Utility Programs

Sybase® Adaptive Server® Enterprise

Version 12.0

Windows 95, Windows 98, and Windows NT

Document ID: 36249-01-1200-01

Last revised: October 1999

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

Contents

About This Book

Utility Programs for Windows 95, Windows 98 and Windows NT is a guide to the Sybase® Adaptive Server® Enterprise utility programs available for Windows 95, Windows 98 and Windows NT. Utility programs are commands that you invoke directly from the operating system.

Audience

This manual is for anyone using Transact-SQL® and Adaptive Server Enterprise version 12. It assumes that you have the basic knowledge to use the Windows 95, Windows 98 and Windows NT operating systems and Adaptive Server.

How to Use This Book

This manual includes the following chapters:

- Chapter 1, "Using the isql Utility" – discusses how to use the interactive SQL (**isql**) utility that allows access to SQL from your operating system.
- Chapter 2, "Using bcp to Transfer Data to and from Adaptive Server" – discusses, in detail, the Bulk Copy (**bcp**) utility which you use to move data between Adaptive Server and an operating system file.
- Chapter 3, "Using dsedit" – explains how to use the Windows NT Directory Services Editor (**dsedit**) utility to modify the interfaces (*sql.ini*) file.
- Chapter 4, "Utility Commands Reference" – lists and describes the utility commands that you use to manage and maintain your databases and Adaptive Server Enterprise.

The examples in this manual are based on the *pubs2* sample database. Ask your System Administrator how to access a clean copy of *pubs2*.

Related documents

The following documents comprise the Sybase Adaptive Server Enterprise documentation:

- The release bulletin for your platform – contains last-minute information that was too late to be included in the books.

  A more recent version of this release bulletin may be available on the World Wide Web. To check for critical product or document information added after the release of the product CD, use the Sybase Technical Library Product Manuals web site.

To access release bulletins at the Technical Library Product Manuals web site:

a Go to Product Manuals at http://sybooks.sybase.com.

b Select a product family link.

c Select a product link.

d From the Collection list in the left frame, select the "platform-specific" link for the product and version you are interested in.

e From the list of individual documents in the right frame, select the link to the release bulletin for your platform.

Browse the document online or download a PDF version by clicking the PDF button at the bottom of the left frame.

- The Adaptive Server installation guide for your platform – describes installation, upgrade, and configuration procedures for all Adaptive Server and related Sybase products.
- *What's New in Adaptive Server Enterprise?* – describes the new features in Adaptive Server version 12, the system changes added to support those features, and the changes that may affect your existing applications.
- *Transact-SQL User's Guide* – documents Transact-SQL, Sybase's enhanced version of the relational database language. This manual serves as a textbook for beginning users of the database management system. This manual also contains descriptions of the *pubs2* and *pubs3* sample databases.
- *System Administration Guide* – provides in-depth information about administering servers and databases. This manual includes instructions and guidelines for managing physical resources, security, user and system databases, and specifying character conversion, international language, and sort order settings.
- *Adaptive Server Reference Manual* – contains detailed information about all Transact-SQL commands, functions, procedures, and datatypes. This manual also contains a list of the Transact-SQL reserved words and definitions of system tables.
- *Performance and Tuning Guide* – explains how to tune Adaptive Server for maximum performance. This manual includes information about database design issues that affect performance, query optimization, how to tune Adaptive Server for very large databases, disk and cache issues, and the effects of locking and cursors on performance.

- The Utility Programs manual for your platform – documents the Adaptive Server utility programs, such as isql and bcp, which are executed at the operating system level.
- *Error Messages and Troubleshooting Guide* – explains how to resolve frequently occurring error messages and describes solutions to system problems frequently encountered by users.
- *Component Integration Services User's Guide* – explains how to use the Adaptive Server Component Integration Services feature to connect remote Sybase and non-Sybase databases.
- *Java in Adaptive Server Enterprise* – describes how to install and use Java classes as datatypes and user-defined functions in the Adaptive Server database.
- *Using Sybase Failover in a High Availability System* – provides instructions for using Sybase's Failover to configure an Adaptive Server as a companion server in a high availability system.
- *Using Adaptive Server Distributed Transaction Management Features* – explains how to configure, use, and troubleshoot Adaptive Server DTM features in distributed transaction processing environments.
- *XA Interface Integration Guide for CICS, Encina, and TUXEDO* – provides instructions for using Sybase's DTM XA Interface with X/Open XA transaction managers.
- *Adaptive Server Glossary* – defines technical terms used in the Adaptive Server documentation.

Other sources of information

Use the Sybase Technical Library CD and the Technical Library Product Manuals web site to learn more about your product:

- Technical Library CD contains product manuals and technical documents and is included with your software. The DynaText browser (included on the Technical Library CD) allows you to access technical information about your product in an easy-to-use format.

  Refer to the *Technical Library Installation Guide* in your documentation package for instructions on installing and starting Technical Library.
- Technical Library Product Manuals web site is an HTML version of the Technical Library CD that you can access using a standard web browser. In addition to product manuals, you'll find links to the Technical Documents web site (formerly known as Tech Info Library), the Solved Cases page, and Sybase/Powersoft newsgroups.

To access the Technical Library Product Manuals web site, go to Product Manuals at http://sybooks.sybase.com.

Sybase certifications on the web

Technical documentation at the Sybase web site is updated frequently.

- **For the latest information on product certifications and/or the EBF Rollups:**

  1 Point your web browser to Technical Documents at http://techinfo.sybase.com.

  2 In the Browse section, click on What's Hot.

  3 Select links to Certification Reports and EBF Rollups, as well as links to Technical Newsletters, online manuals, and so on.

- **If you are a registered SupportPlus user:**

  1 Point your web browser to Technical Documents at http://techinfo.sybase.com.

  2 In the Browse section, click on What's Hot.

  3 Click on EBF Rollups.

  You can research EBFs using Technical Documents, and you can download EBFs using Electronic Software Distribution (ESD).

  4 Follow the instructions associated with the SupportPlus^{SM} Online Services entries.

- **If you are not a registered SupportPlus user, and you want to become one:**

  You can register by following the instructions on the Web.

  To use SupportPlus, you need:

  1 A Web browser that supports the Secure Sockets Layer (SSL), such as Netscape Navigator 1.2 or later

  2 An active support license

  3 A named technical support contact

  4 Your user ID and password

- **Whether or not you are a registered SupportPlus user:**

  You may use Sybase's Technical Documents. Certification Reports are among the features documented at this site.

1 Point your web browser to Technical Documents at http://techinfo.sybase.com

2 In the Browse section, click on What's Hot.

3 Click on the topic that interests you.

Conventions

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

d:\sybase\bin

Table 1 details the typographic (font and syntax) conventions as used in this document.

Table 1: Font and syntax conventions for this document

Element	Example
Command names, command option names, utility names, utility flags, and other keywords are bold.	dsedit
Database names, datatypes, file names and path names are in *italics*.	*master* database *d:\sybase\bin* **Note** Be sure to substitute your Sybase installation drive and directory for *d:\sybase*.
Variables, or words that stand for values that you fill in, are in *italics*.	`select` *column_name* `from` *table_name* `where` *search_conditions*
Parentheses must be typed as part of the command.	`compute row_aggregate (column_name)`
Curly braces indicate that at least one of the enclosed options is required by the command (see comma). **Note** Do not type the curly braces.	`{cheese, sauce}`
Brackets mean that choosing one or more of the enclosed options is optional. **Note** Do not type the brackets.	`[anchovies, pineapple, bell_peppers]`
The *vertical bar* means you may select only one of the options shown.	`{cash | check | credit}`

Element	Example
The *comma* means you may choose as many of the options shown as you like; be sure to separate multiple choices in a command with commas.	`[extra_cheese, avocados, sour_cream]`
An *ellipsis* (...) means that you can *repeat* the unit that the ellipsis follows as many times as you like.	`buy thing = price [cash | check | credit] [, thing = price [cash | check | credit] ]...` – You must buy at least one *thing* (item) and give its price. – You may choose a method of payment: one of the options enclosed in square brackets. – You may choose also to buy additional items: as many of them as you like. For each item you buy, provide its name, its price, and (optionally) a method of payment.
Syntax statements, which display the utility's syntax including all its options, appear as shown here: Or, for a utility with more options: **Note** In syntax statements, utilities and their syntax display in lowercase normal font, although the various parts of the syntax can be either normal font for flags (-c) and options (cash) or italics for user-supplied values (*dsname*).	`dsedit -d`*dsname* `-l`*path* `bldmastr [-d physicalname] [-c cntrltype] [-s size] [-m]`
Examples that illustrate utility commands appear in bold, as shown:	bcp -v
Examples that illustrate computer output appear as shown:	[See example that follows this table.

```
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)
```

If You Need Help

Each Sybase installation that has purchased a support contract has one or more designated people who are authorized to contact Sybase Technical Support. If you cannot resolve a problem using the manuals or online help, please have the designated person contact Sybase Technical Support or the Sybase subsidiary in your area.

CHAPTER 1

Using the *isql* Utility

This chapter describes the interactive SQL utility, **isql**.

Topics covered are:

Note To use Transact-SQL directly from the operating system with the **isql** utility program, you must have an account, or login, on Adaptive Server.

For a detailed description of **isql** syntax, see **isql on page 110**.

Starting and Stopping the *isql* Utility

To start **isql**:

1 Enter the following command at the operating system prompt:

```
isql
```

The following prompt appears:

```
Password:
```

2 Enter your password.

The password does not appear on the screen as you type. The **isql** prompt appears:

```
1>
```

You can now issue Transact-SQL commands.

To exit **isql**:

Enter either of the following commands on a line by itself:

```
quit
exit
```

How to Use Transact-SQL in the *isql* Utility

isql sends Transact-SQL commands to Adaptive Server, formatting the results and printing them to standard output. There is no maximum size for an **isql** statement. For more information about using Transact-SQL, see the Transact-SQL User's Guide.

To terminate aTransact-SQL command, type the default command terminator "go" on a new line.

For example:

```
isql
Password:

1> use pubs2
2> go
1> select *
2> from authors
3> where city = "Oakland"
4> go
```

Formatting *isql* Output

Table 1-1 describes the options that change the format of **isql** output:

Table 1-1: Format options for isql

Option	Default	Meaning
-h *headers*	1	Number of rows to print between column headings
-s *colseparator*	Single space	Changes the column separator character
-w *columnwidth*	80 characters	Changes the line width

To include each command issued to **isql** in the output, use the **-e** option. To remove numbering and prompt symbols, use the **-n** option.

For example:

```
use pubs2
go
isql -e -n -o output
Password:

select *
from authors
where city = "Oakland"
go
```

```
 quit

type output

select *
from authors
where city = "Oakland"
au_id     au_lname               au_fname
phone        address
city          state country    postalcode
----------- ---------------------------------------- -
-----------
--------
------------ ----------------------------------------
-------------------- ----- ------------ ----------
213-46-8915 Green                    Marjorie
          415 986-7020 309 63rd St. #411
          Oakland   CA     USA    94618
274-80-9391 Straight                 Dick
          415 834-2919 5420 College Av.
          Oakland   CA     USA    94609
724-08-9931 Stringer                 Dirk
          415 843-2991 5420 Telegraph Av.
          Oakland   CA     USA    94609
724-80-9391 MacFeather               Stearns
          415 354-7128 44 Upland Hts.
          Oakland   CA     USA    94612
756-30-7391 Karsen                   Livia
          415 534-9219 5720 McAuley St.
          Oakland   CA     USA    94609
```

Note that the *output* file does not include the command terminator.

Correcting Input

If you make an error when typing a Transact-SQL command, you can do one of the following:

- Press Ctrl-c or type the word “reset” on a line by itself.

  This clears the query buffer and returns the **isql** prompt.

- Type the name of your text editor on a line by itself.

  This opens a text file where you can edit the query. When you write and save the file, you are returned to **isql** and the corrected query appears. Type “go” to execute it.

set Options That Affect Output

Table 1-2 lists the **set** options that affect Transact-SQL output. For more information, see **set** in the *Adaptive Server Reference Manual*.

Table 1-2: set options that affect Transact-SQL output

set Option	Default	Meaning
char_convert	Off	Turns character set conversion off and on between Adaptive Server and a client; also starts a conversion between the server character set and a different client character set.
fipsflagger	Off	Warns when any Transact-SQL extensions to entry level SQL92 are used.This option does not disable the SQL extensions. Processing completes when you issue the non-ANSI SQL command.
flushmessage	Off	Sends messages as they are generated.
language	us_english	Sets the language for system messages.
nocount	Off	Turns off report of number of rows affected.
noexec	Off	Compiles each query but does not execute it; often used with showplan.
parseonly	Off	Checks the syntax of queries and returns error messages without compiling or executing the queries.
showplan	Off	Generates a description of the processing plan for a query; does not print results when used inside a stored procedure or trigger.
statistics io statistics time	Off	Displays performance statistics after each execution.
statistics subquerycache	Off	Displays the number of cache hits, misses, and rows in the subquery cache for each subquery.
textsize	32K	Controls the number of bytes of text or image data returned.

Changing the Command Terminator

If you include the command terminator argument (**-c**), you can choose your own terminator symbol; **go** is the default value for this option. Always enter the command terminator without blanks or tabs in front of it.

For example, to use a period as the command terminator, invoke **isql** as follows:

```
isql -c.
```

A sample **isql** session with this command terminator looks like this:

```
1> select name from sysusers
2> .
name
-----------
sandy
kim
leslie
(3 rows affected)
```

Using the **isql** command terminator option with scripts requires advance planning:

- Adaptive Server-supplied scripts, such as **installmaster**, use “go”. Do not change the command terminator for any session that uses these scripts.
- Your own scripts may already have “go” in them. Remember to update your scripts to include the terminator you plan to use.

Performance Statistics Interaction with Command Terminator Values

isql provides a performance statistics option (**-p**).

For example:

```
isql -p
1> select * from sysobjects
2> go
```

returns the following statistics:

```
Execution Time (ms.): 1000   Clock Time (ms.): 1000
1 xact:
```

This means that a single transaction took 100 ms. The clock time value reflects the entire transaction, which starts when Client-Library™ builds the query and ends when Client-Library returns the information from Adaptive Server.

You can gather performance statistics based on the execution of one or more transactions. To gather statistics on more than one transaction, specify a number after the command terminator. For example, the following command:

```
isql -p

1> select * from sysobjects
2> go 3
```

instructs Adaptive Server to execute three **select *** transactions and report the performance statistics. Adaptive Server returns:

```
Execution Time (ms.): 1000   Clock Time (ms.): 1000
Execution Time (ms.): 1000   Clock Time (ms.): 2000
Execution Time (ms.): 1000   Clock Time (ms.): 1000

Execution Time (ms.): 1000   Clock Time (ms.): 4000
3xact:
```

Setting the Network Packet Size

Setting the correct network packet size can greatly increase the performance of Adaptive Server.

Setting the Network Packet Size for the Client

The **-A size** option specifies the network packet size to use for an **isql** session. For example:

```
load isql -A 2048
```

sets the packet size to 2048 bytes for this **isql** session. To check your network packet size, type:

```
select * from sysprocesses
```

The value for this **isql** session appears under the *network_pktsz* heading in the *sysprocesses* table.

See the *System Administration Guide* for more information about setting the network packet size.

Input and Output Files

You can specify input and output files on the command line with the **-i** and **-o** options.

isql does not provide formatting options for the output. However, you can use the **-n** option to eliminate the **isql** prompts and other tools to reformat the output.

If you use the -**e** option, **isql** echoes the input to output. The resulting output file contains both the queries and their results.

CHAPTER 2

Using *bcp* to Transfer Data to and from Adaptive Server

This chapter explains how to use the bulk copy utility, **bcp**, to move data between Adaptive Server and an operating system file. Topics covered are:

For a detailed description of the **bcp** syntax, see **bcp on page 80**.

Methods for Moving Data

You can use the following methods to move data to and from your Adaptive Server databases:

- Use **bcp** as a standalone program from the operating system. This chapter provides instructions for this method.
- Use Client-Library, which calls bulk library routines.

  For more information about the Client-Library, see the *Open Client and Open Server Common Libraries Reference Manual*.

Using *bcp* to Import and Export Data

Because Transact-SQL commands cannot transfer data in bulk, you must use **bcp** for any large transfers. You can use **bcp** to:

- Import data that was previously associated with another program, such as the records from another database management system. This is the most common use for **bcp**.

  Before using **bcp**, you must create a file of the records you want to import. The general steps are:

  a Put the data to transfer into an operating system file.

  b Run **bcp** from the operating system command line.
- Move tables between Adaptive Servers or between Adaptive Server and other data sources that can produce an operating system file.
- Copy out data from a view. For a description of the syntax for using **bcp** to copy out from a view, see **bcp on page 80**.

- **Note** You cannot use **bcp** to copy *in* data to a view.

- Transfer data for use with other programs, for example, with a spreadsheet program. The general steps to transfer data are:

  a Use **bcp** to move the data from Adaptive Server into an operating system file from which the other program imports the data.

  b When you finish using your data with the other program, copy it into an operating system file, and then use **bcp** to copy it into Adaptive Server.

Adaptive Server can accept data in any character or binary format, as long as the data file describes either the length of the fields or the **terminators**, the characters that separate columns.

The structures in the tables involved in the transfer need not be identical, because when **bcp**:

- Imports *from* a file, it appends data to an existing database table.
- Exports *to* a file, it overwrites the previous contents of the file.

When the transfer is complete, **bcp** reports the number of rows successfully copied and other performance information.

Requirements for Using *bcp*

Before using the **bcp** utility, you need to provide it with basic data information and prepare both the data for transfer and the command to access the data.

Basic Information Required

You must supply the following information to transfer data successfully to and from Adaptive Server:

- Name of the database and table or view
- Name of the operating system file
- Direction of the transfer (**in** or **out**)

In addition, you can use **bcp** to modify the storage type, storage length, and terminator for each column if you want to do so.

Permissions Needed to Copy Data

To use **bcp**, you must have an Adaptive Server account and the appropriate permissions on the database tables or views as well as the operating system files to use in the transfer.

- To copy data into a table, you must have **insert** permission on the table.
- To copy a table to an operating system file, you must have **select** permission on the following tables:
  - The table to copy
  - *sysobjects*
  - *syscolumns*
  - *sysindexes*

Pre-Transfer Task

Before you can use **bcp in**, you must prepare the command and the data for transfer:

- To use either fast or slow **bcp**, set **select into/bulkcopy/pllsort** to **true**. For example, to turn on this option for the *pubs2* database, you would enter:

```
sp_dboption pubs2, "select into/bulkcopy/pllsort", true
```

  For more information, see “bcp Modes” on page 16.

- To use fast **bcp**, remove indexes and triggers on the target table. For more information about this requirement, see “bcp Performance Issues” on page 17.

bcp Modes

bcp in works in one of two mode: fast or slow. Table 2-1 describes the differences between the two modes.

Table 2-1: Characteristics of fast and slow bcp

Type of *bcp*	Characteristics
Slow	Logs each row insert that it makes in a table. Used for tables that have one or more indexes or triggers.
Fast	Logs only page allocations. Copies data into tables without indexes or triggers at the fastest speed possible.

To determine the **bcp** mode that is best for your copying task, use the following information:

- The size of the table into which you are copying data,
- The amount of data that you are copying in,
- The number of indexes on the table, and
- The amount of spare database device space that you have for re-creating indexes.

Also, keep in mind that, although fast **bcp** might enhance performance, slow **bcp** gives you greater data recoverability.

bcp Performance Issues

Keeping indexes and triggers on a table causes the bulk copy utility to use slow **bcp** automatically. However, slow **bcp** can fill the transaction log very quickly.

