SYBASE®

Historical Server Users Guide

Adaptive Server Enterprise Monitor™

15.0.2

DOCUMENT ID: DC36556-01-1502-01

LAST REVISED: November 2008

Sybase, Inc., One Sybase Drive, Dublin, CA 94568.

Contents

About This Book

Audience

This book is for people responsible for:

- Configuring and managing Historical Server
- Using Historical Server to monitor Adaptive Server performance

How to use this book

This book contains the following chapters:

- Chapter 1, "Introduction" provides an overview of Adaptive Server Enterprise Monitor and presents some basic Historical Server concepts.
- Chapter 2, "Configuring Historical Server" describes how to complete an initial Historical Server configuration or change a configuration, for both UNIX and Windows platforms.
- Chapter 3, "Starting and Stopping Historical Server" describes how to start and stop Historical Server on both UNIX and Windows platforms.
- Chapter 4, "Command Reference" describes the isql command interface to Historical Server.
- Chapter 5, "Data Files and Output Options" describes the files that Historical Server creates and how to access the data in them, including how to use the bcp utility to load the data into Adaptive Server tables.
- Appendix A, "Data Items" provides descriptions of the data items available through Historical Server.
- Appendix B, "Specifications for Defining Recording Session Views" describes valid combinations of data item names and statistic types for defining recording session views.
- Appendix C, "Specifications for Defining Playback Views" describes valid combinations of data item names and statistic types for defining playback session views.
- Appendix D, "Examples of Recording Session Views" provides examples of recording session views.

Related documents

The Adaptive Server® Enterprise documentation set consists of the following:

- 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 the release bulletin may be available on the World Wide Web. To check for critical product or document information that was added after the release of the product CD, use the Sybase Technical Library.

- The *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 15.0, the system changes added to support those features, and changes that may affect your existing applications.

- *ASE Replicator User's Guide* – describes how to use the Adaptive Server Replicator feature of Adaptive Server to implement basic replication from a primary server to one or more remote Adaptive Servers.

- *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.

- The *Configuration Guide* for your platform – provides instructions for performing specific configuration tasks for Adaptive Server.

- *Enhanced Full-Text Search Specialty Data Store User's Guide* – describes how to use the Full-Text Search feature with Verity to search Adaptive Server Enterprise data.

- *Glossary* – defines technical terms used in the Adaptive Server documentation.

- *Historical Server User's Guide* – describes how to use Historical Server to obtain performance information for SQL Server® and Adaptive Server.

- *Java in Adaptive Server Enterprise* – describes how to install and use Java classes as datatypes, functions, and stored procedures in the Adaptive Server database.

- *Job Scheduler User's Guide* – provides instructions on how to install and configure, and create and schedule jobs on a local or remote Adaptive Server using the command line or a graphical user interface (GUI).

- *Messaging Service User's Guide* – describes how to useReal Time Messaging Services to integrate TIBCO Java Message Service and IBM WebSphere MQ messaging services with all Adaptive Server database applications.
- *Monitor Client Library Programmer's Guide* – describes how to write Monitor Client Library applications that access Adaptive Server performance data.
- *Monitor Server User's Guide* – describes how to use Monitor Server to obtain performance statistics from SQL Server and Adaptive Server.
- *Performance and Tuning Series* – a series of books that explain how to tune Adaptive Server for maximum performance:
  - *Basics* – the basics for understanding and investigating performance questions in Adaptive Server.
  - *Locking and Concurrency Control* – describes how the various locking schemas can be used for improving performance in Adaptive Server, and how to select indexes to minimize concurrency.
  - *Query Processing and Abstract Plans* – describes how the optimizer processes queries and how abstract plans can be used to change some of the optimizer plans.
  - *Physical Database Tuning* – describes how to manage physical data placement, space allocated for data, and the temporary databases.
  - *Monitoring Adaptive Server with sp_sysmon* – describes how to monitor Adaptive Server's performance with sp_sysmon.
  - *Improving Performance with Statistical Analysis* – describes how Adaptive Server stores and displays statistics, and how to use the set statistics command to analyze server statistics.
  - *Using the Monitoring Tables* – describes how to query Adaptive Server's monitoring tables for statistical and diagnostic information.
- *Quick Reference Guide* – provides a comprehensive listing of the names and syntax for commands, functions, system procedures, extended system procedures, data types, and utilities in a pocket-sized book (regular size when viewed in PDF format).
- *Reference Manual* – is a series of four books that contains the following detailed Transact-SQL information:
  - *Building Blocks* – Transact-SQL datatypes, functions, global variables, expressions, identifiers and wildcards, and reserved words.

    - *Commands* – Transact-SQL commands.
    - *Procedures* – Transact-SQL system procedures, catalog stored procedures, system extended stored procedures, and dbcc stored procedures.
    - *Tables* – Transact-SQL system tables and dbcc tables.
- *System Administration Guide* –
    - *Volume 1* – provides an introduction to the basics of system administration, including a description of configuration parameters, resource issues, character sets, sort orders, and diagnosing system problems. The second part of this book is an in-depth description of security administration.
    - *Volume 2* – includes instructions and guidelines for managing physical resources, mirroring devices, configuring memory and data caches, managing multiprocessor servers and user databases, mounting and unmounting databases, creating and using segments, using the reorg command, and checking database consistency. The second half of this book describes how to back up and restore system and user databases.
- *System Tables Diagram* – illustrates system tables and their entity relationships in a poster format. Full-size available only in print version; a compact version is available in PDF format.
- *Transact-SQL User's Guide* – documents Transact-SQL, the Sybase 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.
- *Troubleshooting Series* (for release 15.0):
    - *Troubleshooting: Error Messages Advanced Resolutions* – contains troubleshooting procedures for problems that you may encounter when using Sybase® Adaptive Server® Enterprise. The problems addressed here are those which the Sybase Technical Support staff hear about most often

- *Troubleshooting and Error Messages Guide* – contains detailed instructions on how to resolve the most frequently occurring Adaptive Server error messages. Most of the messages presented here contain error numbers (from the master..sysmessages table), but some error messages do not have error numbers, and occur only in Adaptive Server's error log.

