generate a resource file containing the attributes you specified during the interactive session.

   sybinit prompts you for the path of the resource file to generate.

3. Press Return to accept the default path of SYBASE_SYSTEM:[SYBASE.INIT.LOGS]RESOURCE.DMP, or enter another path.

4. Edit the file to change device names before you execute any resource file sessions.

   You can cause problems with the devices created interactively if you do not change the device names.

The file that is generated when you press Ctrl-w during an interactive sybinit session only partially duplicates your configuration. This file contains the attribute values that you entered during your interactive session but is missing some required values and contains additional attributes. You can use this file as a reference to edit a copy of one of the resource file templates included with SQL Server, and use the edited copy of the template for a **sybinit** resource file session.

In addition to editing the resource file attributes, you must change device names in the file before you can use it in any resource file sessions. The device names must be unique for each SQL Server you configure.

Starting *sybinit* with a Resource File

To use **sybinit** with a resource file:

1. Log in using the account you created in Chapter 2.
2. Verify that SYBASE_SYSTEM was defined correctly when the Sybase software was installed from media.

   $ SHOW LOGICAL SYBASE_SYSTEM

   The value returned should match the SYBASE_SYSTEM definition for the new SQL Server installation, as recorded on your worksheet. Depending on the value returned, go on to either step 3 or step 4.
3. If the values do not match, type the appropriate command at the prompt:

   For a primary installation:

   $ DEFINE /SYS/EXEC SYBASE_SYSTEM *ROOTED_LOGICAL*

   For a secondary installation:

   $ DEFINE SYBASE_SYSTEM *ROOTED_LOGICAL*

   where *rooted_logical* is the SYBASE_SYSTEM definition for the new SQL Server installation, as recorded on your worksheet.
4. Make sure the **sybinit** symbol is defined:

   $ SHOW SYMBOL SYBINIT

   If the symbol is not defined, execute the following:

   $ @SYBASE_SYSTEM:[SYBASE.INSTALL]SYBASE_MANAGER

   If the symbol is defined, go to step 5.
5. Start **sybinit** with the following syntax:

$ SYBINIT /RESOURCE=*RESOURCE_FILE_NAME* [*/QUALIFIER*] [=*VALUE*]

where *qualifier* specifies any additional command line qualifier(s) that you need, and *value* is any accepted value for that qualifier(s).

You can specify more than one command line qualifier.

For a resource file session, you must specify the **/resource** qualifier and the resource file name. *Table B-1: sybinit command line qualifiers* lists options available for a resource file. Some of these qualifiers can also be used in an interactive **sybinit** session.

➤ ***Note***

If you enter certain values for particular attributes, **sybinit** exits from the resource file session with a warning (for instance, if you specify an operating system file for the location of the master device). If you start a **sybinit** resource file session with the **/trace= IGNORE_WARNINGS** flag, the **sybinit** session continues, instead of exiting.

Table B-1: *sybinit* command line qualifiers

qualifier	Valid Values	Explanation
/charset	iso_1, cp437, cp850, mac, roman8, ascii_8*	Specifies the character set to be used for all messages and prompts. This qualifier forces **sybinit** to use a particular character set. Without this qualifier, **sybinit** uses the default character set for your system. **Available with default language of U.S. English only*
/environment	n / a	Records information about the user's environment and exits. Information is written to the log file. Without this qualifier, sybinit records user environment information and continues with the product configuration.
/help	n / a	Prints out usage message and exits.
/language	french, german	Specifies the language to be used for all messages and prompts. This qualifier forces sybinit to use a particular language.
/log	Full specification of log file	Lets user specify location of the session log file. Without this qualifier, sybinit uses the default log location *SYBASE_SYSTEM:[SYBASE.INIT.LOGS]*.
/resource	Full specification of resource file	Specifies that sybinit is to take input from the specified resource file. Use this qualifier when doing non-interactive configurations and upgrades.
/sybase	Full specification of *SYBASE* release directory	Lets user specify the SYBASE release directory. If this qualifier is not used, **sybinit** assumes that the SYBASE release directory is *SYBASE_SYSTEM:[SYBASE*].

Table B-1: *sybinit* command line qualifiers

qualifier	Valid Values	Explanation
/trace=IGNORE_ WARNINGS	n / a	Causes resource file session to continue after **sybinit** issues a warning, rather than exit.
/validate	n / a	Validates the resource file and exits. Without this qualifier, **sybinit** validates the resource file and continues with the product configuration. The **/resource** qualifier must be used with this qualifier.
/version	n / a	Prints out the **sybinit** version string and exits.