- When you are copying a large number of rows, the performance penalty and log space requirements for using slow **bcp** can be severe.
- For extremely large tables, using slow **bcp** is not an option—its detailed log makes it much too slow.

To improve further the performance of **bcp**:

- Use partitioned tables. Several **bcp** sessions with a partitioned table can reduce dramatically the time required to copy the data. However, such performance improvements are more noticeable in fast **bcp** than in slow **bcp**.
- Use **bcp** in parallel to increase performance dramatically. Parallel bulk copy can provide balanced data distribution across partitions. For information about using parallel bulk copy, see "Using Parallel Bulk Copy to Copy Data Into a Specific Partition" on page 22.

Using Fast *bcp* or Slow *bcp*

The existence of indexes and triggers on tables does affect the speed of the transfer:

- When you copy in to a table that has indexes or triggers, **bcp** automatically uses its slow mode, which logs data inserts in the transaction log. These logged inserts can cause the transaction log to become very large.

  To control this data excess and ensure that the database is fully recoverable in the event of a failure, you can back up the log with **dump transaction**.

  Note **bcp** does not fire any trigger that exists on the target table.

- To save time, fast **bcp** does not log data inserts in the transaction log, but logs only the page allocations. For copying data in, **bcp** is fastest if your database table has no indexes or triggers.

However, if you used fast **bcp** to make data inserts, which fast **bcp** does not log, you cannot back up (**dump**) the transaction log to a device. The changes are not in the log, and a restore cannot recover non-existing backup data. The requested backup (**dump transaction**) produces an error message that instructs you to use **dump database** instead. This restriction remains in force until a **dump database** successfully completes.

Configuring the Database to Use Fast *bcp*

To allow a user to copy in data using fast **bcp**, either a System Administrator or the Database Owner first must use the **sp_dboption** system procedure to set **select into/bulkcopy/pllsort** to **true** on the database that contains the target table or tables. Should the option be **false** when a user tries to use fast **bcp** to copy data into a table that does not have indexes or triggers, Adaptive Server generates an error message.

Note You do not need to set the **select into/bulkcopy/pllsort** option to **true** to copy out data from or to copy in data to a table that has indexes or triggers. Slow **bcp** always copies tables with indexes or triggers and logs all inserts.

The following table shows which mode **bcp** automatically uses to copy in data and the possible settings for the **select into/bulkcopy/pllsort** option and indicates which combinations of mode and option allow the data transfer and backup.

Table 2-2: Fast and slow bcp and the select into/bulkcopy/pllsort option

select into/bulkcopy/pllsort	true / on	false / off
fast bcp on a target table that has no indexes or triggers	Transfer occurs. Cannot dump transaction to a device.	Cancels fast bcp automatically.
slow bcp on a target table that has one or more indexes or triggers	Transfer occurs. Allows dump transaction to occur.	Transfer occurs. Allows dump transaction to occur.

By default, the **select into/bulkcopy/pllsort** option is set to **false** (off) in newly created databases. To change the default setting for future databases, turn this option on (set to **true**) in the *model* database.

Dropping Indexes and Triggers

If you are copying a very large number of rows, you must have 1.2 times the amount of space needed for the data and enough space for the server to reconstruct a clustered index.

- If space is available, you can use **drop index** and **drop trigger** to drop all the indexes and triggers beforehand.

- If you don't have enough space for the server to sort the data and build the index or indexes, use slow **bcp.**

Steps for Copying In Data with Fast *bcp*

Table 2-3 summarizes the steps for copying in data to Adaptive Server using fast **bcp**.

Table 2-3: Steps for copying in data using fast bcp

Step	Who Can Do It
Use sp_dboption to set select into/bulkcopy/pllsort to true. Run checkpoint in the database that was changed.	System Administrator or Database Owner
Be sure that you have enough space to re-create any indexes and triggers on the table. Drop the indexes and triggers on the table.	Table owner
Be sure that you have insert permission on the table.	Granted by the table owner
Perform the copy with bcp.	Any user with *insert* permission
Re-create the indexes and triggers.	Table owner
Reset sp_dboption, if desired, and run checkpoint in the database that was changed.	System Administrator or Database Owner
Use dump database to back up the newly inserted data.	System Administrator, Operator, or Database Owner
Run stored procedures or queries to determine whether any of the newly loaded data violates rules.	Table owner or stored procedure owner

Bulk Copying Data into Partitioned Tables

In certain circumstances, you can improve **bcp** performance dramatically by executing several **bcp** sessions with a partitioned table.

Partitioned tables improve insert performance by reducing lock contention and by distributing I/O over multiple devices. **bcp** performance with partitioned tables is improved primarily because of this distributed I/O.

Guidelines

When you execute a **bcp** session on a partitioned table, consider the following:

- A partitioned table improves performance only when you are bulk copying *in* to the table.
- The performance of slow **bcp** does not improve as much with partitioned tables. Instead, drop all indexes and triggers and use fast **bcp**, as described in Table 2-3 on page 19, to increase performance.
- Network traffic can quickly become a bottleneck when multiple **bcp** sessions are being executed. If possible, use a local connection to the Adaptive Server to avoid this bottleneck.

Methods of Copying Data into a Partition

There are two methods for copying data into a partitioned heap table:

- Copying the data randomly without regard to the partition to which data is copied.
- Copying the data into a specific partition.

  If the table has a clustered index, **bcp** runs in slow mode and allows the index to control the placement of rows.

Copying Data Randomly into Partitions

To copy data randomly into partitioned tables when using multiple **bcp** sessions, you must perform the following actions:

1. Configure the table with as many partitions and physical devices as you require for your system.

   For more information, see the *Performance and Tuning Guide*, and "Using Parallel Bulk Copy to Copy Data Into a Specific Partition" on page 22 of this manual.
2. Make sure Adaptive Server is configured with enough locks to support multiple **bcp** sessions. For information on configuring locks, see the *System Administration Guide*.
3. Remove the triggers and indexes on the table and enable fast **bcp**.

To do so, follow the instructions in "Using Fast bcp or Slow bcp" on page 17.

Note If you use slow **bcp**, performance may not improve significantly after you remove the triggers and indexes. Also, if the table contains indexes, you may experience deadlocks on the index pages.

4 Divide the **bcp** input file into as many files of equal size as the number of planned simultaneous **bcp** sessions.

   You also can use the **-F***first_row* and **-L***last_row* options to specify the start and end of each "input file".

5 Execute the **bcp** sessions with separate files in parallel, preferrably on the local Adaptive Server machine.

   For example, on UNIX platforms, you can execute different sessions in different shell windows or start individual **bcp** sessions in the background.

For a detailed description of copying data into partitioned tables, see the *Performance and Tuning Guide.*

Monitor the *bcp* Session with *dbcc checktable* and *sp_helpsegment*

If you do not specify which partition the **bcp** sessions should use, Adaptive Server randomly assigns the multiple **bcp** sessions to the table's available partitions. If this random assignment occurs, be sure to monitor the partitions to ensure that the process has evenly distributed the inserts.

- Use the **dbcc checktable** command periodically to check the total page counts for each partition, or
- Use **sp_helpsegment** or **sp_helpartition**, which do not lock the database objects, to perform a similar check.

  For more information about **dbcc checktable**, see the *System Administration Guide*. For more information about **sp_helpsegment** and **sp_helpartition**, see the *Adaptive Server Reference Manual*.

  For more information about table partitions, see the *Performance and Tuning Guide*.

Reducing Logging by Increasing Page Allocations

If you are using fast **bcp**, consider that each **bcp in** batch requires the page manager to allocate one or more extents. Each such allocation generates a single log record.

Use the **number of preallocated extents** configuration parameter to specify how many extents Adaptive Server is to allocate through the page manager.

- Valid values for the **number of preallocated extents** configuration parameter are from 0 to 31; the default value is 2.
- You must restart Adaptive Server to change the value.
- When performing large **bcp** operations, increase this number to prevent the page allocations from filling the log.
- Set this value to 0 to prevent large extent allocations, so that the page manager performs only single page allocations.

Adaptive Server may allocate more pages than are actually needed, so keep the value small when space is limited. These pages are deallocated at the end of the batch.

For more information, see the *System Administration Guide*.

Using Parallel Bulk Copy to Copy Data Into a Specific Partition

Use parallel bulk copy to copy data in parallel to a specific partition. Parallel bulk copy substantially increases performance during **bcp** sessions because it can split large bulk copy jobs into multiple sessions and run the sessions concurrently.

To use parallel bulk copy:

- The destination table must be partitioned.
  - Use **sp_helpartition** to see the number of partitions on the table.
  - Use **alter table ... partition** to partition the table, if the table is not already partitioned.
- The destination table should not contain indexes because:
  - If the table has a clustered index, this index determines the physical placement of the data, causing the partition specification in the **bcp** command to be ignored.
  - If any indexes exist, **bcp** automatically uses its slow bulk copy instead of its fast bulk copy mode.
- If nonclustered indexes exist on the tables, parallel bulk copy is likely to lead to deadlocks on index pages.
- Each partition should reside on a separate physical disk for the best performance.

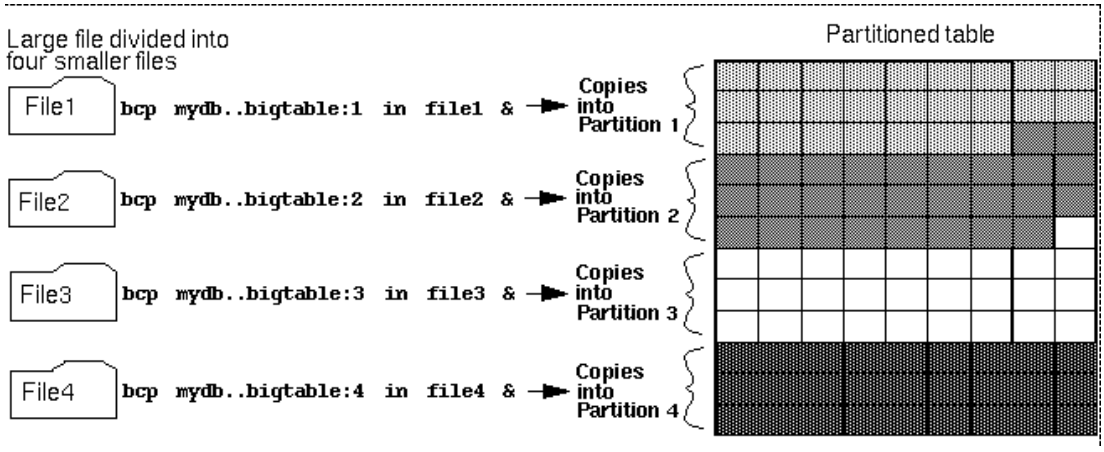
- Before you copy data into your database, you must partition the table destined to contain the data.
- Parallel bulk copy can copy in to a table from multiple operating system files. To do so, use the following syntax:

  ```
  bcp tablename :partition_number in file_name
  ```

  You can also use the **-F***first_row* and **-L***last_row* flags to designate the first and last row of the same host file for each multiple parallel bulk copy session.

Figure 2-1 illustrates the parallel bulk copy process:

Figure 2-1: Copying data into a partitioned table using parallel bulk copy



For information about partitioning a table, see the *Performance and Tuning Guide*.

Note When using parallel bulk copy to copy data out, you cannot specify which partitions **bcp** should use.

bcp in and Locks

When you copy in to a table using **bcp**, and particularly when you copy in to a table using parallel **bcp**, the copy process acquires the following locks:

- An exclusive intent lock on the table,
- An exclusive page lock on each data page, and

- An exclusive lock on index pages, if any indexes exist.

If you are copying in very large tables, and especially if you are using simultaneous copies into a partitioned table, this can involve a very large number of locks.

To avoid running out of locks:

- Increase the number of locks.
  - To estimate the number of locks needed, use the following formula:

```
# of simultaneous batches * (rows_per_batch / (2016/row_length))
```

  - To see the row length for a table, use the following query:

```
1> select maxlen
2> from sysindexes
3> where id = object_id("tablename") and (indid = 0 or indid = 1)
```

    See the *System Administration Guide* for more information about setting the number of locks.

- Use the **-b***batchsize* flag to copy smaller batches; the default batch size is 1000 rows.
- Run fewer batches concurrently.

Parallel Bulk Copy Methods

Use one of the following methods to copy in data using parallel bulk copy:

- Start multiple **bcp** sessions in the background, being sure to:
  - Specify the password at the command line.
  - Use native mode, character mode, or a format file.

  You can start **bcp** as many times as the table is partitioned.
- Create and use a format file:
  a. Start **bcp** in interactive mode.
  b. Answer the prompts.
  c. Create a format file that stores your responses.
  d. Put the process in the background when the copy begins.
  e. Issue the next **bcp** command, and specify the format file created with the first **bcp** command.

- Start **bcp** sessions in multiple windows.

Parallel Bulk Copy Syntax

The syntax for parallel bulk copy is:

```
bcp table_name[:partition_number] in file_name -Pmypassword
```

where:

- *table_name* is the name of the table into which you are copying the data,
- *partition_number* is the number of the partition into which you are copying,
- *file_name* is the host file that contains the data, and
- *mypassword* is your password.

Using Parallel Bulk Copy on Partitioned Tables

To copy sorted data in parallel into a specific partition:

- Specify the partition by appending a colon (:) plus the partition number to the table name. For example:

  ```
  publishers:10
  ```

  Note The partition you specify must exist before you issue the **bcp** command.

- Split the sorted data into separate files, or delineate the "files" by specifying the first row (**-F***first_row*) and the last row (**-L***last_row*) of the host file.
- Note the number of partitions in the table. This number limits the number of parallel bulk copy sessions that you can start.

  For example, if a table has four partitions, and you start five parallel bulk copy jobs, only the first four jobs can run in parallel; the fifth job does not start until one of the first four jobs finish.

bcp copies each file or set of line numbers to a separate partition. For example, to use parallel bulk copy to copy in sorted data to *mydb..bigtable* from four files into four partitions, enter:

```
bcp mydb..bigtable:1 in file1 -Pmypassword -c &
bcp mydb..bigtable:2 in file2 -Pmypassword -c &
```

```
bcp mydb..bigtable:3 in file3 -Pmypassword -c &
bcp mydb..bigtable:4 in file4 -Pmypassword -c &
```

Parallel Bulk Copy and IDENTITY Columns

When you are using parallel bulk copy, IDENTITY columns can cause a bottleneck. As **bcp** reads in the data, the utility both generates the values of the IDENTITY column and updates the IDENTITY column's maximum value for each row. This extra work may adversely affect the performance improvement that you expected to receive from using parallel bulk copy.

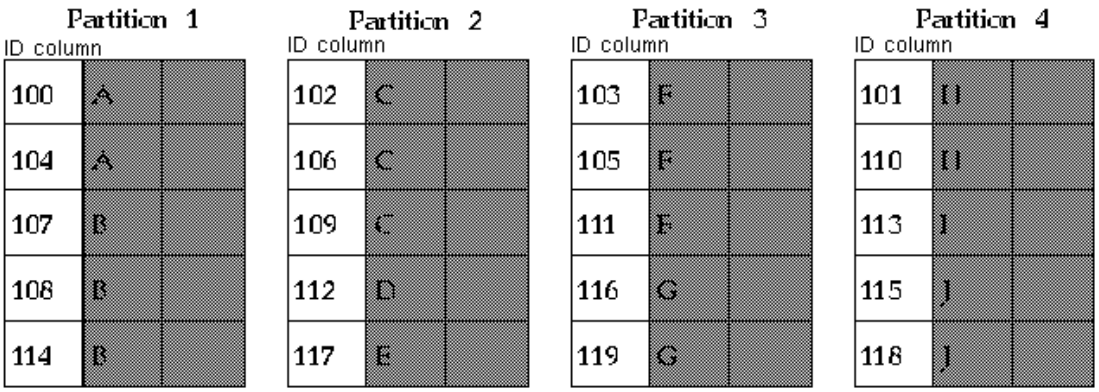
To avoid this bottleneck, you can explicitly specify the IDENTITY starting point for each session.

Retaining Sort Order

If you copy sorted data into the table without explicitly specifying the IDENTITY starting point, **bcp** might not generate the IDENTITY column values in sorted order. Parallel bulk copy reads the information into all the partitions simultaneously and updates the values of the IDENTITY column as it reads in the data.

A **bcp** statement with no explicit starting point would produce IDENTITY column numbers similar to those shown in Figure 2-2:

Figure 2-2: Producing IDENTITY columns in sorted order



The table has a maximum IDENTITY column number of 119, but the order is no longer meaningful.

If you want Adaptive Server to enforce unique IDENTITY column values, you must run **bcp** with either the **-g** or **-E** parameter.

Specifying the Starting Point from the Command Line

Use the **-g***id_start_value* flag to specify an IDENTITY starting point for a session in the command line.

The **-g** parameter instructs Adaptive Server to generate a sequence of IDENTITY column values for the **bcp** session without checking and updating the maximum value of the table's IDENTITY column for each row. Instead of checking, Adaptive Server updates the maximum value at the end of each batch.

Warning! You can create duplicate identity values inadvertently when you specify identity value ranges that overlap. For more information, see the text that follows.

To specify a starting IDENTITY value, enter:

```
bcp [-gid_start_value]
```

For example, to copy in four files, each of which has 100 rows, enter:

```
bcp mydb..bigtable in file1 -g100
bcp mydb..bigtable in file2 -g200
bcp mydb..bigtable in file3 -g300
bcp mydb..bigtable in file4 -g400
```

However, using the **-g** parameter does not guarantee that the IDENTITY column values are unique. To ensure uniqueness, you must:

- Know how many rows are in the input files and what the highest existing value is. Use this information to set the starting values with the **-g** parameter and generate ranges that do not overlap.

  In the example above, if any file contains more than 100 rows, the identity values overlap into the next 100 rows of data, creating duplicate identity values.

- Make sure that no one else is inserting data that can produce conflicting IDENTITY values.

Specifying the Starting Point Using the Data File

Use the **-E** parameter to set the IDENTITY starting point explicitly from the data file.

The **-E** parameter instructs **bcp** to prompt you to enter an explicit IDENTITY column value for each row. If the number of inserted rows exceeds the maximum possible IDENTITY column value, Adaptive Server returns an error.

Using the *bcp* Options

For the **bcp** syntax and a full discussion of the options available with this utility, see **bcp on page 80**. Use the information in this section to clarify some of the more complex options.

Using the Default Formats

bcp provides two command line options that create files with frequently used default formats. These options provide the easiest way to copy in and out data from Adaptive Server.

- The **-n** option uses "native" (operating system) formats.
- The -**c** option uses "character" (char datatype) for all columns. This datatype supplies tabs between fields on a row and a newline terminator, such as a carriage return, at the end of each row.

When you use the native or character options, **bcp** operates non-interactively and only asks you for your Adaptive Server password.

Native Format

The **-n** option creates files using **native** (operating system-specific) formats. Native formats usually create a more compact operating system file. For example, the following command copies the *publishers* table to the file called *pub_out*, using native data format:

```
bcp pubs2..publishers out pub_out -n
```

Here are the contents of *pub_out*:

```
0736^MNew Age Books^FBoston^BMA0877^PBinnet  & Hardley^J
Washington^BDC1389^TAlgodata Infosystems^HBerkeley^BCA
```

bcp prefixed each field, except the *pub_id*, which is a char(4) datatype, with an ASCII character equivalent to the length of the data in the field. For example, "New Age Books" is 13 characters long, and ^M (Ctrl-M) is ASCII 13.