- *User Guide for Encrypted Columns* – describes how configure and use encrypted columns with Adaptive Server
- *Using Adaptive Server Distributed Transaction Management Features* – explains how to configure, use, and troubleshoot Adaptive Server DTM features in distributed transaction processing environments.
- *Using Sybase Failover in a High Availability System* – provides instructions for using Sybase Failover to configure an Adaptive Server as a companion server in a high availability system.
- *Unified Agent and Agent Management Console* – describes the Unified Agent, which provides runtime services to manage, monitor and control distributed Sybase resources.
- *Utility Guide* – documents the Adaptive Server utility programs, such as isql and bcp, which are executed at the operating system level.
- *Web Services User's Guide* – explains how to configure, use, and troubleshoot Web Services for Adaptive Server.
- *XA Interface Integration Guide for CICS, Encina, and TUXEDO* – provides instructions for using the Sybase DTM XA interface with X/Open XA transaction managers.
- *XML Services in Adaptive Server Enterprise* – describes the Sybase native XML processor and the Sybase Java-based XML support, introduces XML in the database, and documents the query and mapping functions that comprise XML Services.

Other sources of information

Use the Sybase Getting Started CD, the SyBooks CD, and the Sybase Product Manuals Web site to learn more about your product:

- The Getting Started CD contains release bulletins and installation guides in PDF format, and may also contain other documents or updated information not included on the SyBooks CD. It is included with your software. To read or print documents on the Getting Started CD, you need Adobe Acrobat Reader, which you can download at no charge from the Adobe Web site using a link provided on the CD.

- The SyBooks CD contains product manuals and is included with your software. The Eclipse-based SyBooks browser allows you to access the manuals in an easy-to-use, HTML-based format.

  Some documentation may be provided in PDF format, which you can access through the PDF directory on the SyBooks CD. To read or print the PDF files, you need Adobe Acrobat Reader.

  Refer to the *SyBooks Installation Guide* on the Getting Started CD, or the *README.txt* file on the SyBooks CD for instructions on installing and starting SyBooks.

- The Sybase Product Manuals Web site is an online version of the SyBooks CD that you can access using a standard Web browser. In addition to product manuals, you will find links to EBFs/Maintenance, Technical Documents, Case Management, Solved Cases, newsgroups, and the Sybase Developer Network.

  To access the Sybase Product Manuals Web site, go to Product Manuals at http://www.sybase.com/support/manuals/.

Sybase certifications on the Web

Technical documentation at the Sybase Web site is updated frequently.

❖ **Finding the latest information on product certifications**

1. Point your Web browser to Technical Documents at http://www.sybase.com/support/techdocs/.
2. Click Certification Report.
3. In the Certification Report filter select a product, platform, and timeframe and then click Go.
4. Click a Certification Report title to display the report.

❖ **Finding the latest information on component certifications**

1. Point your Web browser to Availability and Certification Reports at http://certification.sybase.com/.
2. Either select the product family and product under Search by Base Product; or select the platform and product under Search by Platform.
3. Select Search to display the availability and certification report for the selection.

❖ **Creating a personalized view of the Sybase Web site (including support pages)**

Set up a MySybase profile. MySybase is a free service that allows you to create a personalized view of Sybase Web pages.

1 Point your Web browser to Technical Documents at http://www.sybase.com/support/techdocs/.

2 Click MySybase and create a MySybase profile.

Sybase EBFs and software maintenance

❖ **Finding the latest information on EBFs and software maintenance**

1 Point your Web browser to the Sybase Support Page at http://www.sybase.com/support.

2 Select EBFs/Maintenance. If prompted, enter your MySybase user name and password.

3 Select a product.

4 Specify a time frame and click Go. A list of EBF/Maintenance releases is displayed.

Padlock icons indicate that you do not have download authorization for certain EBF/Maintenance releases because you are not registered as a Technical Support Contact. If you have not registered, but have valid information provided by your Sybase representative or through your support contract, click Edit Roles to add the "Technical Support Contact" role to your MySybase profile.

5 Click the Info icon to display the EBF/Maintenance report, or click the product description to download the software.

Conventions

The following sections describe conventions used in this manual.

SQL is a free-form language. There are no rules about the number of words you can put on a line or where you must break a line. However, for readability, all examples and most syntax statements in this manual are formatted so that each clause of a statement begins on a new line. Clauses that have more than one part extend to additional lines, which are indented. Complex commands are formatted using modified Backus Naur Form (BNF) notation.

Table 1 shows the conventions for syntax statements that appear in this manual:

Table 1: Font and syntax conventions for this manual

Element	Example
Command names,procedure names, utility names, and other keywords display in sans serif font.	select sp_configure
Database names and datatypes are in sans serif font.	master database
Book names, file names, variables, and path names are in italics.	*System Administration Guide* *sql.ini* file *column_name* *$SYBASE/ASE* directory

Variables—or words that stand for values that you fill in—when they are part of a query or statement, are in italics in Courier font.	`select column_name`
	`from table_name`
	`where search_conditions`

Type parentheses as part of the command.	compute *row_aggregate* (*column_name*)
Double colon, equals sign indicates that the syntax is written in BNF notation. Do not type this symbol. Indicates "is defined as".	`::=`
Curly braces mean that you must choose at least one of the enclosed options. Do not type the braces.	`{cash, check, credit}`
Brackets mean that to choose one or more of the enclosed options is optional. Do not type the brackets.	`[cash | check | credit]`
The comma means you may choose as many of the options shown as you want. Separate your choices with commas as part of the command.	`cash, check, credit`
The pipe or vertical bar(|) means you may select only one of the options shown.	`cash | check | credit`
An ellipsis (...) means that you can *repeat* the last unit as many times as you like.	`buy thing = price [cash | check | credit] [, thing = price [cash | check | credit]]...` You must buy at least one thing and give its price. You may choose a method of payment: one of the items enclosed in square brackets. You may also choose to buy additional things: as many of them as you like. For each thing you buy, give its name, its price, and (optionally) a method of payment.