Resource File Templates

Sybase supplies the following resource file templates:

- *new.rs* – for new configurations and reconfiguring existing SQL Servers
- *upgrade.rs* – for upgrading SQL Server to release 11.0.x

  These templates are located in the *SYBASE_SYSTEM:[SYBASE.INIT.SQLSRV]* directory.
- *resource* – for configuring Backup Server

  This template is located in the *SYBASE_SYSTEM:[SYBASE.INIT.BSRV]* directory.

Resource files are in ASCII format, and can be modified with any ASCII editor, such as **edt** or **eve**.

Make a copy of the template you want to use, and rename it to distinguish it from the original. For example, enter the following command at the operating system prompt:

```
$ COPY NEW.RS RESOURCE.TESTSERVER1
```

You should have read and write permissions on the copy of the resource file template that you want to edit. You may need to grant write permission to yourself as the file owner. For example, enter the following command at the operating system prompt:

```
SET FILE/PROTECTION=(O:RWED) RESOURCE_FILE
```

where resource_file is the name of the copy you'll be editing.

➤ ***Note***

Sybase recommends that you grant read and write permissions on SQL Server resource files only to yourself as file owner. Any users who have read permission on the resource file can see the "sa" password.

Editing Resource Files

The resource file templates list attributes to be defined during the **sybinit** session and values for these attributes. Most attribute entries in the resource file are prefaced by a comment line that briefly describes the item. The entries themselves are in the format:

```
PRODUCT_NAME.ATTRIBUTE:VALUE
```

where *product_name* is the product you are configuring or upgrading, attribute refers to the particular item for which you are supplying information (such as network protocol, or the name for SQL Server) and *value* refers to the actual value you supply for the attribute.

After you copy a resource file template, edit it by modifying values for the listed attributes.

Lists of Values

Certain attributes in the resource files can take lists of values rather than just single values. (The attributes that can accept lists of values all end in *_list*.) If you enter multiple values for a single attribute, the values need to be separated by a comma. For example:

```
SQLSRV.LANGUAGE_INSTALL_LIST: FRENCH,GERMAN
```

Multiple Line Entries

Each entry in a resource file can be only one line long (80 characters). **sybinit** does not recognize end-of-line characters, carriage returns, or backslashes (\).

USE_DEFAULT Value

In the resource file template for new installations some attributes have the value USE_DEFAULT. This value indicates that a Sybase-defined default exists for this attribute. If you want to use the default

value, leave the attribute unchanged. If you want to use a non-default value, simply replace USE_DEFAULT with the value you want. You cannot enter USE_DEFAULT for some attributes, because they default to NULL. See *Table B-2: SQL Server resource file attributes*.

UNCHANGED Value

If you are using a resource file to upgrade or modify an existing SQL Server, use the UNCHANGED value to indicate that a particular attribute should remain unchanged from its current status. For such attributes, simply enter the value UNCHANGED. Note that you cannot use UNCHANGED for attributes that default to NULL.

Attributes in Resource Files

Table B-2 and Table B-3 describe the attributes defined in SQL Server and Backup Server resource file templates, and note whether they are required to be listed in a resource file. For required attributes that take "yes" or "no" values, you can specify "no" if you do not want the action executed. If you do not have a value for other required attributes, you may specify "USE_DEFAULT" or "UNCHANGED".

◆ ***WARNING!***

You cannot enter "USE_DEFAULT" or "UNCHANGED" for required attributes with a default value of NULL. If you do not enter a valid value for every required attribute, your resource file session fails.

The **sqlsrv.sa_password** attribute is an exception to the rules for required attributes with null defaults. For new configurations, enter USE_DEFAULT for this attribute. If you enter any other value, your resource file session fails. Follow the instructions in "Setting the System Administrator Password" on page 4-24 to set a password after your resource file session has completed.

For upgrades or reconfigurations, enter USE_DEFAULT for this attribute if you do not want to change the password. Enter the actual

password only if you want to change it during an upgrade or reconfiguration session.