All the table data stored in the *pub_out* file is char or varchar data, so it is human-readable. In a table with numeric data, **bcp** writes the information to the file in the operating system's data representation format, which may not be human-readable.

Note The **bcp** utility does not support copying data in native format from different operating systems; for example, copying from NT to UNIX. Use the **-c** flag if you need to use **bcp** to copy files from one operating system to another.

Warning! Do not use row terminator (**-t)** or field terminator (**-r)** parameters with **bcp** in native format. Results are unpredictable and data could become corrupted.

Character Format

Character format (**-c**) uses the char datatype for all columns. It inserts tabs between fields in each row and a newline terminator at the end of each row.

For example, the following command copies out the data from the *publishers* table in character format to the file *pub_out*:

```
bcp pubs2..publishers out pub_out -c
```

The preceding command produces the following **bcp** output:

```
0736   New Age Books        Boston      MA
0877   Binnet  & Hardley     Washington  DC
1389   Algodata Infosystems Berkeley    CA
```

Changing Terminators From the Command Line

Terminators are the characters that separate data fields (field terminators). The row terminator is the field terminator of the last field in the table or file. Use the **-t***field_terminator* and **-r***row_terminator* command line options with the character format option (**-c**) to change the terminators from the command line.

The following example uses the comma (,) as the field terminator and Enter (\r) as the row terminator:

```
bcp pubs2..publishers out pub_out -c -t , -r \r
```

This **bcp** command line produces the following information:

```
0736,New Age Books,Boston,MA
0877,Binnet  & Hardley,Washington,DC
1389,Algodata Infosystems,Berkeley,CA
```

Note You can use the **-t** and **-r** options to change the default terminators without including the character option (**-c**).

Changing the Defaults: Interactive *bcp*

If you do not specify native (**-n**) or character (**-c**) format, **bcp** prompts you interactively for:

- The file storage type,
- The prefix length,
- The terminator for each column of data to be copied, and
- A field length for fields that are to be stored as char or binary.

The default values for these prompts produce the same results as using the native format and provide a simple means for copying data out of a database for later reloading into Adaptive Server.

If you are copying data to or from Adaptive Server for use with other programs, base your answers to the prompts on the format required by the other software.

These four prompts provide an extremely flexible system that allows you either to read a file from other software or to create a file that requires little or no editing to conform to many other data formats.

The following sections discuss these prompts and the way they interact to affect the data.

File Storage Type

The file storage type prompt offers you choices about how to store the data in the file. You can copy data into a file as:

- Its database table type,
- A character string, or
- Any datatype for which implicit conversion is supported.

Note **bcp** copies user-defined datatypes as their base types.

Table 2-4 shows the default storage type for each Adaptive Server datatype and the abbreviations that are acceptable to **bcp**.

- For the most compact storage, use the default value.
- For character files, use char.

- Keep in mind that the date storage type is the Adaptive Server internal storage format of datetime, not the host operating system format of the date.
- timestamp data is treated as binary(8).

In Table 2-4, brackets [] indicate that you can use the initial character or the beginning characters of the word. For example, for "bit" you can use "b," "bi," or "bit."

Table 2-4: File storage datatypes for bcp

Table Datatype	Storage Type
char, varchar	c[har]
text	T[ext]
int	i[nt]
smallint	s[mallint]
tinyint	t[inyint]
float	f[loat]
money	m[oney]
bit	b[it]
datetime	d[atetime]
binary, varbinary, timestamp	x
image	I[mage]
smalldatetime	D
real	r
smallmoney	M
numeric	n
decimal	e

To display this list while using **bcp** interactively, type a question mark (?) in response to the prompt "Enter the file storage type".

The suggested values that appear in the prompts are the defaults. Remember that your response determines how the data is stored in the output file; you need not indicate the column's type in the database table.

bcp fails if you enter a type that is not either implicitly convertible or char. For example, you may not be able to use smallint for int data (you may get overflow errors), but you can use int for smallint.

When storing noncharacter datatypes as their database types, **bcp** writes the data to the file in Adaptive Server's internal data representation format for the host operating system, rather than in human-readable form.

Prefix Length

By default, **bcp** precedes each field that has a variable storage length with a string of one or more bytes indicating the length of the field. This prefix enables the most compact file storage.

The default values in the prompts indicate the most efficient prefix length:

- For fixed-length fields, the prefix length should be 0.
- For fields of 255 bytes or less, the default prefix length is 1.
- For text or image datatypes, the default prefix length is 4.
- For binary and varbinary datatypes that are being converted to char storage types, the default prefix length is 2, since each byte of table data requires 2 bytes of file storage.
- For binary, varbinary, and image data, use even numbers for the prefix and length. This requirement maintains consistency with Adaptive Server, which stores data as an even number of hexadecimal digits.
- For any data column that permits null values, use a prefix length, other than 0, or a terminator to denote the length of each row's data. **bcp** considers such columns, including columns with integer datatypes that might ordinarily be considered fixed-length columns, to be of variable length.
- For data with no prefix before its column, use a prefix length of 0.

Unless you supply a terminator, **bcp** pads each stored field with spaces to the full length specified at the next prompt, "length".

Because prefix lengths consist of *native* format integers, the resulting host file contains non-printable characters. The nature of these characters could prevent you from printing the host file or from transmitting it through a communications program that cannot handle non-human-readable characters.

For more information about prefix lengths, see Table 2-7 on page 40.

Field Length

In almost all cases, use the **bcp** default value for the storage length while copying data out.

Note The terms “length” and “storage length” in this section refer to the operating system file, not to Adaptive Server field lengths.

- If you are creating a file to reload into Adaptive Server, the default prefixes and length keep the storage space needed to a minimum.
- If you are creating a human-readable file, the default length prevents the truncation of data or the creation of overflow errors that cause **bcp** to fail.

Because it is possible to change the default length by supplying another value, you must be familiar with the data to transfer. If you are copying character data in from other software, carefully examine the source file before choosing length values.

Note If the storage type is non-character, **bcp** stores the data in the operating system’s native data representation and does not prompt for a length.

When **bcp** converts non-character data to character storage, it suggests a default field length that is large enough to store the data without truncating datetime data or causing an overflow of numeric data.

- The default lengths are the number of bytes needed to display the longest value for the Adaptive Server datatype.

  Table 2-5 lists the default field lengths for data conversion to character storage:

Table 2-5: Default field lengths for non-character to character datatypes

Datatype	Default Size
int	12 bytes
smallint	6 bytes
tinyint	3 bytes
float	25 bytes
money	24 bytes
bit	1 byte
datetime	26 bytes
smalldatetime	26 bytes

Datatype	Default Size
real	25 bytes
smallmoney	24 bytes

- If you specify a field length that is too short for numeric data when copying data out, **bcp** prints an overflow message and does not copy the data.
- The default length for binary and varbinary fields is twice the length defined for the column, since each byte of the field requires 2 bytes of file storage.
- If you accept the default storage length, the actual amount of storage space allocated depends on whether or not you specify a prefix length and terminators.
    - If you specify a prefix length of 1, 2, or 4, **bcp** uses a storage space of the actual length of the data, plus the length of the prefix, plus any terminators.
    - If you specify a prefix length of 0 and no terminator, **bcp** allocates the maximum amount of space shown in the prompt, which is the maximum space that may be needed for the datatype in question. In other words, **bcp** treats the field as if it were fixed length to determine where one field ends and the next begins.

      For example, if the field is defined as varchar(30), **bcp** uses 30 bytes for each value, even if some of the values are only 1 character long.
- **bcp** does not know how large any one data value will be before copying all the data, so it always pads char datatypes to their full specified length.

Field and Row Terminators

A terminator can be used to mark the end of a column or ,row separating one from the next. The default is no terminator.

- Field terminators separate table columns.
- A row terminator is a field terminator for the last field in the row of the table or file.

Terminators are very useful for dealing with character data because you can choose human-readable terminators. The **bcp** character option, which uses tabs between each column with a newline terminator at the end of each row, is an example of using terminators that enhance the readability of a data file.

When you prepare data for use with other programs, and when you want to use **bcp** to prepare tabular data, supply your own terminators. The available terminators are:

- Tabs, indicated by `\t`
- New lines, indicated by `\n`
- Carriage returns, indicated by `\r`
- Backslash, indicated by `\`
- Null terminators (no visible terminator), indicated by `\0`
- Any printable character, for example, *, A, t, |
- Strings of up to 10 printable characters, including some or all of the terminators listed above (for example, **\t**, end, !!!!!!!!!!, and \t--\n)

Note Control characters (ASCII 0–25) cannot be printed.

Choosing Terminators

Be sure to choose terminators with patterns that do not appear in any of the data.

For example, using a tab terminator with a string of data that also contains a tab creates an ambiguity: Which tab represents the end of the string? **bcp** always looks for the first possible terminator, which in this case would be incorrect, since the first tab it would encounter would be the one that is part of the data string.

Data in native format can also conflict with terminators. Given a column that contains a 4-byte integer in native format, if the values of these integers are not strictly limited, it will be impossible to choose a terminator that is guaranteed not to appear inside the data. Use **bcp**'s native format option for data in native format.

Note The "no terminator" is different from a "null terminator," which is an invisible, but real, character.

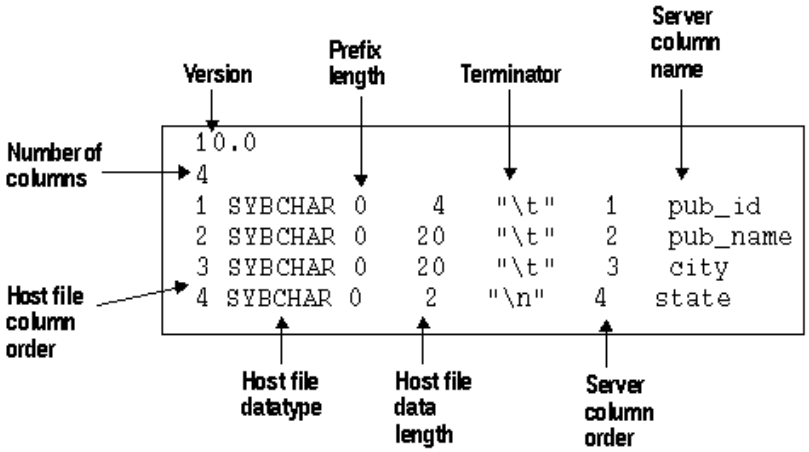
Using Format Files

After gathering information about each field in the table, **bcp** asks if you want to save the information to a **format file** and prompts for the file name.

Using a format file created for the data to be copied with the **bcp** utility allows you to copy data in or out non-interactively; that is, without being prompted by **bcp** for information. The format file supplies the information that **bcp** needs. You can use this newly created format file at any other time to copy the data back into Adaptive Server or to copy data out from the table.

Figure 2-3 illustrates the format of the **bcp** format files. It shows the *publishers* table from the *pubs2* database, with all the host file columns in character format, with no prefix, and using the default data length, a newline terminator at the end of the final column of a row, and tabs as terminators for all other columns.

Figure 2-3: bcp format file



Elements of the *bcp* Format File

The following list names the various elements of a **bcp** format file. Use Figure 2-3 on page 38 as the format file example.

- The **bcp** version is always the first line of the file. It specifies the version of **bcp** that you are using, not the Adaptive Server version, and appears is a literal string without quotation marks. In Figure 2-3, the version is 10.0.

- The second line of a **bcp** format file is the *number of columns*, which refers to the number of records in the format file, not including lines 1 and 2. Each column in the host table has one line.
- One line for each column follows the first and second lines in the database table. Each line consists of elements that are usually separated by tabs, except for the host file datatype and the prefix length which are usually separated by a space. These elements are:
  - Host file column order
  - Host file datatype
  - Prefix length
  - Host file data length
  - Terminator
  - Server column order
  - Server column name
  - Column precision
  - Column scale

The following sections describe the column elements in the format file.

Host File Column Order

The *host file column order* is the sequential number of the field in the host data file, which begins numbering at 1.

Host File Datatype

The *host file datatype* refers to the storage format of the field in the host data file, *not* the datatype of the database table column.

Table 2-6 lists the valid storage formats.

Table 2-6: Host file datatype storage format

Storage Format	Adaptive Server Datatype
SYBCHAR	*char/varchar* (ASCII)
SYBTEXT	text
SYBBINARY	binary
SYBIMAGE	image
SYBINT1	tinyint

Storage Format	Adaptive Server Datatype
SYBINT2	smallint
SYBINT4	int
SYBFLT8	float
SYBREAL	real
SYBBIT	bit
SYBNUMERIC	numeric
SYBDECIMAL	decimal
SYBMONEY	money
SYBMONEY4	smallmoney
SYBDATETIME	datetime
SYBDATETIME4	smalldatetime

Prefix Length

Prefix length indicates the number of bytes in the field length prefix. The length prefix is a 0-, 1-, 2-, or 4-byte unsigned integer value embedded in the host data file that specifies the actual length of data contained in the field. Some fields may have a length prefix while others do not.

Table 2-7 shows the allowable prefix length values.

Table 2-7: Allowable prefix length values

Length (in bytes)	Range
0	No prefix
1	2^8-1; 0–255
2	2^{16}-1; 0–65535
4	2^{32} -1; 0–4,294,967,295

Host File Data Length

Host file data length refers to the maximum number of bytes to copy for the field. To decide how much data to copy in or out, **bcp** uses either:

- The maximum field length,
- The prefix length, if any, or
- The field terminator string, if any.

If more than one method of field length specification is given, **bcp** chooses the one that copies the least amount of data.

Terminator

The *terminator* can be up to 30 bytes of characters enclosed in quotation marks (" "). The terminator designates the end of data for the host data file field.

Server Column Order

The *server column order* represents the *colid* (column ID) of the *syscolumns* column into which the host data file column is to be loaded. Together with the *host file column order*, this element maps host data file fields to the database table columns.

Server Column Name

The *server column name* is the name of the database table column into which this field is to be loaded.

Column Precision

The *column precision* is the precision of the database table column into which this field is to be loaded. This element is present only if the storage format is numeric or decimal.

Column Scale

The *column scale* is the scale of the database table column into which this field is to be loaded. This element is present only if the storage format is numeric or decimal.

Examples: Copying Out Data Interactively

By changing the default values of the prompts to **bcp**, you can prepare data for use with other software.

To create a human-readable file, respond to the **bcp** prompts as follows:

- File storage type, enter `0`
- Prefix length, enter `0`
- Field length, accept the default.
- Terminator – the field terminator you enter depends on the software that you plan to use.
  - Choose between delimited fields or fixed-length fields. Always use `\n`, the newline terminator, to terminate the last field.

    For fixed-length fields, do not use a terminator. Each field has a fixed length, with spaces to pad the fields. Adjacent fields, where the data completely fills the first field seem to run together, since there are no field separators on each line of output. See the example below.
  - For comma-delimited output, use a comma (`,`) as the terminator for each field. To create tabular output, use the tab character (`\t`).

Copying Out Data with Field Lengths

The following example uses fixed-length fields to create output in the personal computer format called SDF (system data format). This format can be easily read or produced by other software.

Note For information about format files, see "Using Format Files" on page 38.

```
bcp pubs2..sales out sal_out
```

The results as stored in the *sal_out* file are as follows:

```
5023   AB-123-DEF-425-1Z3        Oct 31 1985 12:00AM
5023   AB-872-DEF-732-2Z1        Nov 6 1985 12:00AM
5023   AX-532-FED-452-2Z7        Dec 1 1990 12:00AM
5023   BS-345-DSE-860-1F2        Dec 12 1986 12:00AM
5023   GH-542-NAD-713-9F9        Mar 15 1987 12:00AM
5023   NF-123-ADS-642-9G3        Jul 18 1987 12:00AM
5023   XS-135-DER-432-8J2        Mar 21 1991 12:00AM
```

```
5023   ZA-000-ASD-324-4D1  J      ul 27 1988 12:00AM
5023   ZD-123-DFG-752-9G8         Mar 21 1991 12:00AM
5023   ZS-645-CAT-415-1B2         Mar 21 1991 12:00AM
5023   ZZ-999-ZZZ-999-0A0         Mar 21 1991 12:00AM
6380   234518                     Sep 30 1987 12:00AM
6380   342157                     Dec 13 1985 12:00AM
6380   356921                     Feb 17 1991 12:00AM
7066   BA27618                    Oct 12 1985 12:00AM
7066   BA52498                    Oct 27 1987 12:00AM
7066   BA71224                    Aug 5 1988 12:00AM
7067   NB-1.142                   Jan 2 1987 12:00AM
7067   NB-3.142                   Jun 13 1990 12:00AM
7131   Asoap132                   Nov 16 1986 12:00AM
7131   Asoap432                   Dec 20 1990 12:00AM
7131   Fsoap867                   Sep 8 1987 12:00AM
7896   124152                     Aug 14 1986 12:00AM
7896   234518                     Feb 14 1991 12:00AM
8042   12-F-9                     Jul 13 1986 12:00AM
8042   13-E-7                     May 23 1989 12:00AM
8042   13-J-9                     Jan 13 1988 12:00AM
8042   55-V-7                     Mar 20 1991 12:00AM
8042   91-A-7                     Mar 20 1991 12:00AM
8042   91-V-7                     Mar 20 1991 12:00AM
```

The contents of the *sal_fmt* format file are as follows:

```
11.1
3
1  SYBCHAR 04  "" 1  stor_id
2  SYBCHAR 020 "" 2  ord_num
3  SYBCHAR 026 "" 3  date
```

For information about format files, see “Using Format Files” on page 38.

Copying Out Data with Delimiters

In the following examples, **bcp** copies data interactively from the *publishers* table to a file.

Note For information about format files, see “Using Format Files” on page 38.

Comma-delimited, Newline-delimited with Format File

The first example creates an output file with commas between all fields in a row and a newline terminator at the end of each row. This example creates a format file (*pub_fmt*) that you can use later to copy the same or similar data back into Adaptive Server.

```
bcp pubs2..publishers out pub_out
```

The results as stored in the *pub_out* file are as follows:

```
0736,New Age Books,Boston,MA
0877,Binnet  & Hardley,Washington,DC
1389,Algodata Infosystems,Berkeley,CA
```

The contents of the *pub_fmt* format file are as follows:

```
11.1
4
1  SYBCHAR 0 4  ","  1  pub_id
2  SYBCHAR 0 40 ","  2  pub_name
3  SYBCHAR 0 20 ","  3  city
4  SYBCHAR 0 2  "\n" 4  state
```

Tab-delimited with Format File

Similarly, the following example creates tab-delimited output from the table *pubs2..publishers* in the *pub_out* file.

```
bcp pubs2..publishers out pub_out
```

The results as stored in the *pub_out* file are as follows:

```
0736  New Age Books        Boston     MA
0877  Binnet  & Hardley     Washington DC
1389  Algodata Infosystems Berkeley   CA
```

The contents of the *pub_fmt* format file are as follows:

```
11.1
4
1  SYBCHAR 04   "\t"  1 pub_id
2  SYBCHAR 040  "\t"  2 pub_name
3  SYBCHAR 020  "\t"  3 city
4  SYBCHAR 02   "\n"  4 state
```

Examples: Copying In Data Interactively

To copy in data successfully to a table from a file, you must know what the terminators in the file are or what the field lengths are and specify them when you use **bcp**.

The following examples show how to copy data in, either with fixed field lengths or with delimiters, using **bcp** with or without a format file.

Copying In Data with Field Lengths

In this example, **bcp** copies data from the *salesnew* file into the *pubs2..sales* table.