- Syntax statements (displaying the syntax and all options for a command) appear as follows:

  sp_dropdevice [*device_name*]

  For a command with more options:

  select *column_name*
  from *table_name*
  where *search_conditions*

In syntax statements, keywords (commands) are in normal font and identifiers are in lowercase. Italic font shows user-supplied words.

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

```
select * from publishers
```

- Examples of output from the computer appear as follows:

```
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)
```

In this manual, most of the examples are in lowercase. However, you can disregard case when typing Transact-SQL keywords. For example, SELECT, Select, and select are the same.

Adaptive Server's sensitivity to the case of database objects, such as table names, depends on the sort order installed on Adaptive Server. You can change case sensitivity for single-byte character sets by reconfiguring the Adaptive Server sort order. For more information, see the *System Administration Guide*.

Accessibility features

This document is available in an HTML version that is specialized for accessibility. You can navigate the HTML with an adaptive technology such as a screen reader, or view it with a screen enlarger.

Adaptive Server HTML documentation has been tested for compliance with U.S. government Section 508 Accessibility requirements. Documents that comply with Section 508 generally also meet non-U.S. accessibility guidelines, such as the World Wide Web Consortium (W3C) guidelines for Web sites.

Note You might need to configure your accessibility tool for optimal use. Some screen readers pronounce text based on its case; for example, they pronounce ALL UPPERCASE TEXT as initials, and MixedCase Text as words. You might find it helpful to configure your tool to announce syntax conventions. Consult the documentation for your tool.

For information about how Sybase supports accessibility, see Sybase Accessibility at http://www.sybase.com/accessibility. The Sybase Accessibility site includes links to information on Section 508 and W3C standards.

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

Introduction

Description of Adaptive Server Enterprise Monitor

Adaptive Server Enterprise Monitor™ (Adaptive Server Monitor) provides a way to monitor Adaptive Server performance in real time or in a historical data-gathering mode. System administrators can use this information to identify potential resource bottlenecks, to research current problems, and to tune for better performance. Adaptive Server Monitor provides feedback for tuning at several levels:

- Adaptive Server configuration
- Database design
- SQL statements in applications and stored procedures

Adaptive Server Monitor components

Adaptive Server Monitor consists of four components that gather or display Adaptive Server performance data:

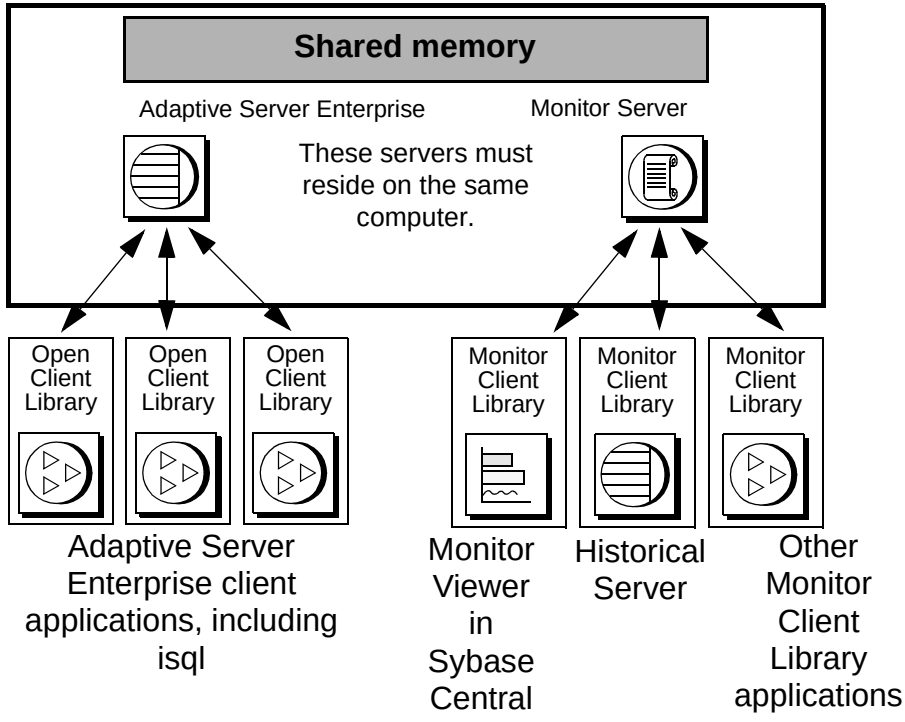
- Adaptive Server Enterprise Monitor Server (Monitor Server) – this server collects Adaptive Server performance data in real time and makes the data available to the other Adaptive Server Monitor components. Monitor Server is a Sybase Open Server application.
- Adaptive Server Enterprise Monitor Historical Server (Historical Server) – this server obtains Adaptive Server performance data from Monitor Server and saves the data in files for deferred analysis. Historical Server is a Sybase Open Server application.

- Monitors in the Adaptive Server Enterprise plug-in for Sybase Central™ (Monitor Viewer) – the monitors obtain Adaptive Server performance data from Monitor Server and display the data in real time in tables and graphs.
- Adaptive Server Enterprise Monitor Client Library (Monitor Client Library) – this application programming interface (API) to Monitor Server and Historical Server is available to users for developing monitoring applications. Historical Server and the monitors in the Adaptive Server Enterprise plug-in for Sybase Central are Monitor Client Library applications.