SQL Server Attributes

Table B-2: SQL Server resource file attributes

SQL Server Attribute	Description	Required	Default Value	Other Valid Values
sybinit.release_directory:	Directory where Sybase files were copied to from media.	Yes	null	any directory specification
sybinit.product:	The product to configure or upgrade.	Yes	null	sqlsrv
sqlsrv.server_name:	Name for this SQL Server.	Yes	SYBASE	any server name
sqlsrv.new_config:	Use “yes” for new installations, “no” for upgrades from previous releases, and “no” for modifying existing SQL Servers.	Yes	No	Yes
sqlsrv.do_add_server:	Specifies whether or not to add this server to the interfaces file.	Yes	No	Yes
sqlsrv.connect_retry_count:	Specifies number of times client programs try to connect to SQL Server before giving up.	No	0	any positive integer
sqlsrv.connect_retry_delay_time:	Specifies amount of time (in seconds) a client program waits between attempts to connect to SQL Server.	No	0	any positive integer
sqlsrv.server_notes:	Specifies optional comments to add to the interfaces file entry.	No	null	any text

Table B-2: SQL Server resource file attributes (continued)

SQL Server Attribute	Description	Required	Default Value	Other Valid Values
sqlsrv.network_protocol_list:	Specifies network protocol or protocols used by the network listener. Can take multiple values.	Yes, *if sqlsrv. do_add_ server set to yes*	decnet	tcp
sqlsrv.network_name_alias_list:	Specifies the alias or aliases used by the network listener. Can take multiple values.	No	null	any alias name(s)
sqlsrv.network_hostname_list:	Name of host machine for SQL Server. Can take multiple values.	Yes, *if sqlsrv. do_add_ server set to yes*	null	any host name(s)
sqlsrv.network_port_list:	Specifies port or object number for network listener. Can take multiple values.	Yes, *if sqlsrv. do_add_ server set to yes*	null	5000–65535 (ports) 128–153 (objects)
sqlsrv.master_device_physical_nam e:	Specifies the full file specification of the master device.	Yes	null	any full device specification
sqlsrv.master_device_size:	Specifies the size of the master device, in megabytes.	Yes	24	any device size
sqlsrv.sa_login:	Enter the name of the user login that has the SA role.	Yes	sa	any user name
sqlsrv.sa_password	Enter the password of the user login that has the SA role.	Yes	null	See page B-6
sqlsrv.default_backup_server:	Specifies a name for the Backup Server that SQL Server will use.	Yes	SYB_ BACKUP	any Backup Server name
sqlsrv.errorlog:	Specifies the location of the SQL Server error log.	No	*SYBASE_ SYSTEM: [SYBASE. INSTALL] errorlog.*	any file name (full specification)

Table B-2: SQL Server resource file attributes (continued)

SQL Server Attribute	Description	Required	Default Value	Other Valid Values
sqlsrv.do_upgrade:	Use "yes" to upgrade SQL Server from a previous release level. Use "no" when configuring a new SQL Server or modifying an existing SQL Server.	Yes	No	Yes
sqlsrv.default_language:	Specifies the default language for SQL Server.	Yes	us_english	french german
sqlsrv.language_install_list:	Specifies which languages to install for SQL Server. Can take multiple values. *This list must contain the value for sqlsrv.default_ language, unless it is U.S. English.*	No	null	french german
sqlsrv.language_remove_list:	Specifies languages to remove from SQL Server. Can take multiple values.	No	null	french german
sqlsrv.default_characterset:	Specifies the default character set for SQL Server. ** Available with default language of U.S. English only*	Yes	iso_1	ascii_8* roman8 cp850 cp437 mac
sqlsrv.characterset_install_list:	Specifies which character sets to install for SQL Server. Can take multiple values. *This list must contain the value for sqlsrv.default_ characterset, unless the value is iso_1.*	No	null	ascii_8* roman8 cp850 cp437 mac

Table B-2: SQL Server resource file attributes (continued)

SQL Server Attribute	Description	Required	Default Value	Other Valid Values
sqlsrv.characterset_remove_list:	Specifies which character sets to remove from SQL Server. Can take multiple values.	No	null	ascii_8* roman8 cp850 cp437 mac
sqlsrv.sort_order:	Specifies the sort order to use for SQL Server. *See [SYBASE.CHARSETS. CHARSET_NAME] charset.loc file to determine available sort orders*	Yes	binary	Varies by installed character set
sqlsrv.do_install_auditing:	Use "yes" to install the auditing feature. Otherwise, use "no".	No	No	Yes
sqlsrv.auditing_db_size:	Specifies the size (in megabytes) of the auditing database. ****Attribute is required if value for sqlsrv.do_ create_auditing_ device is yes*	No***	Null	any database size
sqlsrv.auditing_device_logical_name:	Specifies the logical name of the auditing database.	No***	*sybsecurity*	any device name
sqlsrv.auditing_device_physical_name:	Specifies the name of the physical device on which to install the auditing database.	No***	null	any device name
sqlsrv.auditing_device_size:	Specifies the size (in megabytes) of the auditing database.	No***	null	any database size
sqlsrv.do_create_auditing_device:	Specifies whether or not to create the auditing device.	No	No	Yes