In the *salesnew* file are three fields: the first is 4 characters long, the second is 20, and the third is 26 characters long. Each row ends with a newline terminator (\n), as follows:

```
5023ZS-731-AAB-780-2B9May 24 1993 12:00:00:000AM
5023XC-362-CFB-387-3Z5May 24 1993 12:00:00:000AM
6380837206       May 24 1993 12:00:00:000AM
6380838441       May 24 1993 12:00:00:000AM
```

Use the following command to copy in the data interactively from *salesnew*:

```
bcp pubs2..sales in salesnew
```

The system responds to the **bcp** command as follows:

```
Password:
Enter the file storage type of field stor_id [char]:
Enter prefix-length of field stor_id [0]:
Enter length of field stor_id [4]:
Enter field terminator [none]:
Enter the file storage type of field ord_num [char]:
Enter prefix-length of field ord_num [1]: 0
Enter length of field ord_num [20]:
Enter field terminator [none]:
Enter the file storage type of field date [datetime]: char
Enter prefix-length of field date [1]: 0
Enter length of field date [26]:
Enter field terminator [none]: \n
Do you want to save this format information in a file? [Y/n] y
Host filename [bcp.fmt]: salesin_fmt
Starting copy...
4 rows copied.
Clock Time (ms.): total = 1 Avg = 0 (116000.00 rows per sec.)
```

When you log in to Adaptive Server and access *sales*, you see the following data from *salesnew* appended to the table:

```
select * from sales
stor_idord_num            date
 ------- ------------------- ------------------------
--
 5023  AB-123-DEF-425-1Z3     Oct 31 1985 12:00AM
 5023  AB-872-DEF-732-2Z1     Nov  6 1985 12:00AM
 5023  AX-532-FED-452-2Z7     Dec  1 1990 12:00AM
 5023  BS-345-DSE-860-1F2     Dec 12 1986 12:00AM
 5023  GH-542-NAD-713-9F9     Mar 15 1987 12:00AM
 5023  NF-123-ADS-642-9G3     Jul 18 1987 12:00AM
 5023  XS-135-DER-432-8J2     Mar 21 1991 12:00AM
 5023  ZA-000-ASD-324-4D1     Jul 27 1988 12:00AM
 5023  ZD-123-DFG-752-9G8     Mar 21 1991 12:00AM
 5023  ZS-645-CAT-415-1B2     Mar 21 1991 12:00AM
 5023  ZZ-999-ZZZ-999-0A0     Mar 21 1991 12:00AM
 6380  234518                 Sep 30 1987 12:00AM
 6380  342157                 Dec 13 1985 12:00AM
 6380  356921                 Feb 17 1991 12:00AM
 7066  BA27618                Oct 12 1985 12:00AM
 7066  BA52498                Oct 27 1987 12:00AM
 7066  BA71224                Aug  5 1988 12:00AM
 7067  NB-1.142               Jan  2 1987 12:00AM
 7067  NB-3.142               Jun 13 1990 12:00AM
 7131  Asoap132               Nov 16 1986 12:00AM
 7131  Asoap432               Dec 20 1990 12:00AM
 7131  Fsoap867               Sep  8 1987 12:00AM
 7896  124152                 Aug 14 1986 12:00AM
 7896  234518                 Feb 14 1991 12:00AM
 8042  12-F-9                 Jul 13 1986 12:00AM
 8042  13-E-7                 May 23 1989 12:00AM
 8042  13-J-9                 Jan 13 1988 12:00AM
 8042  55-V-7                 Mar 20 1991 12:00AM
 8042  91-A-7                 Mar 20 1991 12:00AM
 8042  91-V-7                 Mar 20 1991 12:00AM
 (34 rows affected)
```

Since there is a unique clustered index on the *stor_id* and *ord_num* columns of *sales*, the new rows were sorted in order.

A conflict or violation can affect the copy process:

- Had there been any violations of the unique index on the columns in the data being copied from the file, **bcp** would have discarded the entire batch in which the violating row was encountered.

A batch size of 1 evaluates each row individually, but loads more slowly and creates a separate data page for each row during a fast **bcp** session.

- If the types copied in are incompatible with the database types, the entire copy fails.

Copying In Data with Delimiters

In the following example, **bcp** copies data from the file *newpubs* into the table *pubs2..publishers*. In the *newpubs* file, each field in a row ends with a tab character (\t) and each row ends with a newline terminator (\n), as follows:

```
1111 Stone Age Books   Boston    MA
2222Harley  & Davidson WashingtonDC
3333 Infodata Algosystems Berkeley  CA
```

Since *newpubs* contains all character data, you can use the character command line flag and specify the terminators with command line options as follows:

```
bcp pubs2..publishers in newpubs -c -t\t -r\n
```

Copying In Data with a Format File

To copy data back into Adaptive Server using the saved *pub_fmt* format file, run the following command:

```
bcp pubs2..publishers in pub_out -fpub_fmt
```

You can use the *pub_fmt* file to copy any data with the same format into Adaptive Server. If you have a similar data file with different delimiters, you can change the delimiters in the format file.

Similarly, you can edit the format file to reflect any changes to the field lengths, as long as all fields have the same length. For example, the *moresales* file contains the following:

```
804213-L-9 Jan 21 1993 12:00AM
804255-N-8 Mar 12 1993 12:00AM
804291-T-4 Mar 23 1993 12:00AM
804291-W-9 Mar 23 1993 12:00AM
```

Edit the *sal_fmt* format file to read as follows:

```
11.5
3
1   SYBCHAR 0 4 "" 1 stor_id
```

```
   2  SYBCHAR 0 7 "" 2 ord_num
   3  SYBCHAR 0 21 "\n" 3 date
```

Then enter the following command:

```
bcp pubs2..sales in moresale -fsal_fmt
```

The system responds as follows:

```
Starting copy...
4 rows copied.
 Clock Time (ms.): total = 1 Avg = 0 (116000.00 rows per sec.)
```

Using *bcp* with Alternate Languages

Adaptive Server stores data using its default character set, which is configured during installation. If your terminal does not support that default character set, it may send confusing characters to **bcp** when you respond to prompts either by typing or by using host file scripts.

Omitting all character set options causes bcp to use the character set that was named as the default for the platform. This default can cause communication problems:

- The default is not necessarily the same character set that was configured for Adaptive Server.
- The default may not necessarily be the character set that the client is using.

For more information about character sets and the associated flags, see the *System Administration Guide*.

Copy In and Batch Files

Batching applies only to bulk copying in; it has no effect when copying out. By default, Adaptive Server copies all the rows in batches of 1000 lines. To specify a different batch size, use the command line option (**-b***batch_size*).

When data is being copied in, it can be rejected by either Adaptive Server or **bcp**.

- Adaptive Server treats each batch as a separate transaction. If the server rejects any row in the batch, it rolls back the entire transaction.
- When **bcp** rejects a batch, it then continues to the next batch. Only fatal errors roll back the transaction.
- Adaptive Server generates error messages on a batch-by-batch basis, instead of row-by-row, and rejects each batch in which it finds an error. Error messages appear on your terminal and in the error file.
- **bcp** saves its error messages to an error file. A **bcp** stored error, for example, might be when Adaptive Server encounters a duplicate row for a table that has a unique index.

Improving Recoverability

To ensure better recoverability:

- Break large input files into smaller units.

  For example, if you use **bcp** with a batch size of 100,00 rows to bulk copy in 300,000 rows, and a fatal error occurs after row 200,000, **bcp** would have successfully copied in the first two batches—200,000 rows—to Adaptive Server. If you had not used batching, **bcp** would not have been able to copy in any rows to Adaptive Server.

- Set the **trunc log on chkpt** to **true** (on).

  The log entry for the transaction is available for truncation after the batch completes. If you copy into a database that has the **trunc log on chkpt** database option set on (**true**), the next automatic checkpoint removes the log entries for completed batches. This log cleaning breaks up large **bcp** operations and keeps the log from filling.

- Set **-b***batch_size* to **1**.

The batch size parameter set to 1 causes **bcp** to reject only the defective row. The error log from this setting allows you to identify exactly which row failed.

Note Because **bcp** creates 1 data page per batch, and setting **-b***batch_size* to 1 creates data pages with 1 row on each page, this setting causes the data to load slowly and takes up storage space.

Batches and Partitioned Tables

When you bulk copy data into a partitioned table without specifying a partition number, Adaptive Server randomly assigns each batch to an available partition. Copying rows in a single batch places all those rows in a single partition, which can lead to load imbalance in the partitioned table.

To help keep partitioned tables balanced, use a small batch size when bulk copying data or specify the partition ID during the **bcp** session. For information about partitioning tables, see the *Performance and Tuning Guide*.

Copy Out and *text* and *image* Data

When you copy out text or image data, Adaptive Server, by default, copies only the first 32K of data in a text or image field. The -T*text_or_image_size* parameter allows you to specify a different value. For example, if the text field to copy out contains up to 40K of data, you can use the following command to copy out all 40K:

```
bcp pubs2..publishers out -T40960
```

Note If a text or image field is larger than the given value or the default, **bcp** does not copy out the remaining data.

Specifying a Network Packet Size

To improve the performance of large bulk copy operations, you may want to use larger network packet sizes than the defaults. The -A*size* option specifies the network packet size to use for the **bcp** session that you are beginning.

The value of *size* must be:

- Between the values of the **default network packet size** and **max network packet size** configuration parameters, and
- A multiple of 512.

Note The new packet size remains in effect for the current **bcp** session only.

For example:

```
bcp pubs2..authors out -A 2048 -T40960
```

specifies that Adaptive Server send 40K of text or image data using a packet size of 2048 bytes for the **bcp** session.

Copy In and Error Files

When you specify the **-e** *error_file* option with copy in, **bcp** stores the rows that it cannot copy in to Adaptive Server in the specified error file.

- The error file stores the following elements:
  - A line that indicates which row failed and the error that occurred, and
  - A line that is an exact copy of the row in the host file.
- If the file name specified after **-e** already exists, **bcp** overwrites the existing file.
- If **bcp** does not encounter any errors, it does not create the file.

bcp in detects two types of errors:

- Data conversion errors
- Errors in building the row; for example, attempts to insert a NULL into columns that do not accept null values or to use invalid data formats, such as a 3-byte integer

The copy in process displays error messages on your monitor.

The following example loads the *newpubs* file into the *publishers* database, storing any error rows in the *pub_err* file:

```
bcp pubs2..publishers in newpubs -epub_err
```

Keep the following in mind when working with error files generated by copy in:

- **bcp** stores rows in an error file only when the **bcp** program itself detects the error.
- **bcp** continues to copy rows until **bcp** encounters the maximum number of error rows, at which point **bcp** stops the copy.
- **bcp** sends rows to Adaptive Server in batches, so **bcp** cannot save copies of rows that are rejected by Adaptive Server, for example, a duplicate row for a table that has a unique index.
- Adaptive Server generates error messages on a batch-by-batch basis, instead of row-by-row, and rejects the entire batch if it finds an error.
- It is not considered an error for Adaptive Server to reject duplicate rows if either **allow_dup_row** or **ignore_dup_key** was set when a table's index was created. The copy proceeds normally, but the duplicate rows are neither stored in the table nor in the **bcp** error file.

Copy Out and Error Files

During the copy out process, as with copy in, **bcp** overwrites any file of the same name and does not create an error file if no errors occurred.

There are two situations that cause rows to be logged in the error file during a copy out:

- A data conversion error in one of the row's columns
- An I/O error in writing to the host file

Keep the following in mind when working with error files generated by copy out:

- **bcp** logs rows in the error file in the default character format.
- All data values print as characters with tabs between the columns and a newline terminator at the end of each row.

Data Integrity: Defaults, Rules, and Triggers

To ensure integrity, **bcp** handles data to copy depending upon its element.

Defaults and Datatypes

When copying data into a table, **bcp** observes any defaults defined for the columns and datatypes. That is, if there is a null field in the data in a file, **bcp** loads the default value instead of the null value during the copy.

For example, here are two rows in a file to be loaded into *authors*:

```
409-56-7008,Bennet,David,415 658-9932,622 Pine St.,Berkeley,CA,USA,94705213-
46-8915,Green,Marjorie,,309 63rd St. #411,Oakland,CA,USA,94618
```

Commas separate the fields; a newline terminator separates the rows. Note that there is no phone number for Marjorie Green. Because the *phone* column of the *authors* table has a default of "unknown," the rows in the loaded table look like this:

```
409-56-7008  Bennet  David 415 658-9932  622 Pine St.
   Berkeley  CA  USA  94705
213-46-8915  Green   Marjorie  unknown       309 63rd St. #411
   Oakland   CA  USA  94618
```

Rules and Triggers

bcp, to enable its maximum speed for loading data**,** does not fire rules and triggers.

To find any rows that violate rules and triggers, copy the data into the table and run queries or stored procedures that test the rule or trigger conditions.

How *bcp* Differs from Other Utilities

The **bcp** utility, which copies entire tables or portions of a single table, is distinct from the other utilities that move data from one place to another.

The following list names these other utilities and their commands and describes how you can best use them to move data.

- The SQL commands **dump database**, **load database**, **dump transaction**, and **load transaction**.

  Use these commands for backup purposes only. Unlike **bcp**, the **dump** commands create a physical image of the entire database.

  You must use **load database** or **load transaction** to read data backed up with **dump database** or **dump transaction**.

  For information on using the SQL **dump** and **load** commands, see the *System Administration Guide* and the *Adaptive Server Reference Manual*.

- The data modification commands **insert**, **update**, and **delete**.

  Use these commands, respectively, to add new rows to, change existing rows in, or remove rows from a table or view.

  - Use the **insert** command with a **select** statement to move data between tables.
  - Use the **select** statement with an **into** clause to create a new table, based on 1) the columns named in the **select** statement, 2) the tables named in the **from** clause, and 3)data in the rows named in the **where** clause.

  For details on adding, changing, and deleting data, see **insert**, **update**, and **delete** in the *Adaptive Server Reference Manual*.

CHAPTER 3

Using *dsedit*

This chapter explains how to use the **dsedit** utility to configure a directory service or the *sql.ini* file. Topics include:

Getting Started with *dsedit*

dsedit is a Windows 95 and Windows NT utility that allows you to configure a directory service or the *sql.ini* file.

Starting *dsedit*

You can start **dsedit** from the command prompt, the Windows NT Explorer, or the Sybase for Windows NT program group:

- To start **dsedit** from the command prompt, enter:

  ```
  dsedit
  ```

  You can specify the following command line arguments:

Argument	Description
-d *dsname*	Specifies which directory service to connect to. *dsname* is the local name of the directory service, as listed in the *libtcl.cfg* file. If you do not specify the -d *dsname* argument, dsedit presents a list of directory service options in the first dialog box.
-l *path*	Specifies the path to the *libtcl.cfg* file, if other than *SYBASE_home\INI*. Use this argument only if you want to use a *libtcl.cfg* file other than the one located in *SYBASE_home\INI*.

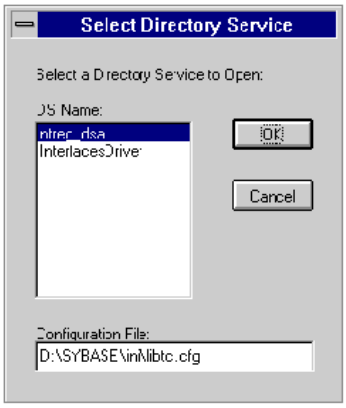
- To start **dsedit** through the Windows NT Explorer:

  a Go to the *Sybase_home\BIN* directory.

  b Double-click on the *DSEDIT.exe* file.

To start **dsedit** from the Sybase for Windows program group, choose Sybase for Windows NT from the Start menu. Choose **dsedit** from the Sybase for Windows NT menu. The following dialog box appears:

Figure 3-1: Selecting directory services



Opening a Session

The Select Directory Service dialog box allows you to open a session with a directory service. You can open a session with:

- Any directory service that has a driver listed in the *libtcl.cfg* file
- The *sql.ini* file

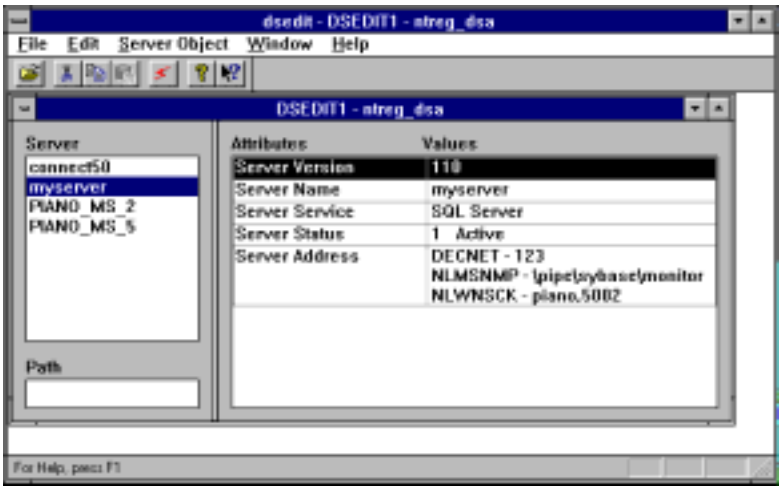
To open a session:

- Double-click on the local name of the directory service you want to connect to, as listed in the DS Name box, or
- Click on the local name of the directory service you want to connect to, as listed in the DS Name box, and click the OK button.

Note **dsedit** uses the SYBASE environment variable to locate the *libtcl.cfg* file. If the SYBASE environment variable is not set correctly, **dsedit** cannot locate the *libtcl.cfg* file.

The following dialog box appears:

Figure 3-2: Using dsedit to edit an interfaces file



The session number and local name of the directory service appear in the header bar. In the illustration above, the session number and local directory service name are “DSEDIT1 - ntreg_dsa.”

Opening Additional Sessions

dsedit allows you to have multiple sessions open at one time. To open additional sessions:

1 Choose Open Directory Service from the File menu.

The Select Directory Service box appears.

2 Double-click the local name of the directory service to which you want to be connected (or click on the directory service name and click OK).

Opening multiple sessions allows you to copy entries between directory services. See “Copying Server Entries” on page 67 for more information.

Switching Between Sessions

If you have multiple sessions open at one time, you need to activate a session before you can work in it. To activate a session:

- Click in the session window, or
- Choose the session from the Window menu.

The **dsedit** title bar shows which session is active.

Making and Modifying Server Entries

Once you open a session with a directory service or the *sql.ini* file, you can add, modify, rename, and delete server entries associated with that session.

The server entries associated with the session appear in the Server box (see Figure 3-2 on page 62). Click on a server entry to select it.

Each server entry is made up of a set of attributes. The attributes are described in Table 3-1:

Table 3-1: Server attributes

Attribute Name	Type of Value	Description	Default Value
Server Version	Integer	Version level of the server object definition. Sybase provides this attribute to identify future changes to the object definition.	110
Server Name	Character string	Server name.	N/A
Server Service	Character string	A description of the service provided by the server. This value can be any meaningful description.	Adaptive Server
Server Status	Integer	The operating status of the server. Values are: • 1: Active • 2: Stopped • 3: Failed • 4: Unknown	4
Security Mechanism	Character string	Object identifier strings (OID) that specify the security mechanisms supported by the server. This attribute is optional. If it is omitted, Open Server allows clients to connect with any security mechanism for which Open Server has a corresponding security driver.	N/A

Attribute Name	Type of Value	Description	Default Value
Server Address	Character string	One or more addresses for the server. The format of the address varies by protocol, and some protocols allow more than one format. The options are: • *TCP/IP* (two formats) 1. *computer name*,*port number* 2. *ip-address*,*portnumber* • *Named Pipe* *pipe name*: "\pipe" is a required prefix to all pipe names. Server pipes can be only local. (Local) *\pipe\sql\'uery* (Remote) *\\computer_name\pipe\sql\'uery* • *IPX/SPX* (three formats) 1. *server name* 2. *net number*,*node number*,*socket number* 3. *server name*, *socket number* • *DECnet* (four formats) 1. *area number.node number,object name* 2. *area number.node number,object number* 3. *node name,object name* 4. *node name,object number*	N/A

Adding a Server Entry

To add a server entry:

1 Choose Add from the Server Object menu.

The Input Server Name box appears.