Adaptive Server Monitor architecture

Figure 1-1 shows the relationships between Adaptive Server and the various components of Adaptive Server Monitor.

Figure 1-1: Adaptive Server and Adaptive Server Monitor components



Adaptive Server Enterprise Monitor architecture

Adaptive Server saves performance data in a shared memory area that Monitor Server reads. Because of this shared memory technique, Monitor Server must be installed and running on the same machine as the Adaptive Server being monitored. A one-to-one relationship exists between an Adaptive Server and a Monitor Server. For more information about Monitor Server, see the *Sybase Adaptive Server Enterprise Monitor Server User's Guide*.

Monitor Client Library applications obtain Adaptive Server performance statistics from Monitor Server. These applications are clients of Monitor Server. For performance reasons, Sybase recommends running Monitor Client Library applications on machines other than the ones where pairs of Adaptive Server and Monitor Server are running. See the *Sybase Adaptive Server Enterprise Monitor Client Library Programmer's Guide* for more information.

The Adaptive Server Enterprise plug-in for Sybase Central includes a set of monitors showing different aspects of Adaptive Server resource usage at various levels of detail. Each open monitor is a separate application, with a unique client connection to Monitor Server. In Sybase Central, each Adaptive Server installation has its own Monitors folder containing the set of monitor objects.

Historical Server collects performance information from Monitor Server and saves the information in files for deferred analysis. Historical Server interfaces let users specify the data to collect and the time period desired. They also include a historical data playback feature. The interfaces are:

- A command interface in isql. For more information, see the *Sybase Adaptive Server Enterprise Monitor Historical Server User's Guide*.
- A programming interface using Monitor Client Library. For more information, see *Sybase Adaptive Server Enterprise Monitor Client Library Programmer's Guide*.

Historical Server concepts

This section describes the following Historical Server concepts:

- Recording sessions
- Playback sessions
- Views
- Data items and statistic types

Recording sessions

Recording sessions gather Adaptive Server performance data and store it in files for later analysis. Some attributes of a recording session are:

- Monitor Server name – by association, this defines the Adaptive Server whose performance you are monitoring.
- Sample interval – this attribute defines how often to collect performance data.
- Views, alarms, and filters – views and filters define the data you want to collect. Alarms define actions that can occur when a specified data item hits a predefined threshold value.
- Start time and end time – these specifications define the time period during which you want to collect the data.

To create a recording session, use a sequence of commands in the Historical Server isql command interface:

- hs_create_recording_session
- hs_create_view
- hs_create_filter (optional)
- hs_create_alarm (optional)
- hs_initiate_recording

When you create a recording session, Historical Server assigns it a session ID. You can list the session IDs of defined recording sessions using the hs_list command. hs_list can also show the complete recording session definition, including view names and the data items, alarms, and filters in the view.

Historical Server stores these recording session definitions in its control file, which resides in the Historical Server home directory. Therefore, hs_list can see only recording session definitions that were created by Historical Server instances using the same home directory that the current Historical Server is using. See "Configuring multiple instances of Historical Server" on page 23 for more information on configuring Historical Server home directories.

To examine the data gathered by a recording session, you can:

- Populate Adaptive Server tables with the data from recording sessions by using the Sybase bulk copy (bcp) utility. See "Bulk copy example" on page 97 for more information.
- Initiate a Historical Server playback session.

Playback sessions

Playback sessions let you retrieve the data gathered during one or more recording sessions. You can play back data in two forms:

- Playback to a client – the results of the playback are sent to the user, who can view the results on the terminal or redirect them to a file.
- Playback to a file – the results of the playback are stored in a file. The resulting files are essentially a new recording session. You can use these files as input to yet another playback session, or as input to the bcp utility to populate Adaptive Server tables, or any other way that you would use recording session files.

The following attributes define a playback session:

- Input recording sessions – the input to a playback session is one or more recording sessions.
- Views, start time, and end time – these attributes define the data from the input recording sessions that you want to include in the playback session.
- Summarization level – you can specify raw playback, which shows you exactly what was recorded, or you can specify various summarization levels.

To create a playback session, use the following sequence of commands from the Historical Server command interface:

- hs_create_playback_session
- hs_create_playback_view
- hs_initiate_playback
- hs_playback_sample (used only for playback to a client)
- hs_terminate_playback

Views

A recording session view defines the performance data you want Historical Server to gather. A playback session view defines which performance data from a recording session view you want Historical Server to play back.

A view consists of a view name and one or more data items. Each data has a statistic type associated with it. See “Data items and statistic types” on page 6 for more information.

When you define a recording session, you define one or more views to be included in that recording session. A recording session must have at least one view. For more information about creating recording session views, see hs_create_view on page 71.

Appendix D, "Examples of Recording Session Views" contains many sample recording session views.

When you define a playback session, you define which views in the previously defined recording sessions should be included in the playback session. The playback session view names must be the same as the names used for the recording sessions views. You can include all data items or a subset of the data items from the recording session view in the corresponding playback view. For more information about creating playback views, see hs_create_playback_view on page 67.

Data items and statistic types

A **data item** identifies specific information that you want to include in the view. If a data item includes embedded spaces, you must surround the name with quotation marks when you use it. Some sample data items are: Page I/O, Login Name, and CPU Time.

Table A-1 on page 107 lists all available data items and describes each one.

Each data item has a **statistic type** associated with it. The statistic type defines the duration of the data item (sample or session) and whether Historical Server performs calculations on the data item.

The statistic types contain embedded spaces. You must surround them with quotation marks when you use them in the Historical Server commands.

Not all statistic types are valid with all of the data items. Table B-3 on page 152 shows valid statistic types for each data item.