Table B-2: SQL Server resource file attributes (continued)

SQL Server Attribute	Description	Required	Default Value	Other Valid Values
sqlsrv.sybsystemprocs_db_size:	Specifies the size (in megabytes) of the *sybsystemprocs* database. ****Attribute is required if value for sqlsrv. do_create_ sybsystemprocs_device is yes*	No****	21	any device size
sqlsrv.sybsystemprocs_device_ logical_name:	Specifies the logical name for the *sybsystemprocs* database.	No****	*sysprocsdev*	any device name
sqlsrv.sybsystemprocs_device_ physical_name:	Specifies the name of the physical device for the *sybsystemprocs* database.	No****	null	any full specification device name
sqlsrv.sybsystemprocs_device_size:	Specifies size (in megabytes) of the physical device on which to install the *sybsystemprocs* database.	No****	21	any device size
sqlsrv.do_create_sybsystemprocs_ device:	Specifies whether or not to create the *sybsystemprocs* device. *If you enter a value of no for this attribute, sybsystemprocs database is placed on master device*	No	Yes	No
sqlsrv.master_host:	Specifies the master node for server. Must be set if using Companion Servers.	Only if using CFT	null	any host name(s)
sqlsrv.companion_node_list:	Lists hosts for companion servers.	Only if using CFT	null	any host name(s)

Table B-2: SQL Server resource file attributes (continued)

SQL Server Attribute	Description	Required	Default Value	Other Valid Values
sqlsrv.companion_protocol_list:	Specifies network protocols for companion servers. Can take multiple values.	Only if using CFT	decnet	tcp
sqlsrv.companion_port_list:	Specifies port or object number for companion network listeners. Can take multiple values.	Only if using CFT	null	5000–65535 (ports) 128–153 (objects)
sqlsrv.quota_file:	Specifies file specification for quota file.	No	null	any file name (full specification)

Backup Server Attributes

Table B-3: Backup Server resource file attributes

Backup Server Attributes	Description	Required	Default Value	Other Valid Values
sybinit.release_directory:	Directory where Sybase files were copied to from media.	Yes	null	any directory specification
sybinit.product:	The product to configure or upgrade.	Yes	null	bsrv
bsrv.do_add_backup_server:	Specifies whether or not to add Backup Server to the interfaces file.	Yes	Yes	No (use if reconfiguring existing Backup Server)
bsrv.server_name:	Name for Backup Server.	Yes	SYB_ BACKUP	any Backup Server name
bsrv.errorlog:	Specifies the location of Backup Server's errorlog.	No	*SYBASE_ SYSTEM: [SYBASE. INSTALL] backup.log*	any full file specification
bsrv.connect_retry_count:	Specifies number of times client programs try to connect to Backup Server before giving up.	No	0	any positive integer
bsrv.connect_retry_delay_time:	Specifies amount of time (in seconds) a client program waits between attempts to connect to Backup Server.	No	0	any positive integer
bsrv.network_protocol_list:	Specifies network protocol(s) used by the network listener. Can take multiple values.	Yes, *if bsrv. do_add_ server set to yes*	decnet	tcp
bsrv.network_name_alias_list:	Specifies the alias used by the network listener. Can take multiple values.	No	null	any alias name(s)

Table B-3: Backup Server resource file attributes (continued)

Backup Server Attributes	Description	Required	Default Value	Other Valid Values
bsrv.network_hostname_alias_list:	Name of host machine for Backup Server. Can take multiple values.	Yes, *if bsrv.do_add_server set to yes*	null	any host name(s)
bsrv.network_port_list:	Specifies port or object number for network listener. Can take multiple values.	Yes, *if bsrv.do_add_server set to yes*	null	5000–65535 (ports) 128–253 (objects)
bsrv.language:	Specifies the language for Backup Server.	Yes	us_english	french german
bsrv.character_set:	Specifies the character set for Backup Server.	Yes	iso_1	roman8 cp850 cp437 mac
bsrv.quota_file:	Specifies file specification for quota file.	No	null	any full file specification

C Sample Sessions

Included in the SQL Server release directory are online sample sessions. Use these sample sessions as a guide for the installation and upgrade process.