2 Type a server name in the Server Name box.

3 Click the OK button.

The server entry appears in the Server box. To specify an address for the server, you must modify the entry.

Modifying a Server Entry

You can modify any attribute of a server entry. To modify a server attribute:

1 Click on a server entry in the Server box.

2 Click on the attribute you want to modify in the Attributes box.

3 Choose Modify Attribute from the Server Object menu.

A dialog box appears that shows the current value of the attribute.

4 Type a new value for the attribute, or select a value from the drop-down list.

See Table 3-1 on page 63 for a description of each attribute.

5 Click the OK button.

Renaming a Server Entry

To rename a server entry:

1 Click on a server entry in the Server box.

2 Choose Rename from the Server Object menu.

The Input Server Name box appears.

3 Type a new name for the server entry in the Server Name box.

4 Click the OK button.

Deleting Entries

To delete a server entry:

1 Click on a server entry in the Server box.

2 Choose Delete from the Server Object menu.

Using the *Ping* Command

You can verify your network connection with a server using the **Ping** server command from the Server Object menu. To ping a server:

1 Click on a server entry in the Server box.

2 Select the **Ping** command from the Server Object menu.

The Ping dialog box appears.

3 Click on the address you want to ping.

4 Click the Ping button.

A message box appears to notify you if the connection is successful.

Copying Server Entries

dsedit allows you to copy server entries within a session and between sessions. This includes copying entries from a *sql.ini* file to a directory service.

Copying Entries Within a Session

To copy server entries within the current session:

1 Click on one or more server entries in the Server box.

Use the Shift key to select multiple entries.

2 Click the Copy button (below the menu bar), or choose Copy from the Edit menu.

3 Click the Paste button (below the menu bar), or choose Paste from the Edit menu.

dsedit appends the copied server entries with a version number of *_n*. You can rename the copied server entries using the Rename command in the Server Object menu. See “Renaming a Server Entry” on page 65 for more information.

Copying Entries Between Sessions

To copy server entries between sessions:

1 Open a session with the directory service or *sql.ini* file that you want the entries copied to.

To open a session, choose Open Directory Service from the File menu. See “Opening Additional Sessions” on page 62 for more information.

2 Click on one or more server entries in the Server box of the session that you want the entries copied from.

Use the Shift key to select multiple entries.

3 To copy the server entries, click the Copy button (below the menu bar), or choose Copy from the Edit menu.

To cut the server entries, click the Cut button (below the menu bar), or choose Cut from the Edit menu.

4 Activate the session where you want to paste the server entries.

See “Switching Between Sessions” on page 62 for instructions for activating a session.

5 Click the Paste button (below the menu bar), or choose Paste from the Edit menu.

You can rename the copied server entries using the Rename command in the Server Object menu. See “Renaming a Server Entry” on page 65 for more information.

Exiting *dsedit*

To exit **dsedit**, choose Exit from the File menu.

CHAPTER 4

Utility Commands Reference

This chapter contains reference pages for the following utility program commands:

Launching utility programs

If a utility has an icon in the Sybase for Windows or Sybase for Windows NT program group, double-click the icon to launch the utility program.

If a utility does not have an icon in the program group, enter the utility program command at the Windows or Windows NT Command Prompt to launch the utility program.

Place characters that have special meaning to the Command Prompt in quotes; for example, the backslash (\), asterisk (*), slash (/), and spaces. You can precede some special characters with the backslash (\) to "escape" them. This prevents the Command Prompt from interpreting the special characters.

Table 4-1 describes the utility programs available with Adaptive Server for Windows and Windows NT.

Table 4-1: Utility programs for Windows NT

Utility Name	Executable File Name	Description
bcksrvr	*bcksrvr.exe*	Executable form of the Backup Server™ program.
bcp	*bcp.exe*	Copies rows in a database table to or from an operating system file in a user-specified format.
bldmastr	*bldmastr.exe*	Builds the master device and creates the *master*, *model*, and *tempdb* databases on the device.
defncopy	*defncopy.exe*	Copies definitions for specified views, rules, defaults, triggers, procedures, or reports from a database to an operating system file or from an operating system file to a database.
dsedit	*dsedit.exe*	Edits the *sql.ini* interfaces file.
extrjava	*extrjava.exe*	Copies a retained JAR from an Adaptive Server to a client file.
instjava	*instjava.exe*	Installs a JAR from a client file into an Adaptive Server.
isql	*isql.exe*	Interactive SQL parser to Adaptive Server.
langinst	*langinst.exe*	Installs one new language on the Adaptive Server.
sqlsrvr	*sqlsrvr.exe*	Executable form of the Adaptive Server program.
srvmgr	*srvmgr.exe*	Starts Adaptive Server and Backup Server as Windows NT services.
wbcp	*wbcp.exe*	Windows application version of bcp.
wdefncopy	*wdefncop.exe*	Windows application version of defncopy.

Utility Name	Executable File Name	Description
wdllvers	*wdllvers.exe*	Provides information about the Sybase DLLs (dynamic link libraries) loaded into memory.
wisql32	*wisql32.exe*	Win 32-bit version of isql.
wisql	*wisql.exe*	Windows application version of isql.

Utilities Quick Reference

This section provides a quick reference for the utilities, divided into the following categories:

- Installation or Configuration Utilities
- Utilities for Languages, Character Sets and Sort Orders
- Utilities to Start Servers
- Database Creation and Manipulation Utilities
- Utilities to Gather Information

Installation or Configuration Utilities

Use the following utilities to install or configure databases:

- **dsedit** – allows you to create and modify network connection information in the interfaces file.

Utilities for Languages, Character Sets and Sort Orders

Use the following utilities to set languages, character sets and sort orders:

- **langinst** – installs a new language on an Adaptive Server.
- **charset** – loads the character sets and sort order files.

Utilities to Start Servers

Use the following utilities to start servers manually:

- **bcksrvr** – starts the Backup Server executable. Use the **srvmgr** utility instead of this utility to start Backup Server manually.
- **sqlsrvr** – starts the Adaptive Server executable. Use the **services manager** utility instead of this utility to start Adaptive Server manually.

- **srvmgr** – starts, pauses, and stops Adaptive Server, Backup Server, and Adaptive Server Monitor™ as Windows NT services.

Database Creation and Manipulation Utilities

Use the following utilities to create and manipulate databases:

- **bcp** – copies a database table to or from an operating system file in a user-specified format.
- **bldmastr** – builds the master device and creates the *master*, *model*, and *tempdb* databases on the device.
- **extrjava** – copies a retained JAR and the classes it contains from an Adaptive Server to a client file.
- **instjava** – installs a JAR from a client file into an Adaptive Server database.
- **optdiag** – displays optimizer statistics or loads updated statistics into system table.

Utilities to Gather Information

Use the following utilities to gather information:

- **wdllvers** – provides information about the Sybase DLLs (dynamic link libraries) that are loaded into memory.

bcksrvr

Description

The executable form of the Backup Server program.

Syntax

bcksrvr
[-C*server_connections*]
[-S*b_servername*]
[-I*sqlini_filename*]
[-e*error_log_file*]
[-M*sybmbuf_binary*]
[-N*network_connections*]
[-T*trace_value*]
[-L*Sybase_language_name*]
[-J*Sybase_character_set_name*]
[-P*active_service_threads*]
[-V*level_number*]

or

bcksrvr -v

Parameters

-C*server_connections*
specifies the number of server connections for the Backup Server. The Backup Server requires:

- Two connections for each dump session
- One connection for each load session
- One connection for volume change messages

Allow a maximum of three times the number of expected concurrent dump and load sessions. The default value is 20 server connections.

-S*b_servername*
specifies the name of the Backup Server to start. The default is *server_name_BS*. This entry must specify the name of a release 12.0 Backup Server that is listed in the *sql.ini* interfaces file.

-I*sqlini_filename*
specifies the name and location of the interfaces file (*sql.ini*) to search when connecting to Backup Server. If **-I** is omitted, **bcksrvr** looks for a file named *sql.ini* in the *ini* subdirectory of your Sybase release directory.

-e*error_log_file*
specifies the name and location of the Backup Server error log file used to report Open Server™ internal errors, errors that halt the Backup Server, and errors for disconnected sessions. All other errors are sent to the **notify** destination specified in the **dump database**, **dump transaction**, **load database**, and **load transaction** commands.

-M*sybmbuf_binary*
specifies the full path name of the **sybmbuf.exe** executable. Use this parameter only when starting Backup Server from a directory other than the *bin* directory of the Sybase installation tree, or when using a diagnostic version of **sybmbuf.exe**.

-N*network_connections*
specifies the number of total network connections (DBPROCESSes) that the master Backup Server can originate. The default value is 25.

-T*trace_value*
interprets *trace_value* as a bit mask (base-2 number). The 1 bits in *trace_value* correspond to Open Server trace flags to turn on. If you specify more than one **-T** parameter on the command line, the final **-T** value overrides the values from earlier **-T** parameters. The *trace_value* must be a positive integer.

-L*Sybase_language_name*
specifies the default language for Backup Server. If not specified, Backup Server uses the locale specified by the LC_ALL or LANG environment variables. If these variables are not set, Backup Server searches for the "default" entry in *locales.dat*.

Note The **-L** parameter does not override the value set in the LANG environment variable.

-J*Sybase_character_set_name*
specifies the default character set for Backup Server. If not specified, Backup Server uses the locale specified by the LC_ALL or LANG environment variable. If these variables are not set, Backup Server searches for the "default" entry in *locales.dat*.

-P*active_service_threads*
allows you to increase the number of stripes during multiple dump/load operations (with a maximum of 32 stripes per single operation). The default system-wide limit is 48, which includes all dumps and loads.

-v
prints the version number and copyright message of the **bcksrvr** software and then exits. Works only when used in conjunction with the **-S** parameter.

-V*level_number*

limits the messages that are printed to the Backup Server error log. The *level_number* variable determines the degree of error verbosity (-V) for Backup Server:

- **-V3** – displays only completion messages from a normal **dump** or **load** command and the following types of messages:

  Error messages from Backup Server and **sybmultbuf**

  Other **sybmultbuf** messages

  Volume change messages

  Open Server™ messages

  Trace print messages

  Informational messages from the System & Tape Auto Config modules

- **-V2** – displays:

  All **-V3** messages *plus*

  File creation and file mount messages

- **-V1** – displays:

  All **-V2** messages *plus*

  Phase messages

- **-V0** (default) – displays:

  All messages, including backup progress

This limitation does not involve the messages that are sent to the client or console as determined by the **NOTIFY=** parameter in a **dump** or **load** command.

This option also does not affect logging for the following message types:

- Open Server messages,
- Trace printing messages from **bs_traceprint**, and
- **sybmultbuf** messages

Usage

- Start Backup Server using the **services manager** utility rather than by executing the **bcksrvr** program directly. To change default values, use Server Config to change the command line parameters of the Backup Server. See the *Adaptive Server Enterprise Configuration Guide for Windows 95, Windows 98, and Windows NT* for details.

- If you do not specify a Backup Server name with the **-S** parameter, and you have not set the environment variable DSLISTEN, **bcksrvr** uses the default Backup Server name *server_name_BS*. The value of the DSLISTEN environment variable overrides the default value, and the **-S** parameter overrides both the default and the value specified in DSLISTEN.
- Whenever possible, the Backup Server and any Adaptive Servers that dump or load directly through the Backup Server should share the same *sql.ini* file. The *sql.ini* file Backup Server uses must contain entries for:
  - Backup Server
  - Any other Backup Server with which this Backup Server communicates
- Trace flags cause the Backup Server to print information regarding its operation while it is running. This is useful for debugging problems in the Backup Server. See the *Open Server Server-Library/C Reference Manual* for more details on trace flags. The Windows NT Backup Server does not support use of the Open Server-defined SRV__TR symbols for **-T**.
- If Backup Server cannot find the *locales* and *charsets* directories specified by the **-L** and **-J** parameters, or if these parameters specify an incorrect language and character set combination, Backup Server issues an error message and uses the default language and character set.

See also

Utility	**srvmgr**

bcp

Description

Copies a database to or from an operating system file in a user-specified format.

Syntax

```
bcp [[database_name.]owner.][view_name | table_name
        [:partition_id]] {in | out} datafile
[-c] [-E] [-n] [-X]
 [-a display_charset]
 [-A size]
 [-b batchsize]
 [-e errfile]
 [-f formatfile]
 [-F firstrow]
 [-g id_start_value]
 [-I interfaces_file]
 [-J client_charset]
 [-L lastrow]
 [-m maxerrors]
 [-P password]
 [-q datafile_charset]
 [-r row_terminator]
 [-R remote_server_principal]
 [-S server]
 [-t field_terminator]
 [-T text_or_image_size]
 [-U username]
 [-z language]
```

or

```
bcp -v
```

Parameters

database_name

is optional if the table being copied is in your default database or in *master*. Otherwise, you must specify a database name.

owner

is optional if you or the Database Owner owns the table being copied. If you do not specify an owner, **bcp** looks first for a table of that name that you own, and then looks for one owned by the Database Owner. If another user owns the table, you must specify the owner name or the command fails.

view_name

is the name of the view you are copying out.

table_name

is the name of the database table to copy.

Partition number *partition_number* does not exist in table *table_name*.

partition_id
is the identifier of the partition into which to copy.

in | **out**
is the direction of the copy. **in** indicates a copy from a file into the database table; **out** indicates a copy to a file from the database table or view.

datafile
is the full path name of an operating system file. The path name can be from 1 to 255 characters in length.

-c
performs the copy operation with char datatype as the default. This parameter does not prompt for each field; it uses char as the default storage type, no prefixes, \t (tab) as the default field terminator, and \n (newline) as the default row terminator.

-E
explicitly specifies the value of a table's IDENTITY column.

By default, when you bulk copy data into a table with an IDENTITY column, **bcp** assigns each row a temporary IDENTITY column value of 0. As **bcp** inserts each row into the table, the server assigns the row a unique, sequential IDENTITY column value, beginning with the value 1. If you specify the **-E** flag when copying data into a table, **bcp** prompts you to enter an explicit IDENTITY column value for each row. If the number of inserted rows exceeds the maximum possible IDENTITY column value, Adaptive Server returns an error.

By default, when you bulk copy data from a table with an IDENTITY column, **bcp** excludes all information about the column from the output file. If you specify the **-E** flag, **bcp** copies the existing IDENTITY column values into the output file.

You cannot use the **-E** and **-g** flags together.

-n

performs the copy operation using native (operating system) formats. Specifying the **-n** parameter means **bcp** will not prompt for each field. Files in native data format are not human-readable.

Warning! Do not use **bcp** in native format for data recovery or salvage or to resolve an emergency situation. Do not use **bcp** in native format to transport data between different hardware platforms, different operating systems, or different major releases of Adaptive Server. Do not use field terminators (**-t**) or row terminators (**-r**) with **bcp** in native format. Results are unpredictable and data could become corrupted. Using **bcp** in native format can create flat files that cannot be reloaded into Adaptive Server and it may be impossible to recover the data. If you are unable to re-run **bcp** in character format (for example, a table was truncated or dropped, hardware damage occurred, a database was dropped, and so on) the data will be unrecoverable.

-X

specifies that, in this connection to the server, the application initiates the login with client-side password encryption. **bcp** (the client) specifies to the server that password encryption is desired. The server sends back an encryption key, which **bcp** uses to encrypt your password, and the server uses the key to authenticate your password when it arrives.

-a *display_charset*

allows you to run **bcp** from a terminal where the character set differs from that of the machine on which **bcp** is running. Use **-a** in conjunction with **-J** to specify the character set translation file (*.xlt* file) required for the conversion. Use **-a** without **-J** only if the client character set is the same as the default character set.

The following error message will appear if the character translation file(s) named with the **-a** parameter is missing, or you mistype the name(s):

```
Error in attempting to determine the size of a pair of
translation tables.:'stat' utility failed.
```

-A *size*

specifies the network packet size to use for this **bcp** session. For example:

```
bcp pubs2..titles out table_out -A 2048
```

sets the packet size to 2048 bytes for this **bcp** session. *size* must be between the values of the **default network packet size** and **maximum network packet size** configuration variables, and it must be a multiple of 512.

Use larger-than-default network packet sizes to improve the performance of large bulk-copy operations.

-b *batchsize*
is the number of rows per batch of data copied (the default is to copy all the rows in one batch). Batching applies only when you are bulk copying in; it has no effect on bulk copying out.

-e *errfile*
is the full path name of an error file where **bcp** stores any rows that it was unable to transfer from the file to the database. Error messages from the **bcp** program appear on your terminal. **bcp** creates an error file only when you specify this parameter.

-f *formatfile*
is the full path name of a file with stored responses from a previous use of **bcp** on the same table. After you answer **bcp**'s format questions, it prompts you to save your answers in a format file. Creation of the format file is optional. The default file name is *bcp.fmt*. The **bcp** program can refer to a format file when you are copying data so that you do not have to duplicate your previous format responses interactively. Use the **-f** parameter only if you previously created a format file that you want to use now for a copy in or copy out. If you do not specify this parameter, **bcp** queries you for format information interactively.

-F *firstrow*
is the number of the first row to copy from an input file (default is the first row).

-g *id_start_value*
specifies the value of the IDENTITY column to use as a starting point for copying data in.

You cannot use the **-g** and **-E** flags together.

-I *interfaces_file*
specifies the name and location of the interfaces file to search when connecting to Adaptive Server. If you do not specify **-I**, **bcp** looks for an interfaces file (*sql.ini* on Windows platforms) located in the *ini* directory, which is below the directory specified by the SYBASE environment variable.

-J *client_charset*
specifies the character set to use on the client. **bcp** uses a filter to convert input between *client_charset* and the Adaptive Server character set.

-J *client_charset* requests that Adaptive Server convert to and from *client_charset*, the character set used on the client.

-J with no argument disables character set conversion. Use this if the client and server use the same character set.

Omitting **-J** sets the character set to a default for the platform, which may not necessarily be the character set that the client is using. See the *System Administration Guide* for more information about character sets and associated flags.

The following error message will appear if an incorrect or unrecognized character set is named with the **-J** parameter:

```
Unrecognized localization object. Using default value
'iso_1'.
Starting copy...
=> warning.
```

-L *lastrow*
is the number of the last row to copy from an input file (default is the last row).

-m *maxerrors*
is the maximum number of errors permitted before **bcp** aborts the copy. **bcp** discards each row that it cannot build, counting each rejected row as one error. If you do not include this parameter, **bcp** uses a default value of 10.

-P *password*
specifies an Adaptive Server password. If you do not specify **-P** *password*, **bcp** prompts for a password. You can leave out the **-P** flag If your password is NULL.

-q *datafile_charset*
runs **bcp** to copy character data to or from a file system that uses a character set different from the client character set.

In Japanese language environments, the **-q** flag translates Hankaku Katakana (half-width characters) into Zenkaku Katakana (full-width characters).

The following error message will appear if the character translation file(s) named with the **-q** parameter is missing, or you mistype the name(s):

```
Error in attempting to determine the size of a pair of
```

```
translation tables.:'stat' utility failed.
```

Note The ascii_7 character set is compatible with all character sets. If either the Adaptive Server character set or the client's character set is set to ascii_7, any 7-bit ASCII character is allowed to pass between client and server unaltered. Other characters produce conversion errors. Character set conversion issues are covered more thoroughly in the *System Administration Guide*.