The six statistic types are:

- "Value for Sample" – this statistic type returns a count of activity or some type of information that applies to the most recent sample interval. It implies no calculations.
    - Activity Counts – for data items that represent activity counts, this statistic type returns the number of occurrences of an activity during the most recent sample interval. For example, Value for Sample for Page I/O is the number of page I/Os that occurred during the most recent sample interval.

- Other information – this is the only statistic type valid for data items that represent character strings. For example, Value for Sample for Object Name returns the name of a table. This statistic type is also the only one valid for data items that represent values such as IDs and values for configured parameters, such as Process ID and Code Memory Size.

- "Value for Session" – this statistic type returns a cumulative count of activity since the start of gathering the data (since the connection was opened). No calculations are performed. For example, Value for Session for Page I/O is the number of page I/Os that occurred since the recording session started.

- "Rate for Sample" – this statistic type calculates a rate per second. It returns the average number of occurrences per second of an activity during the most recent sample interval. For example, Rate for Sample for Page I/O is the average number of page I/Os that occurred each second during the most recent sample interval.

  The calculation is:

  $$\frac{\text{Count for the most recent sample interval}}{\text{Number of seconds in the sample interval}}$$

- "Rate for Session" – this statistic type calculates a rate per second. It returns the average number of occurrences per second of an activity during the current recording session. For example, Rate for Session for Page I/O is the average number of page I/Os that occurred per second since the recording session started.

  The calculation is:

  $$\frac{\text{Count for the session}}{\text{Number of seconds in the session}}$$

- "Average for Sample" – this statistic type calculates an average value per occurrence of an activity over the most recent sample interval. Only a few data items can use this statistic type. The meaning of the returned value depends on the data item. For example, Average for Sample for Procedure Elapsed Time is the average execution time per execution of a stored procedure during the most recent sample interval.

- "Average for Session" – this statistic type calculates an average value per occurrence of an activity over the session. Only a few data items can use this statistic type. The meaning of the returned value depends on the data item. For example, Average for Session for Procedure Elapsed Time is the average execution time per execution of a stored procedure during the recording session.

Table B-3 on page 152 shows valid combinations of data items and statistic types.

CHAPTER 2

Configuring Historical Server

Historical Server configuration concepts

This section describes concepts that you should understand before configuring Historical Server including:

- The Historical Server control file and home directory
- The operating system start-up account
- The Historical Server superuser account

The Historical Server control file and home directory

The Historical Server control file maintains information about recording sessions that users create. This information persists across start-ups, so users can access recording sessions that they created during previous executions of Historical Server. The control file restricts user access to private recording session files. (Recording session files can have public or private access.)

The Historical Server home directory is important for two reasons:

- It contains the Historical Server control file. When Historical Server starts, it looks for a control file in the home directory. If a control file does not exist, Historical Server creates it.

- It is the default directory location for the data files that Historical Server writes during recording sessions. The users who create the recording sessions can override this default location.

The -D parameter in the Historical Server start-up command specifies the home directory location. This is a required parameter.

The current execution of Historical Server can access data files from previous executions only if the current execution is using the same control file, in the same home directory, as the previous executions were using. Therefore, in most cases, you should not change the Historical Server home directory between start-ups. See "Configuring multiple instances of Historical Server" on page 23 for a discussion about using different home directories.

For more information about the files created by Historical Server, see Chapter 5, "Data Files and Output Options."

Accessing control file information

Use the Historical Server hs_list command to gain access to the information in the Historical Server control file.

Do not edit the control file

Do not edit the control file. You might inadvertently corrupt it. Regardless of the editor you use, do not open and then save this file.

This is true especially if Historical Server is running on Windows. Unlike the other files created by Historical Server, the control file is not a standard-format Windows text file. Lines of text in the control file are terminated only with newline characters, rather than the usual carriage-return/newline pairs. The editing program may corrupt the file by embedding unwanted carriage-return/newline pairs into the text.

The operating system start-up account

The account that starts Historical Server must satisfy these conditions:

- The same account must start Historical Server each time.

The operating system account that starts Historical Server the first time, when Historical Server creates the control file, is the only account that has subsequent access to that control file. The same account must perform all subsequent start-ups of Historical Server using the same home directory, to gain access to the control file. This restriction prevents unauthorized reading and modification of Historical Server files.

A different account can start Historical Server if the start-up command specifies a different home directory. In the new location, if no control exists, Historical Server creates one. The recording session data files in a home directory are not visible to a Historical Server using a different home directory.

- The account must have search (execute) and write access to the Historical Server home directory specified in the start-up command.
- The account must have search (execute) and write access to the locations of recording session data files, as specified by users who create recording sessions. The default location is the Historical Server home directory, but users can override that default on a session-by-session basis when they create recording sessions.

The Historical Server superuser account

The -U parameter in the Historical Server start-up command can optionally specify a superuser for Historical Server. If this parameter is specified, the superuser can:

- Shut down Historical Server
- View or delete any historical data files

If the start-up command does not include the -U parameter, any user may stop Historical Server, but no user has unrestricted access to historical data files.

Sybase Open Client/Server connections

Historical Server establishes client/server connections using either *interfaces* files (the *interfaces* file on UNIX; *sql.ini* file on Windows) or a directory service, as supported by Sybase Open Client/Server version 11.1.x.

The advantage of using a directory service is that you do not need to update the *interfaces* or *sql.ini* files on all of the client machines. A single directory service entry replaces these files. Changes such as moving a server to a new address or changing the server name are easier to administer.

See *Open Client/Server Configuration Guide for UNIX* or *Open Client/Server Configuration Guide for Desktop Platforms* for more information.

Initial configuration on UNIX platforms

This section describes how to configure Historical Server on UNIX platforms.