These ASCII files are samples of:

- Sample Installation Session
- Sample Upgrade Session
- Sample **VMSINSTAL** Session

Table C-1 lists the names of the ASCII files containing the sample sessions. See page C-2 for a sample **VMINSTAL** session. Refer to Chapter 3, "Loading Software from Media," for details about running the **VMSINSTAL** utility.

Table C-1: Sample session ASCII files

Task	ASCII File Name	See
Loading software with **VMINSTAL**	*VMSINSTAL.SPL*	page C-2; Chapter 3
Installing SQL Server with **sybinit**	*INSTALL.SPL*	Chapter 4
Upgrading SQL Server	*UPGRADE.SPL*	Chapter 5
Using resource files	*RESOURCE.SPL*	Appendix B

Sample Installation Session

The table below summarizes the details of the configuration used in the sample session for installing a new SQL Server.

Item	Response
SYBASE home directory:	*SYBASE_SYSTEM:[SYBASE]*
SQL Server name:	TESTER
Backup Server name:	TESTER_BSRV

Sample Upgrade Session

The sample session, *upgrade.spl*, contains the text of a **sybinit** session used to upgrade an existing SQL Server 4.9.2 to SQL Server release 11.0.x. Table C-2 below summarizes the details of the configuration. For instructions on upgrading SQL Server to 11.0.x, read Chapter 5 of this guide.

Table C-2: Sample configuration for upgraded SQL Server

Item	Response
Sybase release 11.0.x directory:	*ETF11:[SYBASE]*
Sybase release 10. 0.2 directory:	*10.0.2_SERVER:[SYBASE491]*
SQL Server Name:	SYS10SRV
System Procedures Database Configuration	
Database size:	21MB
Internal Device Name:	*sysprocsdev*
Physical Device Name:	*ETF3:[DEVICES]SYB_PROCS.DAT*
Physical Device Size:	21MB

Sample *VMSINSTAL* Session

Following is a sample of a **VMSINSTAL** session in which the Sybase distribution files are copied in from media. Note that your session may differ from this sample.