-r *row_terminator*
specifies the row terminator.

Warning! Do not use **-t** or **-r** parameters with **bcp** in native format. Results are unpredictable and data could become corrupted.

Note When specifying terminators from the command line with the **-t** or **-r** parameter, you must escape characters that have special significance to the Windows NT Command Prompt shell (see the following examples). Either place a backslash in front of the special character or enclose it in quotes. This is not necessary when **bcp** prompts you (interactive mode).

-R *remote_server_principal*
specifies the principal name for the server as defined to the security mechanism. By default, a server's principal name matches the server's network name (which is specified with the **-S** parameter or the DSQUERY environment variable). Use the **-R** parameter when the server's principal name and network name are not the same.

-S *server*
specifies the name of the Adaptive Server to which to connect. If you specify **-S** with no argument, **bcp** uses the server specified by the DSQUERY environment variable.

-t *field_terminator*
specifies the default field terminator.

Warning! Do not use **-t** or **-r** parameters with **bcp** in native format. Results are unpredictable and data could become corrupted.

-T *text_or_image_size*
allows you to specify, in bytes, the maximum length of text or image data that Adaptive Server sends. The default is 32K. If a text or an image field is larger than the value of **-T** or the default, **bcp** does not send the overflow.

-U *username*
specifies an Adaptive Server login name. If you do not specify *username*, **bcp** uses the current user's operating system login name.

-z *language*
is the official name of an alternate language the server uses to display **bcp** prompts and messages. Without the **-z** flag, **bcp** uses the server's default language. You can add languages to an Adaptive Server during installation or afterward, using the **langinst** utility or the **sp_addlanguage** stored procedure.

The following error message will appear if an incorrect or unrecognized language is named with the **-z** parameter:

```
Unrecognized localization object. Using default value
'us_english'.
Starting copy...
=> warning.
```

-v
reports the current version and copyright message of the **bcp** program.

Examples

```
bcp pubs2..publishers out pub_out -c -t , -r \r
```

Copies data out of the *publishers* table in character format (using char for all fields) using the **-c** parameter. The **-t** *field_terminator* parameter ends each field with a comma, and the **-r** *row_terminator* parameter ends each line with a Return. **bcp** prompts only for a password.

```
bcp pubs2..publishers out pub_out

Password:
Enter the file storage type of field pub_id [char]:
Enter prefix length of field pub_id [0]:
Enter length of field pub_id [4]:
Enter field terminator [none]:
Enter the file storage type of field pub_name [char]:
Enter prefix length of field pub_name [1]:
Enter length of field pub_name [40]:
Enter field terminator [none]:
Enter the file storage type of field city [char]:
Enter prefix length of field city [1]:
Enter length of field city [20]:
Enter field terminator [none]:
```

```
Enter the file storage type of field state [char]:
Enter prefix length of field state [1]:
Enter length of field state [2]:
Enter field terminator [none]:
```

Copies data from the *publishers* table to a file named *pub_out* for later reloading into Adaptive Server. Press Return to accept the defaults specified by the prompts. The same prompts appear when you copy data into the *publishers* table.

```
bcp pubs2..publishers out pub_out -c -t , -r \r
```

Copies data out of the *publishers* table in character format (using char for all fields). The **-t** *field_terminator* parameter ends each field with a comma, and the **-r** *row_terminator* parameter ends each line with a Return. **bcp** prompts only for a password.

```
bcp pubs2..publishers out pub_out
```

```
Password:
Enter the file storage type of field pub_id [char]:
Enter prefix length of field pub_id [0]:
Enter length of field pub_id [4]:
Enter field terminator [none]:
Enter the file storage type of field pub_name [char]:
Enter prefix length of field pub_name [1]:
Enter length of field pub_name [40]:
Enter field terminator [none]:
Enter the file storage type of field city [char]:
Enter prefix length of field city [1]:
Enter length of field city [20]:
Enter field terminator [none]:
Enter the file storage type of field state [char]:
Enter prefix length of field state [1]:
Enter length of field state [2]:
Enter field terminator [none]:
```

Copies data from the *publishers* table to a file named *pub_out* for later reloading into Adaptive Server. Press Return to accept the defaults specified by the prompts. The same prompts appear when you copy data into the *publishers* table.

Usage

- See Chapter 2, “Using bcp to Transfer Data to and from Adaptive Server” for an in-depth discussion of **bcp**.
- See the *Performance and Tuning Guide* for more information on how changing certain parameters can affect **bcp** for large batches.

- **bcp** provides a convenient, high-speed method for transferring data between a database table or view and an operating system file. **bcp** can read or write files in a wide variety of formats. When copying in from a file, **bcp** inserts data into an existing database table; when copying out to a file, **bcp** overwrites any previous contents of the file.
- You cannot copy into a view from an operating system file.
- Upon completion, **bcp** informs you of the number of rows of data successfully copied, the total time the copy took, the average amount of time, in milliseconds, that it took to copy one row and the number of rows copied per second.
- The current version of **bcp** ignores the **-y** *sybase_directory* parameter.
- You cannot use named pipes to copy files in or out.
- Error message format is different than previous versions of **bcp**. If you have scripts that perform routines based on the values of these messages you may need to re-write them, for example:

  The display message that indicates the number of rows transferred has been changed. During a session, this version of **bcp** periodically reports a running total of rows transferred. This message replaces the "1000 rows transferred" message displayed by the previous **bcp**.
- Do not use **-t** or **-r** parameters with **bcp** in native format. Results are unpredictable and data could become corrupted.

Copying Tables with Indexes or Triggers

- The **bcp** program is optimized to load data into tables that do not have indexes or triggers associated with them. It loads data into tables without indexes or triggers at the fastest possible speed, with a minimum of logging. Page allocations are logged, but the insertion of rows is not.

  When you copy data into a table that has one or more indexes or triggers, a slower version of **bcp** is automatically used, which logs row inserts. This includes indexes implicitly created using the unique integrity constraint of a **create table** statement. However, **bcp** does not enforce the other integrity constraints defined for a table.

Because the fast version of **bcp** inserts data without logging it, the System Administrator or Database Owner must first set the system procedure **sp_dboption**, **"DB"**, to **true**. If the option is not set to **true**, and you try to copy data into a table that has no indexes or triggers, Adaptive Server generates an error message. You do not need to set this option to copy data out to a file or to copy data into a table that contains indexes or triggers.

Note Because **bcp** logs inserts into a table that has indexes or triggers when you use slow **bcp**, the log can grow very large. You can truncate the log with **dump transaction** after the bulk copy completes and after you have backed up your database with **dump database**.

- While the **select into/bulkcopy/pllsort** option is on, you cannot dump the transaction log. Issuing **dump transaction** produces an error message instructing you to use **dump database** instead.

Warning! Be certain that you dump your database before you turn off the **select into/bulkcopy/pllsort** flag. If you have inserted unlogged data into your database, and you then perform a **dump transaction** before performing a **dump database**, you will not be able to recover your data.

- Fast **bcp** runs more slowly while a **dump database** is taking place.
- Table 4-2 shows which version **bcp** uses when copying in, the necessary settings for the **select into/bulkcopy/pllsort** option, and whether the transaction log is kept and can be dumped.

Table 4-2: Comparing fast and slow bcp

	select into/bulkcopy/pllsort	
	on	**off**
fast bcp (no indexes or triggers on target table)	OK dump transaction prohibited	bcp dump transaction prohibited
slow bcp (one or more indexes or triggers)	OK dump transaction prohibited	OK dump transaction OK

- By default, the **select into/bulkcopy/pllsort** option is off in newly created databases. To change the default situation, turn this option on in the *model* database.

Note The performance penalty for copying data into a table that has indexes or triggers in place can be severe. If you are copying in a very large number of rows, it may be faster to drop all the indexes and triggers beforehand with **drop index** (or **alter table**, for indexes created as a unique constraint) and **drop trigger**; set the database option; copy the data into the table; re-create the indexes and triggers; and then dump the database. Remember to allocate disk space for the construction of indexes and triggers—about 2.2 times the amount of space needed for the data.

Responding to *bcp* Prompts

When you copy data in or out using the **-n** (native format) or **-c** (character format) parameters, **bcp** prompts you only for your password, unless you supplied it with the **-P** parameter. If you do not supply either the **-n**, **-c** or **-f** *formatfile* parameter, **bcp** prompts you for information for each field in the table or view.

- Each prompt displays a default value, in brackets, which you can accept by pressing Return. The prompts include:
  - The file storage type, which can be character or any valid Adaptive Server datatype
  - The prefix length, which is an integer indicating the length in bytes of the following data
  - The storage length of the data in the file for non-NULL fields
  - The field terminator, which can be any character string
  - Scale and precision for numeric and decimal data types

  The row terminator is the field terminator of the last field in the table, view, or file.
- The bracketed defaults represent reasonable values for the datatypes of the field in question. For the most efficient use of space when copying out to a file:
  - Use the default prompts
  - Copy all data in the datatypes defined by their table
  - Use prefixes as indicated

- Do not use terminators
- Accept the default lengths

Table 4-3 shows the bcp prompts, defaults, and the possible alternate user responses:

Table 4-3: bcp prompts—their defaults and user responses

Prompt	Default Provided	Possible User Response
File Storage Type	Use database storage type for most fields except: char for varchar binary for varbinary	char, to create or read a human-readable file; any Adaptive Server datatype where implicit conversion is supported.
Prefix Length	0 for fields defined with char datatype (not storage type) and all fixed-length datatypes 1 for most other datatypes 2 for binary and varbinary saved as char 4 for text and image	0, if no prefix is desired; otherwise, defaults are recommended.
Storage Length	For char and varchar, use defined length. For binary and varbinary saved as char, use the default. For all other datatypes, use the maximum length needed to avoid truncation or data overflow.	Default values, or greater, are recommended.
Field or Row Terminator	None	Up to 30 characters or one of the following: \t tab \n newline \r carriage return \0 null terminator \ backslash

- **bcp** can copy data out to a file either as its native (database) datatype or as any datatype for which implicit conversion is supported for the datatype in question. **bcp** copies user-defined datatypes as their base datatype or as any datatype for which implicit conversion is supported. For more information on datatype conversions, see **dbconvert** in the *Open Client DB-Library/C Reference Manual* or the *Adaptive Server Reference Manual*.

  Note Be careful when you copy data from different operating systems because not all operating systems use the same native datatypes (for example, copying from NT into a UNIX server).

- A prefix length is a 1-, 2-, or 4-byte integer that represents the length of each data value. It immediately precedes the data value in the host file.
- Fields defined in the database as char, nchar, and binary are always padded with spaces (null bytes for binary) to the full length defined in the database. timestamp data is treated as binary(8).

  If data in the varchar and varbinary fields is longer than the length specified for copy out, **bcp** silently truncates the data in the file at the specified length.
- A field terminator string can be up to 30 characters long. The most common terminators are a tab (entered as "\t" and used for all columns except the last one), and a newline (entered as "\n" and used for the last field in a row). Other terminators are: "\0" (the null terminator), "\" (backslash), and "\r" (Return). When choosing a terminator, be sure that its pattern does not appear in any of your character data.

  For example, if you used tab terminators with a string that contained a tab, **bcp** would not be able to identify which tab represents the end of the string. **bcp** always looks for the first possible terminator, so, in this example it would find the wrong one.

  When a terminator or prefix is present, it affects the length of data transferred. If the length of an entry being copied out to a file is less than the storage length, it is immediately followed by the terminator or the prefix for the next field. The entry is not padded to the full storage length (char, nchar, and binary data is returned from Adaptive Server already padded to the full length).

When **bcp** is copying in from a file, data is transferred until either the number of bytes indicated in the "Length" prompt has been copied or the terminator is encountered. Once the number of bytes equal to the specified length has been transferred, the rest of the data is flushed until the terminator is encountered. When no terminator is used, the table storage length is strictly observed.

- Table 4-4 and Table 4-5 show the interaction of prefix lengths, terminators, and field length on the information in the file. "P" indicates the prefix in the stored table; "T" indicates the terminator; and dashes, (--) show appended spaces. An ellipsis (...) indicates that the pattern repeats for each field. The field length is 8 bytes for each column; "string" represents the 6-character field each time.

Table 4-4: Adaptive Server char data

	Prefix length = 0	***Prefix length–1, 2, or 4***
No terminator	string--string--.	Pstring--Pstring--.
Terminator	string--Tstring--T.	Pstring--TPstring--T.

Table 4-5: Other datatypes converted to char storage

	Prefix length = 0	***Prefix length–1, 2, or 4***
No terminator	string--string--.	PstringPstring.
Terminator	stringTstringT.	PstringTPstringT.

- The file storage type and length of a column do not have to be the same as the type and length of the column in the database table. [If the types and formats copied in are incompatible with the structure of the database table, the copy fails.
- File storage length generally indicates the maximum amount of data that can be transferred for the column, excluding terminators and/or prefixes.
- When copying data into a table, **bcp** observes any defaults defined for columns and user-defined datatypes. However, **bcp** ignores rules in order to load data at the fastest possible speed.
- **bcp** considers any data column that can contain a null value to be variable length, so use either a length prefix or a terminator to denote the length of each row of data.

- Data written to a host file in its native format preserves all of its precision. datetime and float values preserve all of their precision, even when they are converted to character format. Adaptive Server stores money values to a precision of one ten-thousandth of a monetary unit. However, when money values are converted to character format, their character format values are recorded only to the nearest two places.
- Before copying data that is in character format from a file into a database table, check the datatype entry rules in the *Adaptive Server Reference Manual*. Character data that is being copied into the database with **bcp** must conform to those rules. Note especially that dates in the undelimited format *(yy)yymmdd* may result in overflow errors if the year is not specified first.
- When you send host data files to sites that use terminals different from your own, inform them of the datafile_charset that you used to create the files.

See also

System procedures	sp_audit, sp_dboption, sp_displayaudit
System tables	sysaudits_01 – sysaudits_08

bldmastr

Description

Builds the master device and creates the *master*, *model*, and *tempdb* databases on the device.

Syntax

```
bldmastr
 [-d physicalname]
 [-c cntrltype]
 [-s size]
 [-m]
 [-x
```

or

```
bldmastr -v
```

Parameters

-d *physicalname*
is the physical name of the operating system file where the master device resides.

-c *cntrltype*
is the controller number for the master device. Together, *cntrltype* and *physicalname* specify the device. The default value for *cntrltype* is 0. Do not change this value unless instructed to do so by Sybase Technical Support.

-s *size*
is the size of the master device in 2K blocks. There are 512 2K blocks in 1MB. For example, a size of “15360” creates a 30MB master device. **bldmastr** verifies that the value you specify for this parameter does not exceed the space available to the master device, unless you use the **-m** parameter. See the installation or configuration guide for your platform for more information about the size of master devices.

-m
rewrites only the *master* database, without initializing the master device. Use this parameter when the *master* database is corrupted but the other databases on the master device are undamaged.

-x
rewrites only the *model* database, without changing the configuration block or initializing the master device. Use this parameter when the *model* database is corrupted and you cannot load it successfully from a backup. If you modified *model*, you must restore it from a backup after re-initializing it with this parameter.

-v
prints the version number and copyright message for **bldmastr** and then exits.

Examples

```
bldmastr -d d:\devices\master.dat -s10752
```

Initializes the operating system file *d:\devices\master.dat* as a 21MB master device and creates the system databases *master*, *model*, and *tempdb* on the device.

Usage

- The **bldmastr** program initializes the specified database device as an Adaptive Server master device and builds the *master* and *model* databases on it.
- The installer runs **bldmastr** and builds an initial *master* database on the database device you specify in answer to the program's prompts.
- If you run **bldmastr** with no parameters, it prompts for the following information:

  ```
  Master disk name? (default is master.dat) :
  Master disk size (in 2k blocks)? (default is 6144):
  Databases only? (default is N) (y or n) (same as -m)
  ```

  You must enter a response for each prompt.
- If you rebuild the *model* database, run the *instmodl* script, located in *%sybase%\scripts*. *instmodl* sets up the necessary permissions for the *model* database. To run the script enter:

  ```
  cd %sybase%\scripts
  isql -Usa -P password -S server_name < instmodl
  ```
- For information on backing up and recovering system databases, see the *System Administration Guide*.
- The password to the default "sa" account reverts to NULL after you run **bldmastr -m**, and the account is unlocked. Loading a backup of the *master* database restores the password and lock state to what they were when the dump was taken.

charset

Description

Loads the character sets and sort order files.

Syntax

```
charset
 [-P password]
 [-S server]
 [-I interface]
 sort_order | charset
```

or

```
charset -v
```

Parameters

-P*password*

specifies your password. If you do not specify **-P**, **charset** prompts for your password.

-S*server*

specifies the name of the server on which to change the character set and sort order.

-I*interface*

specifies the network interface used by the server.

sort_order

specifies the name of the sort order file Adaptive Server will use.

charset

specifies the character set Adaptive Server will use.

-v

displays the version number and copyright message for **charset.**

Usage

- Before using **charset** you must set your SYBASE environment variable to point to the current release directory.

Permissions

You must be a System Administrator to use **charset.**

See also

Command	set
Utility	**langinst**

defncopy

Description

Copies definitions for specified views, rules, defaults, triggers, or procedures from a database to an operating system file or from an operating system file to a database.

Note The **defncopy** utility cannot copy table definitions or reports created with Report Workbench™.

Syntax

```
defncopy
 [-U username]
 [-P password]
 [-S server]
 [-I sqlini_file]
 [-a display_charset]
 [-J client_charset]
 [-z language]
 [-X]
 {in filename dbname | out filename dbname
          [owner.]objectname [[owner.]objectname]...}
```

or

```
defncopy -v
```

Parameters

-U *username*
specifies a login name. Login names are case sensitive. If you do not specify a user name, **defncopy** uses the current Windows NT login name.

-P *password*
specifies your password. If you do not specify **-P**, **defncopy** prompts for your password.

-S *server*
specifies the name of the Adaptive Server to which to connect. If you specify **-S** with no argument, **defncopy** looks for a server named SYBASE. If you do not specify **-S**, **defncopy** uses the server specified by the DSQUERY environment variable.

-I *sqlini_file*
specifies the name and location of the interfaces file (*sql.ini*) to search when connecting to Adaptive Server. If you do not specify **-I**, **defncopy** looks for a file named *sql.ini* in the *ini* subdirectory of your Sybase release directory.

-a *display_charset*
runs **defncopy** from a terminal whose character set differs from that of the machine on which **defncopy** is running. Use **-a** in conjunction with **-J** to specify the character set translation file (*.xlt* file) required for the conversion. Use **-a** without **-J** only if the client character set is the same as the default character set.

Note The ascii_7 character set is compatible with all character sets. If either the Adaptive Server character set or the client character set is set to ascii_7, any 7-bit ASCII character can pass unaltered between client and server. Other characters produce conversion errors. See the*System Administration Guide* for more information on character set conversion.

-J *client_charset*
specifies the character set to use on the client. A filter converts input between *client_charset* and the Adaptive Server character set.

-J *client_charset* requests that Adaptive Server convert to and from *client_charset*, the client's character set.

-J with no argument sets character set conversion to NULL. No conversion takes place. Use this if the client and server are using the same character set.

Omitting **-J** sets the character set to a default for the platform. The default may not be the character set that the client is using. For more information about character sets and their associated flags, see the *System Administration Guide* and *Configuring and Administering Sybase Adaptive Server for Windows NT*.