Assumptions on UNIX platforms

The instructions in this section are based on the following assumptions:

- Historical Server software was unloaded from the delivery media using the instructions provided with the delivery media.
- An Adaptive Server/Monitor Server pair is configured on your network.

Configuration procedures on UNIX platforms

To configure Historical Server on a UNIX platform:

1 Set the $SYBASE environment variable to the value of the Sybase installation directory where you unloaded Historical Server.

2 Log on using the "sybase" account or another account that has read, write, and search (execute) permissions on the $SYBASE directory.

3 Set the $PATH environment variable.

The Historical Server executable resides in *$SYBASE/bin.* Add this path name to the $PATH environment variable for the account that will start Historical Server.

4 Create a script file for Historical Server start-up.

A script file ensures that correct parameters are used for each Historical Server start-up. The script file contains the Historical Server start-up command, histserver, and its parameters.

To create a Historical Server script file:

a Using any editor, create a new file. The recommended name and location for the new file is:

```
install_dir/install/run_histServerName,
```

where *histServerName* is the name of the Historical Server.

b Edit the new file, inserting the histserver command and supplying parameters and values appropriate to your installation. Do *not* use carriage returns within the command; use the UNIX continuation character (\) to continue the command on multiple lines. Spaces between a parameter and its value are optional.

Table 2-1 on page 21 describes the command and its parameters. The parameters marked "required" in the table must appear in the script file. The ones with default values may be omitted if the default values are acceptable.

A sample script file for starting Historical Server follows:

```
histserver -Dserver1HistDir -Sserver1Hist  \
-Usa -PsaPasswd                            \
-lserver1HistLog -n15 &
```

c Use the chmod command to give the account that will start Historical Server execute permission on the script file.

5 Add connectivity information for Historical Server.

This task assigns a port or network address to Historical Server. It also ensures that Historical Server can connect to one or more Adaptive Server/Monitor Server pairs.

Add Historical Server connection information either to the *interfaces* files or to a directory service. See *Open Client/Server Configuration Guide for UNIX* for information on dsedit, dscp, dsedit_dce, and dscp_dce.

If you are relying on *interfaces* files for making client/server connections:

a Check the server listings in the *interfaces* file used by Historical Server. For Historical Server to run, this file must contain entries for *all* of the following servers:

- Any Adaptive Server you want to monitor

- A Monitor Server paired with each Adaptive Server
- Historical Server

b Use either dsedit (if your system is running X-Windows) or dscp (a command line utility) to add entries to an *interfaces* file. Follow the instructions in *Open Client/Server Configuration Guide for UNIX*. To add these entries, you must know the:

- Monitor Server and Adaptive Server names to which you want Historical Server to connect.
- Port numbers or network addresses assigned to these servers when they were configured. To research this information, use dsedit or dscp on the machine where a server was configured to examine the appropriate *interfaces* file.

If you are relying on a directory service for making client/server connections:

- Make sure that the *libtcl.cfg* file on the machine where Historical Server was installed points to the appropriate directory service. Use an editor to check and update *libtcl.cfg* files.
- Add Historical Server to the directory service, using dsedit_dce (if your system is running X-Windows) or dscp_dce (a command line utility). You need to know the Historical Server name to complete this step.

6 Configure Historical Server on client machines.

This task enables clients to connect to Historical Server. Historical Server clients are users who create recording sessions or playback sessions. Each client machine must be configured appropriately.

See *Open Client/Server Configuration Guide for UNIX* for instructions on using dsedit and dscp.

If you are relying on *interfaces* files for making client/server connections:

a Update *all* of the *interfaces* files used by Historical Server clients. The client *interfaces* files must contain entries for:

- Historical Server
- Any Adaptive Server that you want to monitor through Historical Server
- Monitor Server associated with each Adaptive Server listed

b Use dsedit (if your system is running X-Windows) or dscp (a command line utility) to add entries to an *interfaces* file. To add these entries, you must know the following information:

- Historical Server name.
- Monitor Server and Adaptive Server names that you want Historical Server to connect to.
- Port numbers or network addresses assigned to these servers when they were configured. To research this information, use dsedit or dscp on the machine where a server was configured to examine the appropriate *interfaces* file.

If you are relying on a directory service for making client/server connections, make sure that the *libtcl.cfg* file on *all* Historical Server client machines points to the appropriate directory service. Use a text editor to check and update *libtcl.cfg* files.

Initial configuration on Windows platforms

This section describes how to configure Historical Server on Windows machines. It includes the following topics:

- Assumptions on Windows
- Results of installation on Windows
- Configuration procedures on Windows

Assumptions on Windows

These procedures assumes that:

- The Historical Server software was loaded from the delivery media, using the instructions provided with the delivery media.
- An Adaptive Server/Monitor Server pair is configured on your network.

Results of installation on Windows

On the Windows platform, the Sybase installation process performs a nearly complete installation of Historical Server. The installation process:

- Copies Historical Server files to the Sybase installation directory.
- Adds parameter values for the Historical Server start-up command to the Registry. It uses default values for these parameters. The entries are under:

  ```
  \\HKEY_LOCAL_MACHINE\SOFTWARE\SYBASE\SERVER\
  servername\Parameters
  ```

- Adds the new Historical Server to the list of services in the Registry. The entry is under:

  ```
  HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\
  Services
  ```

- Adds Historical Server connectivity information to the *sql.ini* file in the Sybase installation directory on the Historical Server machine.

Configuration procedures on Windows

To complete Historical Server configuration:

1 Add connectivity information for Historical Server.

  This task assigns a port or network address to Historical Server. It also ensures that Historical Server can connect to one or more Adaptive Server/Monitor Server pairs. Add connection information either to *sql.ini* files or to a directory service. See *Open Client/Server Configuration Guide for Desktop Platforms* for more information on any of the following procedures and on ocscfg or dsedit.