In this sample script, user input is displayed in **bold** type.

```
$ @sys$update:vmsinstal sybase020 MKB100:
OpenVMS AXP Software Product Installation Procedure V6.2

It is 08-Nov-1996 at 21:01.

Enter a question mark (?) at any time for help.

%VMSINSTAL-W-NOTSYSTEM, You are not logged in to the SYSTEM account.
%VMSINSTAL-W-ACTIVE, The following processes are still active:
       _NTY2:
* Do you want to continue anyway [NO]? yes
* Are you satisfied with the backup of your system disk [YES]?

Please mount the first volume of the set on  MKB100:.
* Are you ready? yes
%MOUNT-I-MOUNTED, SYBASE mounted on MKB100:
The following products will be processed:

SYBASE V2.0

        Beginning installation of SYBASE V2.0 at 21:02
```

```
%VMSINSTAL-I-RESTORE, Restoring product save set A ...
******************************************************************************
          SYBASE (System 11) Product Distribution Procedure
******************************************************************************
Except for Customer Authorization String entry and product choice entry,
      enter "?" at any prompt to receive help.

* Enter the definition of SYBASE_SYSTEM: sybase_110

A Secondary Installation allows two SYBASE product lines to co-exist
on the same cpu.  The Primary Installation is intended to be the
production environment.  The Secondary Installation  is to be used as
a test environment, or conceivably, a second Primary system.

The main difference between the two types of installation is the way
SYBASE logicals are defined.  Logicals defined by the SYBASE
definition command file SYLOGICALS.COM are defined with the "/SYSTEM"
qualifier in a Primary system.  In a Secondary installation, these
logicals are defined at a process level, so that the same SYBASE logical
names are used without overwriting the Primary SYBASE logicals.  It is
then up to each user of the Secondary system to execute the Secondary
SYLOGICALS.COM command file, and it is suggested that these users
place the following command in their LOGIN.COM:

"$ @'sybase_secondary_path':[sybase.install]sylogicals.com"

where sybase_secondary_path is the logical or device entered as the
SYBASE_SYSTEM path during installation of the Secondary system.

If sybase_secondary_path is a logical, it should be defined in
SYS$MANAGER:SYSTARTUP.COM as a system logical.

* Is this a secondary installation [NO]? yes

Please enter your Customer Authorization String, letters only:
>TOTOXR MRYSTO URURPP OZTOGO MSMSPW ZQHO

Select the Sybase products to be installed:
Product No.  1:  SQL Server, 11.0.2, OpenVMS
Product No.  2:  SQL Monitor Server, 11.0.2, OpenVMS
Product No.  3:  SQL Monitor Historical Server, 11.0.2, OpenVMS
Product No.  4:  Language Module French Server, 11.0.2, OpenVMS
Product No.  5:  Language Module German Server, 11.0.2, OpenVMS
Product No.  6:  Language Module Spanish Server, 11.0.2, OpenVMS
Please enter the Product Numbers that you wish to install, one per line.
Terminate your entries with a blank line.

>1
>2
>3
>4
>5
>6
>
The following products were chosen for installation:
```

```
Choice No.  1:  SQL Server, 11.0.2, OpenVMS
Choice No.  2:  SQL Monitor Server, 11.0.2, OpenVMS
Choice No.  3:  SQL Monitor Historical Server, 11.0.2, OpenVMS
Choice No.  4:  Language Module French Server, 11.0.2, OpenVMS
Choice No.  5:  Language Module German Server, 11.0.2, OpenVMS
Choice No.  6:  Language Module Spanish Server, 11.0.2, OpenVMS
If this list is correct as shown,

please enter 'y' or 'Y' to continue, 'q' or 'Q' to quit, or any other
character to make another set of choices: y

%SYBASE-I-RESTORE_SAVESET, Restoring SQL Server, Release 11.0.2 Files...
%SYBASE-I-RESTORE_SAVESET, Restoring SQL Monitor Server, Release
11.0.2 Files...
%SYBASE-I-RESTORE_SAVESET, Restoring SQL Monitor Historical Server,
Release 11.0.2 Files...
%SYBASE-I-RESTORE_SAVESET, Restoring Language Module, French Server,
Release 11.0.2 Files...
%SYBASE-I-RESTORE_SAVESET, Restoring Language Module, German Server,
Release 11.0.2 Files...
%SYBASE-I-RESTORE_SAVESET, Restoring Language Module, Spanish Server,
Release 11.0.2 Files...
%SYBASE-I-DECOMP_SAVESET, Decompressing contents of saveset 1102_ds.b
%SYBASE-I-DECOMP_SAVESET, Decompressing contents of saveset 1102_sqlms.a
%SYBASE-I-DECOMP_SAVESET, Decompressing contents of saveset 1102_hist.a
%SYBASE-I-DECOMP_SAVESET, Decompressing contents of saveset 1102_lmsf.y
%SYBASE-I-DECOMP_SAVESET, Decompressing contents of saveset 1102_lmsg.y
%SYBASE-I-DECOMP_SAVESET, Decompressing contents of saveset 1102_lmss.y
%SYBASE-I-CREATE_TREE, Creating the Sybase Directory Structure...
%SYBASE-I-WRITE_SYLOGICALS, Writing the Sybase Definitions Command file
-SYBASE-I-WRITE_SYLOGICALS,
SYBASE_SYSTEM:[SYBASE.INSTALL]SYLOGICALS.COM...
%SYBASE-I-SETPERMS, Setting Permissions on the SYBASE files...
**********************************************************************
 The Sybase Product Distribution has successfully completed.
**********************************************************************
 Installation of SYBASE V2.0 completed at 21:48

Adding history entry in VMI$ROOT:[SYSUPD]VMSINSTAL.HISTORY

Creating installation data file: VMI$ROOT:[SYSUPD]SYBASE020.VMI_DATA

VMSINSTAL procedure done at 21:48
```