-z *language*
is the official name of an alternate language that the server uses to display **defncopy** prompts and messages. Without the **-z** flag, **defncopy** uses the server's default language. You can add languages to Adaptive Server during installation or add them afterward, using the **langinst** utility or the stored procedure **sp_addlanguage**.

-X
initiates the login with client-side password encryption in this connection to the server. **defncopy** (the client) specifies to the server that password encryption is desired. The server sends back an encryption key, which **defncopy** uses to encrypt your password, and the server uses the key to authenticate your password when it arrives.

If **defncopy** crashes, the system creates a core file that contains your password. If you did not use the encryption option, the password appears in plain text in the file. If you used the encryption option, your password is not readable.

in | **out**
specifies the direction of definition copy.

filename
specifies the name of the operating system file destination or source for the definition copy. The copy out overwrites any existing file.

dbname
specifies the name of the database to copy the definitions from or to.

owner
is optional if you or the Database Owner own the table being copied. If you do not specify an owner, **defncopy** first looks for a table of that name that you own, and then looks for one owned by the Database Owner. If another user owns the table, you must specify the owner name or the command fails.

objectname
specifies name(s) of database object(s) for **defncopy** to copy out. Do not use *objectname* when copying definitions in.

Examples

```
defncopy -Usa -P -SMERCURY in new_proc stagedb
```

Copies definitions from the file *new_proc* into the database *stagedb* on server MERCURY. The connection with MERCURY is established with a user of name “sa” and a NULL password.

```
defncopy -S -z french out dc.out employees sp_calccomp
sp_vacation
```

Copies definitions for objects “*sp_calccomp*” and “*sp_vacation*” from the “*employees*” database on the SYBASE server to the file *dc.out*. Messages and prompts are displayed in “french”. The user is prompted for a password.

Usage

- Invoke the **defncopy** program directly from the Windows NT Command Prompt. **defncopy** provides a non-interactive way of copying out definitions (**create** statements) for views, rules, defaults, triggers, or procedures from a database to an operating system file. Alternatively, it copies in all the definitions from a specified file.

  You must have **select** permission on the *sysobjects* and *syscomments* tables to copy out definitions; you do not need permission on the object itself.

- You must have the appropriate **create** permission for the type of object you are copying in. Objects copied in belong to the copier. A System Administrator copying in definitions on behalf of a user must log in as that user to give the user proper access to the reconstructed database objects.

- The **in** *filename* or **out** *filename* and the database name are required and must be stated unambiguously. For copying out, use file names that reflect both the object's name and its owner.

- **defncopy** ends each definition that it copies out with the comment:

  ```
  /* ### DEFNCOPY: END OF DEFINITION  */
  ```

  Definitions created as text must end with this comment so that **defncopy** can copy them in successfully.

- Enclose values specified to **defncopy** in quotation marks, if they contain characters that could be significant to the shell.

Warning! Long comments of more than 100 characters that are placed before a **create** statement may cause **defncopy** to fail.

See also

System procedures	sp_checkreswords, sp_procqmode, sp_remap

dsedit

Description

Creates and modifies network connection information in the interfaces file.

Syntax

```
dsedit
```

or

```
dsedit -v
```

Parameters

-v

displays the version and copyright message of **dsedit**.

Usage

- For more information about the **dsedit** utility program, see Chapter 4, "Using dsedit." Also see the *Adaptive Server Enterprise Installation Guide for Windows NT* and the *Adaptive Server Enterprise Configuration Guide for Windows 95, Windows 98, and Windows NT*.
- **dsedit** creates and modifies network connection information in the interfaces file.
- You must set the SYBASE environment variable to the location of the current version of Sybase Adaptive Server before you can use **dsedit**.
- You must set the DISPLAY environment variable before invoking **dsedit**, unless you are only using the **-v** parameter to display the version number.

extrjava

Description

Copies a retained JAR and the classes it contains from an Adaptive Server into a client file.

Refer to *Java in Adaptive Server Enterprise* for more information about how this utility is used when Java is enabled in the database.

Syntax

extrtjava
-j *jar_name*
-f *file_name*
[-S *server*]
[-U *username*]
[-P *password*]
[-D *database*]
[-I *interfaces_file*]
[-a *display_charset*]
[-J *client_charset*]
[-z *language*]
[-t *timeout*] [-v]

Parameters

-j *jar_name*
the name assigned to the retained JAR in the database that is the source of the transfer.

-f *file_name*
the name of the client file that is the target of the transfer.

-S *server*
the name of the server.

-U *user_name*
an Adaptive Server login name. If you omit the **–U** flag and parameter, or if you specify the **–U** flag with no parameter, Adaptive Server uses the current user's operating system login name.

-P *password*
an Adaptive Server password. If you omit the **–P** flag and parameter, **extrjava** prompts for a password. If you specify the **–P** flag with no password, the null password is used.

-D *database*
the name of the database in which to install the JAR. If you omit the **–D** flag, or if you specify the **–D** flag with no parameter, the user's default database is used.

-I *interfaces_file*
the name and location of the interfaces file to search when connecting to Adaptive Server. If you omit the **–I** flag and parameter, or if you specify the **–I** flag with no parameter, the interfaces file in the directory designated by your SYBASE environment variable is used.

-a *display_charset*
allows you to use **extrjava** from a machine where the character set differs that of the server. Use **-a** in conjunction with **-J** to specify the character set translation file (*.xlt* file) required for the conversion. Use **-a** without **-J** only if the client character set is the same as the default character set.

-J *client_charset*
specifies the character set to use on the client. **extrjava** uses a filter to convert input between client_charset and the Adaptive Server character set.

-J client_charset requests that Adaptive Server convert to and from client_charset, the character set used on the client.

-J with no argument disables character set conversion. Use this if the client and server use the same character set.

Omitting **-J** sets the character set to a default for the platform, which may not necessarily be the character set that the client is using. See the *System Administration Guide* for more information about character sets and associated flags.

-z *language*
the name of an alternate language for displaying **extrjava** prompts and messages. Without the -z flag, extrjava uses the server's default language. You can add languages to an Adaptive Server during installation or afterward, using the **langinst** utility or the **sp_addlanguage** stored procedure.

-t *timeout*
specifies the number of seconds before a SQL command times out. If you do not specify a timeout, the command runs indefinitely. This affects commands issued from within **extrjava**, not the connection time. The default timeout for logging into **extrjava** is 60 seconds.

-v
prints the version number and copyright message for **extrjava** and then exits.

Examples

```
extrjava -j employees -f '/home/usera/jars/addr.jar' -
new
```

Downloads the classes associated with the employees JAR to the client file *newaddr.jar*.

Usage

- If the target client file already exists, **extrjava** overwrites its contents.
- The parameter flags **-f**, **-j**, **-S**, **-U**, **-P**, **-D**, and **–I** can be written with or without a space between the flag letter and the following parameter.
- When you execute **extrjava**, an exclusive lock is placed on *sysxtypes*.
- If **-jar** is specified, an exclusive table lock is placed on *sysjars*.

Permissions

You need to be a System Administrator or Database Owner to use **extrjava**.

See also

Commands	remove java
System procedures	sp_helpjava
System tables	*sysjars*, *sysxtypes*
Utilities	**instjava**

instjava

Description

Installs a JAR from a client file into an Adaptive Server.

Refer to *Java in Adaptive Server Enterprise* for more information about how this utility is used when Java is enabled in the database.

Syntax

```
instjava
-f file_name
 [ -new | -update ]
 [ -j  jar_name]
 [ -S server ]
 [ -U username ]
 [ -P password ]
 [ -D database ]
 [ -I interfaces_file ]
 [ -a display_charset ]
 [ -J client_charset ]
 [ -z language ]
 [ -t timeout] [-v]
```

Parameters

-f *file_name*
the name of the source file containing the classes to be installed in the database.

new | update
specifies whether the classes in the file already exist in the database.

If you specify the new parameter, you cannot install a class with the same name as an existing class.

If you specify the update parameter, you can install a class with the same name as an existing class, and the newly installed class replaces the existing class.

-j *jar_name*
the name of the JAR containing the classes to be installed in the database. Indicates that the JAR file should be saved in the database and associated with the classes it contains.

-S *server*
the name of the server.

-U *user_name*
an Adaptive Server login name. If you omit the **–U** flag and parameter, or if you specify the **–U** flag with no parameter, Adaptive Server uses the current user's operating system login name.

-P *password*
an Adaptive Server password. If you omit the **-P** flag and parameter, **instjava** prompts for a password. If you specify the **-P** flag with no password, the null password is used.

-D *database*
the name of the database in which to install the JAR. If you omit the **–D** flag, or if you specify the **–D** flag with no parameter, the user's default database is used.

-I *interfaces_file*
the name and location of the interfaces file to search when connecting to Adaptive Server. If you omit the **–I** flag and parameter, or if you specify the **–I** flag with no parameter, the interfaces file in the directory designated by your SYBASE environment variable is used.

-a *display_charset*
allows you to use **instjava** from a machine where the character set differs that of the server. Use **-a** in conjunction with **-J** to specify the character set translation file (*.xlt* file) required for the conversion. Use **-a** without **-J** only if the client character set is the same as the default character set.

-J *client_charset*
specifies the character set to use on the client. instjava uses a filter to convert input between client_charset and the Adaptive Server character set.

-J *client_charset* requests that Adaptive Server convert to and from client_charset, the character set used on the client.

-J with no argument disables character set conversion. Use this if the client and server use the same character set.

Omitting **-J** sets the character set to a default for the platform, which may not necessarily be the character set that the client is using. See the *System Administration Guide* for more information about character sets and associated flags.

-z *language*
the name of an alternate language for displaying instjava prompts and messages. Without the **-z** flag, **instjava** uses the server's default language. You can add languages to an Adaptive Server during installation or afterward, using the **langinst** utility or the **sp_addlanguage** stored procedure.

-t *timeout*
specifies the number of seconds before a SQL command times out. If you do not specify a timeout, the command runs indefinitely. This affects commands issued from within **instjava**, not the connection time. The default timeout for logging into **instjava** is 60 seconds.

-v
prints the version number and copyright message for **instjava** and then exits.

Examples

```
instjava -f '\home\usera\jars\addr.jar' -new
```

Installs **addr.jar** and its classes, but does not retain the association between the JAR and classes.

```
instjava -f '\home\usera\jars\addr.jar' -update
   -j employees
```

Reinstalls *addr.jar* and associates its classes with the employees JAR name.

Usage

- Any user can reference installed classes.
- The parameter flags **-f**, **-j**, **-S**, **-U**, **-P**, **-D**, and **-I** can be written with or without a space between the flag letter and the following parameter.

Adding New JARs

- If you use new with the **-jar** option and a JAR of that name already exists in the database, an exception is raised.
- If any classes of the same name as those in the source JAR already exist in the database, an exception is raised.

Updating JARs and Classes

Warning! If you alter a class used as a column datatype by reinstalling a modified version of the class, you must make sure that the modified class can read and use existing objects (rows) in tables using that class as a datatype. Otherwise, you may be unable to access those objects without reinstalling the class.

- If you use **-update** with the **-jar** option:
  - All classes in the database associated with the target JAR are deleted from the database and the classes in the source JAR file installed in their place.
  - If a class in the source JAR file is already installed in the database but is not attached to a JAR, the class in the source JAR is installed in the database and the unattached class is deleted.

- If you use **-update** without the **-jar** option:
  - Classes in the source JAR file replace unattached classes of the same name.
  - Classes in the source JAR that do not correspond to an installed class are installed as unattached classes in the database.

Locks

- When you execute **instjava**, an exclusive lock is placed on *sysxtypes*.
- If **-jar** is specified, an exclusive table lock is placed on *sysjars*.

Permissions

You need to be a System Administrator or Database Owner to use **instjava**.

See also

Commands	remove java
System procedures	sp_helpjava
System tables	*sysjars*, *sysxtypes*
Utilities	**extrjava**

isql

Description

Interactive SQL parser to Adaptive Server.

Syntax

isql [-b] [-e] [-F] [-n] [-p] [-X] [-Y]
[-a *display_charset*]
[-A *size*]
[-c *cmdend*]
[-D *database*]
[-E *editor*]
[-h *headers*]
[-H *hostname*]
[-i *inputfilename*]
[-I *interfaces_file*]
[-J *client_charset*]
[-l *login_timeout*]
[-m *errorlevel*]
[-o *outputfilename*]
[-P *password*]
[-R *remote_server_principal*]
[-s *colseparator*]
[-S *server*]
[-t *timeout*]
[-U *username*]
[-w *columnwidth*]
[-z *language*]
or

isql -v

To terminate a command: **go**

To clear the query buffer: **reset**

To call the default editor: **vi**

To execute an operating system command: **!! *command***

To exit from **isql**: **quit or exit**

Parameters

-b
disables the display of the table headers output.

-e
echoes input.

-F
enables the FIPS flagger. When you specify the **-F** parameter, Adaptive Server flags any non-standard SQL commands.

-n
removes numbering and the prompt symbol (>) from the echoed input lines in the output file when used in conjunction with **-e**.

-p
prints performance statistics.

-X
initiates the login connection to the server with client-side password encryption. **isql** (the client) specifies to the server that password encryption is desired. The server sends back an encryption key, which **isql** uses to encrypt your password, and the server uses the key to authenticate your password when it arrives.

-Y
tells the Adaptive Server to use chained transactions.

-a *display_charset*
runs **isql** from a terminal whose character set differs from that of the machine on which **isql** is running. Use **-a** in conjunction with **-J** to specify the character set translation file (*.xlt* file) required for the conversion. Use **-a** without **-J** only if the client character set is the same as the default character set.

-A *size*
specifies the network packet size to use for this **isql** session. For example:

```
isql -A 2048
```

sets the packet size to 2048 bytes for this **isql** session. To check your network packet size, enter:

```
select * from sysprocesses
```

The value is displayed under the *network_pktsz* heading.

size must be between the values of the **default network packet size** and **maximum network packet size** configuration variables, and must be a multiple of 512.

Use larger-than-default packet sizes to perform I/O-intensive operations, such as **readtext** or **writetext** operations.

Setting or changing Adaptive Server's packet size does not affect the packet size of remote procedure calls.

-c *cmdend*
resets the command terminator. By default, you terminate commands and send them to Adaptive Server by entering “go” on a line by itself. When you reset the command terminator, do not use SQL reserved words or control characters. Make sure to escape shell meta characters such as , ? ( ) [ ] $ and so on.

-D *database*
selects the database in which the **isql** session begins.

-E *editor*
specifies an editor other than your default editor.

-h *headers*
specifies the number of rows to print between column headings. The default prints headings only once for each set of query results.

-H *hostname*
sets the client host name.

-i *inputfilename*
specifies the name of the operating system file to use for input to **isql**. The file must contain command terminators (“go” is the default).

-I *interfaces_file*
specifies the name and location of the interfaces file to search when connecting to Adaptive Server. If you do not specify **-I**, **isql** looks for an interfaces file (*sql.ini* for Windows platforms) located in the *ini* directory which is below the directory specified by the SYBASE environment variable.

-J *client_charset*
specifies the character set to use on the client.
-J *client_charset* requests that Adaptive Server convert to and from *client_charset*, the character set used on the client. A filter converts input between *client_charset* and the Adaptive Server character set.

-J with no argument sets character set conversion to NULL. No conversion takes place. Use this if the client and server use the same character set.

Omitting **-J** sets the character set to a default for the platform. The default may not necessarily be the character set that the client is using.

-l *login_timeout*
specifies the maximum timeout value allowed when connecting to Adaptive Server.

-m *errorlevel*
customizes the error message display. For errors of the severity level specified or higher, only the message number, state, and error level are displayed; no error text appears. For error levels lower than the specified level, nothing appears.

-o *outputfilename*
specifies the name of an operating system file to store the output from **isql**.

-P *password*
specifies your current Adaptive Server password. This parameter is ignored if **-V** is used. Passwords are case sensitive and can be from 6 to 30 characters in length.

-R *remote_server_principal*
specifies the principal name for the server as defined to the security mechanism. By default, a server's principal name matches the server's network name (which is specified with the **-S** parameter or the DSQUERY environment variable). Use the **-R** parameter when the server's principal name and network name are not the same.

-s *colseparator*
resets the column separator character, which is blank by default. To use characters that have special meaning to the operating system (for example, |, ;, &, <, >), enclose them in quotes or precede them with a backslash.

-S *server*
specifies the name of the Adaptive Server to which to connect. Without **-S**, **isql** looks for the server specified by your DSQUERY environment variable.

-t *timeout*
specifies the number of seconds before a command times out. If you do not specify a timeout, the command runs indefinitely. This affects commands issued from within **isql**, not the connection time. The default timeout for logging into **isql** is 60 seconds.

-U *username*
specifies a login name. Logins are case sensitive.

-w *columnwidth*
sets the screen width for output. The default is 80 characters. When an output line reaches its maximum screen width, it breaks into multiple lines.

-z *language*
is the official name of an alternate language to display **isql** prompts and messages. Without **-z**, **isql** uses the server's default language. You can add languages to an Adaptive Server during installation, or you can add them afterward using the **langinst** utility or the **sp_addlanguage** stored procedure.

-v
prints the version and copyright message of the **isql** software that you are using.

Examples

```
isql
Password:
1> select *
2> from authors
3> where city = "Oakland"
4> go
```

Executes the command.

```
isql -Ujoe -Pabracadabra
1> select *
2> from authors
3> where city = "Oakland"
4> ed
```

Lets you edit the query. When you write and save the file, you are returned to **isql**. The query is displayed. Type **go** to execute it.

```
isql -Ualma
Password:
1> select *
2> from authors
3> where city = "Oakland"
4> reset
5> quit
```

reset clears the query buffer. **quit** returns you to the operating system.

Usage

- See Chapter 1, "Using the isql Utility" for details on **isql**.
- See the *Adaptive Server Reference Manual* for more information regarding **default network packet size** and **maximum network packet size** configuration parameters.
- The 5701 ("changed database") server message is no longer displayed after login or issuing a **use database** command.
- There are two new optional flags:

-b – disables column headers from printing
-D *database* – selects the startup database that **isql** uses

- Error message format differs from previous versions of **isql**. If you have scripts that perform routines based on the values of these messages you may need to re-write them.
- The **-y** *sybase_directory* parameter has been removed.
- To use **isql** interactively, give the command **isql** (and any of the optional parameters) at your operating system prompt. The **isql** program accepts SQL commands and sends them to Adaptive Server. The results are formatted and printed on standard output. Exit **isql** with **quit** or **exit.**
- Terminate a command by typing a line beginning with the default command terminator **go** or another command terminator, if the **-c** parameter is used. You can follow the command terminator with an integer to specify how many times to run the command. For example, to execute this command 100 times, type:

  ```
  select x = 1
  go 100
  ```

  The results display once at the end of execution.
- If you enter an option more than once on the command line, **isql** uses the last value. For example, if you enter the following command:

  ```
  isql -c. -csend
  ```

  "send", the second value for **-c**, overrides ".", the first value. This enables you to override any aliases you set up.
- To clear the existing query buffer, type **reset** on a line by itself. **isql** discards any pending input. You can also press Ctrl-c anywhere on a line to cancel the current query and return to the **isql** prompt.
- Case is significant for the **isql** flags.
- **isql** displays only 6 digits of float or real data after the decimal point, rounding off the remainder.
- When using **isql** interactively, read an operating system file into the command buffer with the command:

  ```
  :r filename
  ```

  Do not include a command terminator in the file; enter the terminator interactively once you have finished editing.