  If you are relying on *sql.ini* files for making client/server connections, check the server listings in the *sql.ini* file. For Historical Server to run, this file must contain entries for *all* of the following servers:

  - Any Adaptive Server you want to monitor.
  - A Monitor Server paired with each Adaptive Server.

- Historical Server. Entries for Historical Server should exist, since the Sybase installation process adds them. However, since the Adaptive Server/Monitor Server pairs usually are running on a different machine from Historical Server, entries probably do not exist for them on the Historical Server machine. If this is the case, you must add entries for each Adaptive Server/Monitor Server pair to which Historical Server intends to connect.

Use dsedit to add entries to a *sql.ini* file. To add these entries, you must know the following information:

- Monitor Server and Adaptive Server names to which you want Historical Server to connect.
- Port numbers or network addresses assigned to these servers when they were configured. To research this information, use dsedit on the machine where a server is configured to examine the appropriate *sql.ini* or *interfaces* file.

Note The Adaptive Server name cannot be an alias name. It must be the name that Monitor Server knows it by. For example, use the value you used in the -S parameter in the Monitor Server start-up command.

If you edit a *sql.ini* file using a text editor instead of dsedit, make sure a carriage return exists at the end of the last line in the file.

If you are relying on a directory service for making client/server connections:

- Make sure that the *libtcl.cfg* file on the machine where Historical Server is installed points to the appropriate directory service. Use ocscfg to check and update *libtcl.cfg* files.
- Add Historical Server to the appropriate directory service, using dsedit. You must know the Historical Server name to complete this step. The default name created by the installation process is in the format *machineName_hs*. For example, *smith_hs*.

2 Configure Historical Server client machines.

This task enables clients to connect to Historical Server. Historical Server clients are users who create recording sessions or playback sessions. Each client machine must be configured appropriately.

If you are relying on *sql.ini* files for making client/server connections, then update *all* of the *sql.ini* files used by Historical Server clients. The client *sql.ini* files must contain entries for:

- Historical Server.
- The Monitor Server associated with each Adaptive Server listed.
- Any Adaptive Server that you want Historical Server to collect data for.

Use dsedit to add entries to a *sql.ini* file. To add these entries, you must know the following information:

- Historical Server name.
- Monitor Server and Adaptive Server names to which you want Historical Server to connect.
- Port numbers or network addresses assigned to these servers when they were configured. To research this information, use dsedit on the machine where a server was configured to examine the appropriate *sql.ini* file.

If you are relying on a directory service for making client/server connections, then make sure that the *libtcl.cfg* file on *all* Historical Server client machines points to the appropriate directory service.

Use ocscfg to check and update *libtcl.cfg* files.

3 Review start-up parameters in the Registry.

This task ensures that the default start-up parameter values that the installation process inserted into the Registry are suitable for your site.

When you start Historical Server using the Control Panel Services window, the server reads its start-up parameters from the Registry entry. If you start the server from a command line or by means of a batch file, Historical Server uses the start-up parameters from both the registry entry and from the command. If the same parameter appears in both places, the value specified in the command takes precedence over the value in the registry entry. If you do not specify any start-up parameters in the command, by default all of the Registry entry parameters are used.

Changing start-up parameters

To change start-up parameters or to change the server name in the Registry:

1 Start the Registry Editor (*regedt32.exe*, usually in *winnt\system32*).

2 For Windows, select the window named:

```
\\HKEY_LOCAL_MACHINE
```

or, for Windows 3.5.1:

```
\\HKEY_USER
```

and select the correct user entry.

3 Save or print the existing settings before proceeding. From the registry menu, select the Save Subtree As command or the Print command.

4 In the tree view, highlight:

```
\SOFTWARE\SYBASE\Server\
srvrName\Parameters
```

where *srvrName* is the name of the server whose start-up parameters you want to change. On the right side of the window, review the list of existing start-up parameters which appear in this format:

```
Argx, dataType, parameter
```

where:

- *x* is an integer in sequential order.
- *dataType* defines the type of data in the parameter value.
- *parameter* is a start-up option, preceded by a dash and followed by the parameter value.

An example containing the Historical Server -D, -U, and -P start-up parameters follows:

```
Arg0:REG_SZ:-Dc:\sybase\data\hs_data
Arg1:REG_SZ:-Uhssuper
Arg2:REG_SZ:-Pxwdfr
```

5 To add a new start-up parameter:

- Select Edit | Add Value.
- In the resulting dialog box, in the Value Name box, enter Arg*x* where *x* is the next integer not currently assigned.
- From the Data Type drop-down list, choose REG_SZ.
- In the resulting String dialog box, enter the parameter and value.

6 To modify existing parameters:

- Double-click the parameter line you want to change.
- In the resulting String dialog box, change the entry.

7 From the Registry menu, choose Exit.

Setting Historical Server start-up parameters

This section describes the Historical Server start-up command and parameters. The section applies to Historical Server running on both UNIX and Windows platforms.

Function

Starts Historical Server.

Syntax

```
histserver -U<user name> -P<password> -D<output dir> -l<log file> -
I<interfaces file> [-d<delimiter>] [-O<ASE name>] [-o<DATABASE
name>] [-f] -u<output ASE user name> -p<outputASEpassword> [-m]
```

where:

- -O specifies the name in the interfaces file of the Adaptive Server that will be used to output the Historical Server monitoring data
- -o specifies the database name in which to store the monitoring data
- -u enters the login name for connecting to the output Adaptive Server
- -p enters the corresponding password for the output Adaptive Server
- -f specifies that you want to create output files, and is required even if you've already designated an output Adaptive Server
- -m indicates that parallel output mode should be used

The executable name is histserver on UNIX platforms and histsrvr on Windows.

Parameters

Table 2-1 describes the parameters to the Historical Server start-up command.