D Sample SQL Server System Specifications

Table D-1: Sample SQL Server system specifications

System Specifications		
Hardware		
Processor	64-bit	
RAM required for SQL Server	20-24 Mb	
Minimum RAM per additional user	56K	With default stack size and packet size
Database Statistics		
Databases per SQL Server	32,767	Note that practical limit is approximately 100
Default database size	2 Mb	
Default stack size	34816	minimum 34816
Maximum database size	2^{38} (256 gigabytes)	
Minimum database size	Default (2 Mb)	
Creation of clustered index	Free space needed in database is (1.1 * size of table) + 10%	
Maximum size of a database device	2^{31} (2 gigabytes)	
Maximum number of database devices per server	255	
Maximum number of devices or device pieces per database	128	
Maximum number of segments per database	31	
Maximum number of data bytes per text/image page	1800	
Table Statistics		
User objects per database	2^{31} - 100	
Columns per table	250	
Indexes per table	250 (one clustered)	

Table D-1: Sample SQL Server system specifications (continued)

System Specifications		
Rows per table	Limited by available storage	
Maximum size of a row	1960 bytes	
Columns per composite index	16	
Bytes per index key	256	
Characters per database object name	30	
Query Statistics		
Maximum number of databases participating in one transaction	16	Includes database where transaction began, all databases changed during transaction, and *tempdb* if it is used for results or work tables
Practical number of databases participating in one query	16 (dependent on available memory)	Includes each occurrence of each database queried, and *tempdb* if it is used for results or worktables
Maximum number of tables participating in a query	16	Includes all work tables, result tables, tables referenced by views (the view itself is not counted), correlations, and self-joins
Maximum number of where clauses referencing a table in a query	128	
System Statistics		
Number of buffers and procedure buffers	Configurable	Limited by amount of RAM and maximum size of shared memory segment
Minimum memory required per stored procedure	2K	
Maximum number of parameters per stored procedure	255	

E Available Character Sets and Sort Orders

Character Sets

By default, SQL Server and Backup Server have character set definition files for the following character sets:

Table E-1: Platforms and their native character sets

Platform	Native Character Set
Sun Solaris	ISO 8859-1
NCR System 3000	ISO 8859-1
Digital OpenVMS	ISO 8859-1
IBM RS/6000 AIX	ISO 8859-1
SCO UnixWare	ISO 8859-1
IBM PC	Code Page 437 Code Page 850
Macintosh	Macintosh Roman
HP-UX	Roman8

These character sets support the Western European languages on the specified platforms. Each platform has a native character set and, by default, **sybinit** configures SQL Server and Backup Server with your platform's native character set.

Selecting the Default Character Set

The **default character set** is the character set in which data is encoded and stored on the SQL Server databases. You can select any character set as the default on your SQL Server or Backup Server, including character sets that are not native to the platform. Determine SQL Server's default character set based on the character set used by your client. For example, if most of your clients use CP 850, you should specify CP 850 on your server in order to minimize the amount of code conversion that has to take place. You can do this even if your server is an AIX (whose native character set for the Western European languages is ISO 8859-1).

Conversion Between Character Sets

In a heterogeneous environment, SQL Server and Backup Server may need to communicate with clients running on different platforms using different character sets. To maintain data integrity, the server converts the code between the character sets. You need to install the character set definition files on the server for all the character sets that your clients are using to enable automatic code conversion.

Backup Server returns messages to SQL Server in the client's language and SQL Server's character set. SQL Server then converts the messages and issues them in the client's language and character set.

If SQL Server or Backup Server does not support a client's language or character set, it issues a warning message to this effect. Errors also occur if the Backup Server character set is not compatible with the SQL Server character set.

➤ ***Note***

Code conversion is only supported between character sets for the same language or group of languages. For example, automatic code conversion is supported between the character sets for the Western European languages (ISO 8859-1, CP 437, CP 850, Mac, and Roman 8). For more information on supported conversions, refer to the *System Administration Guide*.

Available Character Sets

Platforms have native character sets. By default, **sybinit** configures SQL Server and Backup Server with the native character set for your platform. You can specify a different character set for SQL Server or Backup Server during configuration.

In heterogeneous environments, SQL Server and Backup Server may need to communicate with other servers or applications running on a platform with a different default character set. You can prevent data translation delays by installing the other platform's default character set as SQL Server's and Backup Server's default. Alternatively, you can install additional character sets to SQL Server to allow SQL Server to translate data from servers and clients using other character sets.

Your SQL Server Release 11.0.x software includes character set definition files (*charset.loc* files) in the directory for each character set under *[SYBASE.CHARSETS]*. These files permit you to configure your system for optimum server-client and server-server communication.