- You can include comments in a Transact-SQL statement submitted to Adaptive Server by **isql**. Open a comment with "/*". Close it with "*/", as shown in the following example:

```
select au_lname, au_fname
/*retrieve authors' last and first names*/
from authors, titles, titleauthor
where authors.au_id = titleauthor.au_id
and titles.title_id = titleauthor.title_id
/*this is a three-way join that links authors
**to the books they have written.*/
```

  If you want to comment out a **go** command, it should not be at the beginning of a line. For example:

```
/*
**go
*/
```

  should be used to comment out the **go** command instead of:

```
/*
go
*/
```

See also

Commands	create schema, set
Datatype	Exact Numeric Datatypes
System ESP	xp_sendmail
System procedures	sp_addlanguage, sp_addlogin, sp_addremotelogin, sp_add_resource_limit, sp_bindexeclass, sp_configure, sp_defaultlanguage, sp_droplanguage, sp_helplanguage, sp_processmail, sp_remoteoption, sp_serveroption, sp_showcontrolinfo, sp_unbindexeclass, sp_volchanged

langinst

Description

Installs one new language on Adaptive Server.

Syntax

```
langinst
 [-S server]
 [-I sqlini_file]
 [-P password]
 [-R release_number]
  language character_set
```

or

```
langinst -v
```

Parameters

-S *server*

specifies the name of the Adaptive Server to which to connect. If you do not specify **-S**, **langinst** uses the server specified by your DSQUERY environment variable. If DSQUERY is not set, **langinst** attempts to connect to a server named SYBASE.

-I *sqlini_file*

specifies the name and location of the interfaces file (*sql.ini*) that **langinst** searches when connecting to Adaptive Server. If you do not specify **-I**, **langinst** uses the *sql.ini* file in the *ini* subdirectory of your *%sybase%* release directory. If *%sybase%* is not set, **langinst** looks for the default Sybase installation directory, *d:\SYBASE*.

-P *password*

specifies the System Administrator's password. Only the System Administrator can run **langinst**. If you omit **-P**, **langinst** prompts for the System Administrator's password.

-R *release_number*

specifies the release number, in the format *n.n.n.*, to use to upgrade messages in *master..sysmessages*. Use **-R** only in failure conditions, such as if **langinst** fails or in case of user error, when you think that messages in *sysmessages* are out of date.

language

is the official name of the language to install. You must specify a language.

character_set

is the name of Adaptive Server's default character set. character_set indicates the directory name of the localization files for the language. The *common.loc* and *server.loc* localization files for an official language reside in the character set directory *%sybase%\locales\language\character_set*. You must specify a character set.

-v

prints the version number and copyright message for **langinst** and then exits.

Usage

- **setup** runs **langinst** automatically for both new installations and upgrades.
- **langinst** does the following:
  - Adds the language-specific information to *master..syslanguages* using the **sp_addlanguage** stored procedure. If the language already exists, **langinst** updates the appropriate row in *syslanguages*.
  - Adds to, updates, and deletes error messages, as necessary, from *master..sysmessages*.
  - Updates *syslanguages.update*, inserting the new release number.
- **langinst** validates the entries in the localization file sections that it uses. If anything is missing, **langinst** prints an error message and does not add the language to *syslanguages*.
- **langinst** compares the version numbers of each localization file it uses, *common.loc* and *server.loc*. If they are not the same, it prints a warning message. *syslanguages.upgrade* is always set according to the version number in *server.loc*.
- The **-R** parameter forces **langinst** to collect messages from a release previous to the current one. **langinst** compares the existing messages with the ones to be installed and replaces any that have changed.

  For example, if the current version is 12, and the previous version was 11.5, and you think *sysmessages* may not be correct, include the messages from the previous release in the *syslanguages.upgrade* column (11.5 in this case) by specifying **-R 11.5**. **langinst** then installs all messages from Adaptive Server 11.5.

Permissions

Only a System Administrator can run **langinst**.

Tables used

master.dbo.syslanguages, master.dbo.sysmessages

See also

System procedures	sp_addlanguage, sp_addlogin, sp_configure, sp_defaultlanguage, sp_droplanguage, sp_helplanguage

optdiag

Description

Displays optimizer statistics or loads updated statistics into system tables.

Syntax

```
optdiag [ binary ] [simulate ] statistics
 { -i input_file |
          database[.owner[.[table[.column]]]]
          [-o output_file] }
 [-U user_name]
 [-P password]
 [-I interfaces_file]
 [-S server]
 [-v]
 [-h]
 [-s]
 [-T flag_value]
 [-z language]
 [-J client_charset]
 [-a display_charset]
```

Parameters

binary

extracts statistics in human-readable form and in binary form. When used with an input file (**-i** *input_file*), loads binary statistics into system tables.

simulate

specifies that **optdiag** display or load simulated statistics. See the *Performance and Tuning Guide*.

-i *input_file*

specifies the name of the operating system file to use for **optdiag** input. Specifying an input file causes **optdiag** to update optimizer statistics for the table or column by using the values in the specified file. (Also called “input mode”).

database

is the name of the database whose statistics you want displayed. In input mode, **optdiag** uses the database name as specified in the file, and does not accept a database name from the command line.

owner

is the name of a table owner.

- In display mode, if you do not specify an owner, but do specify a table name, **optdiag** displays output for all of the owners of a table.
- In input mode, **optdiag** ignores the table owner specified on the command line and uses the value in the input file.

table
is the name of the table to survey for statistics.

- If the command does not include an owner name or a table name, **optdiag** displays statistics for all tables in the database.
- If the command includes an owner name, but no table name, **optdiag** displays all of the tables that belong to the specified owner.
- In input mode, **optdiag** ignores the table name specified on the command line and uses the value from the input file.

column
is the name of the colum to survey.

- If the command does not include a column name, **optdiag** displays all statistics for a table.
- In input mode, **optdiag** ignores the column name on the command line and uses the values from the input file.

-o *output_file*
specifies the name of an operating system file to store the output from **optdiag**. If a file with the same name already exists, **optdiag** overwrites that file without warning.

-U *user_name*
specifies an Adaptive Server login name.

-P *password*
specifies your Adaptive Server password. If you do not specify the **-P** flag, **optdiag** prompts for a password.

-I *interfaces_file*
specifies the name and location of the interfaces file to use when connecting to Adaptive Server.

If you do not use **-I** and specify an interfaces file name, **optdiag** looks for a file named *sql.ini* in the *ini* subdirectory in the Sybase installation directory (*d:\sybase*).

-S *server*
specifies the name of the Adaptive Server to which to connect. **isql** looks for this name up in the *sql.ini* file.

When you do not use **-S**, **optdiag** looks for the server that your DSQUERY environment variable specifies.

-a *display_charset*
runs **optdiag** from a terminal whose character set differs from that of the machine on which **optdiag** is running.

- Use **-a** in conjunction with **-J** to specify the character set translation (*.xlt*) file required for the conversion.
- Use **-a** without **-J** only if the client character set is the same as the default character set.

Note The ascii_7 character set is compatible with all character sets. If either the Adaptive Server character set or the client character set is set to ascii_7, any 7-bit ASCII character can pass unaltered between client and server. Any other characters produce conversion errors. For more information on character set conversion, see the *System Administration Guide*.

-z *language*
is the official name of an alternate language that the server uses both for date formats and to display **optdiag** prompts and messages. Without the **-z** flag, **optdiag** uses the server's default language.

You can add languages to Adaptive Server either during or after installation, After Adaptive Server installation, use either the **langinstall** utility or the **sp_addlanguage** stored procedure to add a language.

-J *client_charset*
specifies the character set to use on the client. A filter converts input between *client_charset* and the Adaptive Server character set.

- By using **-J** *client_charset*, you request that Adaptive Server convert data to and from *client_charset*, the client's character set.
- By using **-J** without a character set name, you specify character set conversion as NULL; no conversion takes place. Use **-J** alone when the client and server are using the same character set.
- By omitting **-J**, you set the character set to the default set for the platform. A filter converts input between the default set and the Adaptive Server character set. Keep in mind that the default may not necessarily be the character set that the client is using.

  For more information about character sets and their associated flags, see the *System Administration Guide*.

-v
displays the version number of and a copyright message for **optdiag** and exits.

-h
displays the **optdiag** syntax help.

-s
includes system tables in **optdiag** output. By default, only user tables are included.

-T*flag_value*
sets trace flags for the **optdiag** session. The **optdiag** trace flags are shown in Table 4-6.

Table 4-6: optdiag trace flags

Flag Value	Meaning
1	Do not stop with a warning if the optdiag version of Adaptive Server in use does not match the Adaptive Server version in the input file.
2	Display status message "Next table is *table_name*" when in input mode.
4	Skip consistency checking for step numbers while loading histograms in input mode.
6	Display lines of input file during input mode. This flag has no effect in display mode.

Note **optdiag** works only with single-byte character sets. If your server is using a multibyte character set, **optdiag** displays a warning message and exits.

Examples

```
optdiag statistics pubs2 -Usa -Ppasswd
-o pubs2.opt
```

Displays statistics for all user tables in the *pubs2* database and places the output in the *pubs2.opt* file.

```
optdiag statistics pubs2..titles -Usa -Ppasswd
   -o titles.opt
```

Displays statistics for the *titles* table.

```
optdiag statistics pubs2..titles -Usa -Ppasswd
   -o titles.opt -J roman8 -z french
```

Displays statistics using the roman8 character set and row labels and error messages in French.

```
optdiag binary statistics pubs2..titles.price
```

```
-Usa -Ppasswd -o price.opt
```

Displays binary statistics for the *price* column in the *titles* table.

```
optdiag statistics -i price.opt -Usa -Ppasswd
```

Loads edited statistics from the *price.opt* file.

Usage

- For more information about the **optdiag** command and an explanation of the **optdiag** output, see "Viewing Statistics with the optdiag Utility" in the *Performance and Tuning Guide*.
- For more information on changing statistics using **optdiag**, see "Changing Statistics with the optdiag Utility" in the *Performance and Tuning Guide*.
- By default, **optdiag** does not include the system tables when you display statistics for a database. To include the system tables in the output, use the **-s** flag.
- When you use **binary** mode, **optdiag** displays the human-readable values with comment marks (#s) at the beginning of the lines, as shown in this example:

```
Statistics for column:           "price"
Last update of column statistics: Jan 20 1998  7:16PM
 Statistics loaded from Optdiag.

  Range cell density:         0x3f8b9cfefece26bf
# Range cell density:         0.0134830400000000
  Total density:              0x3f8b9cfefece26bf
# Total density:              0.0134830400000000
  Range selectivity:          default used (0.33)
# Range selectivity:          default used (0.33)
  In between selectivity:     default used (0.25)
# In between selectivity:     default used (0.25)
```

- When you use **optdiag** with an input file to change statistics, it ignores all characters after the "#" in a line.
- Converting floating-point values may lead to rounding errors when you use files for input.

  When you are loading statistics on the same hardware platform, edit the statistics using the binary values to provide greater precision.

Byte Ordering and Binary *optdiag* Files

- Do not use the **binary** mode option to move statistics between Adaptive Servers on machines that use different byte ordering.

- On an incompatible architecture server, always comment out binary statistics and load the human-readable statistics.
- On a compatible architecture server,you can load either binary statistics or human-readable statistics.

Input Mode

- When you use the **-i** *input_file* syntax, **optdiag** reads the file as named and updates statistics in *sysstatistics*.
- **optdiag** input mode changes the configuration parameter **allow update to system tables** by setting the parameter to 1 at the beginning of the session, and then to 0 at the end of the session.
- During histogram input, the process checks the following rules and displays error messages for any violated rules:
  - The step numbers must increase monotonically, unless the command includes the **-T4** trace flag.
  - The column values for the steps must increase monotonically.
  - The weight for each cell must be between 0.0 and 1.0.
  - The total of weights for a column must be close to 1.0.
  - The first cell represents null values, and it must be present, even in columns that do not allow null values. There must be only one cell to represent the null value.
  - Two adjacent cells must not both use the **<** (less than) operator.

See also

Commands	create index, delete statistics, set, update statistics
System procedures	sp_addlogin, sp_configure, sp_defaultlanguage, sp_droplanguage, sp_flushstats, sp_helplanguage

sqlsrvr

Description

The executable form of the Adaptive Server program.

Syntax

```
sqlsrvr
 [-d devicename]
 [-e errorlogfile]
 [-m]
 [-r mastermirror]
 [-M sharedmem_directory]
 [-i interfaces_file_directory]
 [-s servername]
 [-p sso_login_name]
 [-g]
 [-G machine_name]
 [-C configuration_file_path]
```

or

```
sqlsrvr -v
```

Parameters

-d *devicename*
is the full path name of the *master* database device file. The *master* database device must be writable by the user who starts Adaptive Server. The default *master* database device name is *d_master*.

-e *errorlogfile*
is the full path name of the error log file for Adaptive Server system-level error messages.

-m
starts Adaptive Server in single-user mode.

-r *mastermirror*
starts the mirror of the *master* database device. Use this parameter to start Adaptive Server if the *master* database device has been damaged.

-M *sharedmem_directory*
places shared memory files in the specified directory instead of in the default location, *%sybase%*. If *sharedmem_directory* starts with "\", the directory name is assumed to be absolute. Otherwise, the directory name is interpreted relative to *%sybase%*.

-i *interfaces_file_directory*
specifies the directory location of the interfaces file to search when connecting to an Adaptive Server. If **-i** is omitted, **sqlsrvr** looks for a file named *sql.ini* in the *ini* subdirectory under the directory pointed to by your SYBASE environment variable.

-s *servername*
specifies the name of the Adaptive Server to be started. If **-s** is omitted, a server pointed to by the DSLISTEN environment variable is started.

-p *sso_login_name*
specifies the login name of a System Security Officer when starting Adaptive Server for the purpose of getting a new password for that account. Adaptive Server generates a random password, displays it, encrypts it, and saves it in *master..syslogins* as that account's new password.

-g
disables error logging to the Windows NT event log.

-G *machine_name*
specifies the name of the Windows NT event-logging PC that logs this server's error messages.

-C
starts Adaptive Server using the specified configuration file.

-v
prints the version number and copyright message and then exits.

Usage

- The **sqlsrvr** utility is referred to as **dataserver** in other Sybase documents.
- Start Adaptive Server using the **services manager** utility rather than by executing the **sqlsrvr** program directly. If you need to change any of the default parameters, edit the Adaptive Server's Registry keys. See the configuration guide for your platform for details.
- Adaptive Server derives its running environment from values in the *sysconfigures* system table. Run the system procedure **sp_configure** to see the configuration values; use **sp_configure** and **reconfigure** to change the configuration.
- Because Adaptive Server passwords are encrypted, you cannot recover forgotten passwords. If all System Security Officers lose their passwords, the **-p** parameter generates a new password for a System Security Officer's account. Start Adaptive Server with **-p**, immediately log into Adaptive Server with the new random password, and execute **sp_password** to reset your password to a more secure one.

- By default, Adaptive Server logs error messages in both the local error log file and the local Windows NT event log. You can disable Windows NT event logging by including the **-g** parameter and specifying a different event-logging machine with **-G***machine_name*. Use standard Windows NT conventions when entering the *machine_name*. For example, to designate a PC named "LOGSITE", substitute "\\LOGSITE" for the *machine_name*. See the configuration guide for your platform for details on logging error messages.
- After you have finished running the installer, set the file permissions on the **sqlsrvr** executable to limit who can execute it.
- Use either of the following methods to start Adaptive Server with a specified configuration file:
  - Use Server Config to configure the server to have the **-c** parameter. In the Configure Adaptive Server window, select the Command Line option, and in the Command Line Parameters window, enter the following information:

    ```
    -Cconfiguration_file_pathname
    ```

    For example, entering "-chaze.cfg" will start the server using the *haze.cfg* configuration file.
  - Start Adaptive Server from the command line and provide the **-c** parameter.

See also

Utility	**srvmgr**

srvmgr

Description

Starts, pauses, and stops Adaptive Server, Backup Server, and Adaptive Server Monitor™ as Windows NT services.

Syntax

srvmgr

Or double-click the Services Manager icon in the Sybase for Windows NT program group.

Applies to

The Services Manager window appears:

The controls in the services manager window are:

- Server – specifies the name of the server to start, stop, or pause.
- Services – specifies the type of server (Adaptive Server, Backup Server, or Adaptive Server Monitor) listed in the Server drop-down list box.
- Stop – shuts down the indicated server.
- Pause – pauses the indicated server by stopping future login attempts. Pause is valid only for Adaptive Server.
- Start / Continue – starts the indicated server as a Windows NT service, or resumes the normal operation of a paused server.

Usage

- The **services manager** utility was replaced by Sybase Central in Adaptive Server version 12.0. Use this utility only if you are upgrading to an older version of Adaptive Server.
- The **services manager** utility is also referred to as **startserver** in other Sybase documents.
- **services manager** starts servers as Windows NT services.
- You must have a *sql.ini* entry for a server to have the name appear in the Server drop-down list box.
- You must have the SYBASE environment variable set in order for **services manager** to locate the interfaces (*sql.ini*) files.
- Double-clicking the Stop control has the same effect as entering the Transact-SQL **shutdown** command; Adaptive Server waits for all currently executing transactions to complete before shutting down.
- Double-clicking the Pause control has no effect on users who are currently logged onto the server. It simply prohibits additional users from logging on.

- You can also start servers as automatic services. See the configuration guide for your platform for details.

Permissions

Any user with Windows NT administrator privileges and read permission can run **services manager** to start, pause, or stop servers.

See also

Utility commands	**bcksrvr**, **sqlsrvr**

wdllvers

Description

Provides the following information about the Sybase DLLs (dynamic link libraries) loaded into memory:

- DLL name
- DLL usage count
- DLL file size
- DLL file date
- Directory path from which the DLL was loaded

The wdllvers executable (*wdllvers.exe*) is located in the *bin* directory of the Sybase release directory.

Syntax

wdllvers

Or double-click the WDLLVERS icon in the Sybase for Windows program group.

Applies to

The WDLLVERS - Sybase DLL Manager window appears:

File Menu Options

Using the File menu options you can save and print the currently displayed DLL information and load and unload DLLs from memory. The options are:

- Save Output – saves the currently displayed DLL information to a text file.
- Print Output – sends the currently displayed DLL information to a printer.
- Load Module – tests whether a DLL will load properly into memory. You can use this option as a debugging tool to determine why a DLL cannot be loaded.
- Unload Module – decrements the usage count for this DLL by 1 each time you choose this option. When the usage count is 0, choosing this option unloads the DLL from memory. Or you can click the Unload Module button.

  Warning! Unloading a DLL may cause Windows or another application to terminate unexpectedly, if the DLL you are unloading is needed by the application or by Windows.

- Unload Module All – reduces a DLL's usage count to 0 and unloads the DLL from memory in one step (as opposed to the Unload Module option, which reduces the usage count one by one). Use this option to unload a DLL quickly.

- Exit – exits the **wdllvers** utility.

View Menu Options

The View menu displays a list of DLLs loaded into memory:

- All Modules Loaded – displays a list of all DLLs loaded into memory on this PC.
- Sybase DLL Loaded – displays a list of all Sybase DLLs loaded into memory on this PC.

Refresh Menu Options

Using the Refresh menu options, you can update the displayed DLL information by taking a new "snapshot" of the PC's memory, either when you request it or automatically, at specified intervals.

- Refresh – updates the displayed DLL information by taking another snapshot of the PC's memory. The Refresh button performs the same function.
- Auto Refresh – automatically updates the displayed DLL information by taking snapshots of the PC's memory at intervals set by the Set Auto Refresh Rate option.
- Set Auto Refresh Rate – sets the length of the interval used by the Auto Refresh option to determine when to update the DLL information.

Help Menu Option

Provides version information for the **wdllvers** utility.

Permissions

No special permissions are required to run wdllvers.

Index

Symbols

A

B

C

D

E

F

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W