The character sets available to most customers include:

- iso_1 (ISO-8859 / 1) – 8-bit character set used for many systems. This is the default for SQL Server with Sun, NCR System 3000, OpenVMS, and IBM AIX workstation installations.
- cp437 (Code Page 437) – character set used in IBM PCs.
- cp850 (Code Page 850) – IBM / Microsoft Multilingual Character Set, used in IBM PCs.
- mac – Macintosh default character set used for many systems.
- roman8 – Hewlett-Packard character set.

Sort Orders

SQL Server release 11.0.x permits you to select a sort order (collating sequence) for your data according to your facility's needs. Your Sybase software includes sort order definition files (*.srt* files) under the *charsets* directory.

The available sort orders vary depending on the available character sets. To see what sort orders are available, refer to the *sort order=* field of the *SYBASE_SYSTEM:[SYBASE.CHARSETS.CHARSET_NAME]CHARSET.LOC* file for each character set you plan to install.

◆ ***WARNING!***

If possible, make all changes to the character set and sort order of SQL Server before creating any user databases or making any changes to the Sybase-supplied databases. Changing the character set and sort order after data or data structures have been added to SQL Server can cause difficulties. To change character set or sort order after you have added data, see the *System Administration Guide*.

- **Binary sort order**, which sorts all data according to ASCII numeric byte value, is the default sort order. Binary order sorts all

uppercase letters before lowercase letters. It is compatible with pre-10.0 SQL Server releases.

- **General purpose dictionary order** is case-sensitive. It sorts uppercase letters before their lowercase counterparts. Dictionary order recognizes the various accented forms of a letter and sorts them after the associated unaccented letter.
- **Dictionary order, case-insensitive**, sorts data in dictionary order but does not recognize case differences. Uppercase letters are equivalent to their lowercase counterparts and are intermingled in sorting results.
- **Dictionary order, case-insensitive order with preference**, does not recognize case difference in determining equivalency of items. A word written with uppercase letters is equivalent to the same word written with lowercase letters. The only time uppercase and lowercase letters are distinguished is when you use an **order by** clause. When an **order by** clause is used, SQL Server sorts uppercase letters before lowercase.

  Do not select "case-insensitive order with preference" unless your installation requires that uppercase letters be sorted before lowercase letters in otherwise equivalent strings for **order by** clauses. Using "case-insensitive with preference" could cause poor performance in large tables when the columns specified in an **order by** clause match the key of the table's clustered index.
- **Dictionary order, case- and accent-insensitive**, treats accented forms of a letter as equivalent to the associated unaccented letter. It intermingles accented and unaccented letters in sorting results.
- **Spanish dictionary order** sorts data according to Spanish lexical rules, and is case- and accent-sensitive.
- **Spanish case-insensitive dictionary order** sorts data according to Spanish lexical rules, but does not recognize case differences. It does recognize the various accented forms of a letter and sorts them after the associated unaccented letter.
- **Spanish case- and accent-insensitive dictionary order** sorts data according to Spanish lexical rules. This sort order does not recognize case differences and treats accented forms of a letter as equivalent to the associated unaccented letter. It intermingles uppercase and lowercase letters, and accented and unaccented letters, in sorting results.

By default, each of the Western European character sets included with SQL Server and Backup Server come with the following sort orders:

Table E-2: Available sort orders for Western European languages

Language	Available Sort Orders
All Western European languages	• Binary sort order
English, French, and German	• Dictionary order, case sensitive, accent sensitive • Dictionary order, case insensitive, accent sensitive • Dictionary order, case insensitive, accent insensitive • Dictionary order, case insensitive, accent sensitive, order with preference
Spanish	• Dictionary order, case sensitive, accent sensitive • Dictionary order, case insensitive, accent sensitive • Dictionary order, case insensitive, accent insensitive

Selecting the Sort Order

By default, your server will be configured with binary sort order. The server can support only one sort order at a time, so be sure to select a sort order that will work for all of your clients.

Message Language

By default, U.S. English messages are installed as the default language on SQL Server. If your clients require software messages in a language other than English, you must purchase and install the Language Module for those languages.

During the installation of SQL Server, you specify a default language for SQL Server. If messages are not supported in the client's language, they will receive their messages in the default language. For example, if your client's language is Portuguese, a language for which we currently do not provide translated messages, and the Spanish Language Module is installed and Spanish is specified as the

SQL Server default language, the client will receive their software messages in Spanish.

Index

D

O

P

Q

R

S

T

U

V

W