Table 2-1: Historical Server start-up parameters

Parameter	Description
-D*homeDir*	**Required**. This parameter specifies the home directory for this instance of Historical Server. The account that starts Historical Server must have read, write, and search (execute) permissions on the Historical Server home directory. The default home directory is the current working directory for the account that starts Historical Server. The control file used by this instance of Historical Server resides in the home directory. The control file contains information about past recording sessions and is updated during the current run as new recording sessions are established. The directory specified by this parameter is also the default directory into which historical data files are written during a recording session. The user who creates a recording session can override this default location without affecting the definition of the control file. Multiple instances of Historical Server can share the same home directory. See "Configuring multiple instances of Historical Server" on page 23 for more information. **Note** If any UNIX shell-specific characters, such as the C shell's tilde (~), appear at the start of the directory's name, you must separate the -D parameter and the directory name by one or more space characters. Otherwise, the shell cannot recognize that file name expansion is required.
-d *delim*	Specify a default file delimiter for the view files that Historical Server generates. Delim can be: • A character. For example: ./histserver -d ' ' or ./histserver -d/ • an escaped character. For example: ./histserver -d'/t' • A four digit hexadecimal number representing the ascii code of the delimiter. For example: ./histserver -d0x09
-i*interfacesFile* (Upper or lower case "i" is valid after the dash.)	Path name of the *interfaces* file to use. The file must contain connection information, including the correct names, for Historical Server and all Adaptive Server and Monitor Server pairs to which you want this Historical Server to connect. If you omit this parameter, the name and location of the default *interfaces* file is: UNIX: *$SYBASE/interfaces* Windows: *SYBASE\sql.ini* (where *SYBASE* is the value of the *SYBASE* environment variable)

Parameter	Description
-l*logFile*	Path name of the log file in which information and error messages about the Historical Server are logged. The data collected includes start-up information, error messages, and possibly a record of alarms that were triggered. The default path name is *hs.log* in the current directory.
-U*userName*	Name of Historical Server superuser. When the superuser logs in to Historical Server, that user is allowed to issue the shutdown command to terminate Historical Server, and to view or delete any data files created by Historical Server, regardless of the user who initiated the recording sessions. This user name is not required to correspond to either an Adaptive Server login account or to an operating system registration. See "The Historical Server superuser account" on page 11 for more information. If you do not specify this parameter, any user may stop Historical Server, but no user has unrestricted access to the Historical Server data files.
-P*password*	Password of user specified with the -U parameter. A user who logs in to Historical Server must supply this *username* and *password* to exercise Historical Server superuser privileges. See "The Historical Server superuser account" on page 11 for more information. **Note** If you specify the -U parameter on UNIX platforms, but do not specify this parameter, Historical Server prompts for a password during start-up. However, Historical Server must be started in foreground mode for you to receive the prompt. Make the following changes to the start-up script to be prompted for a password: • Remove the -P parameter and the ampersand (&) from the start-up script file. • Execute the script file. • When prompted for a password, enter the password for the *username* you specified in the -U parameter. • Put the Historical Server process in the background.
-n*maxConnections*	Specifies the maximum number of concurrent Open Client connections allowed to Historical Server. Valid values are 1 through 20. The default is 20. Open Client connections to Historical Server are isql connections.

Parameter	Description
-S*hsName*	Name of Historical Server as it appears in the *interfaces* file. If omitted, the Historical Server name defaults to the value of the DSLISTEN environment variable. If DSLISTEN is not set, the name "histserver" is used.
-v	Displays Historical Server version information and exits, ignoring all other parameters.

Configuring multiple instances of Historical Server

There are times when running more than one instance of Historical Server is appropriate. However, you must configure the multiple instances correctly to either allow or restrict user access to the recording sessions of the multiple instances.

This section discusses:

- When to create multiple instances of Historical Server
- Configuring an additional Historical Server on UNIX platforms
- Configuring an additional Historical Server on Windows

When to create multiple instances of Historical Server

If a single instance of Historical Server is managing many concurrent recording sessions that have short sample intervals, the amount of time required to process and record the samples may interfere with the timely acquisition of new samples. If the timestamps in the recorded data files indicate that samples are not being collected at a reasonable approximation of the requested sample interval, you may choose to spread the load among two or more instances of Historical Server.

When multiple instances of Historical Server use the *same* home directory:

- Each instance must be started by the same operating system account.
- All instances that use the same home directory use the same control file.
- The sessions recorded by one instance are visible to the other instances.
- File-locking mechanisms ensure that the control file is not corrupted by concurrent accesses.

When multiple instances of Historical Server use *different* home directories:

- The Historical Server instances can be started by different user accounts.
- The Historical Server instances use different control files.
- The recording sessions of one server are not visible to the others. For example, the hs_list command, issued through isql, cannot access the recording sessions defined by another Historical Server using a different home directory.

When deciding whether multiple instances should use the same or different home directories, consider these factors:

- Visibility of data – if users of the multiple instances need access to each other's recording session definitions, use the same home directory. If sharing is undesirable, use different home directories. All input sessions to a playback session must be in the same home directory.
- Visibility of status information – the hs_status command cannot distinguish between multiple instances of Historical Server using the same home directory. If this distinction is important to Historical Server administration, use different home directories.
- Contention on the control file – if a large control file receives heavy access from multiple users, different home directories may give better user response time. Actions such as creating recording sessions and listing information about recording sessions use the control file; recording sessions themselves, when they are running, do not block other users from the control file.

Configuring an additional Historical Server on UNIX platforms

To configure an additional Historical Server on UNIX platforms:

1 Copy the start-up script that you created for the original Historical Server. Name the new file to reflect the name of the new Historical Server you want to start.

2 Edit the start-up file, changing the parameters to the start-up command as appropriate for the new instance of Historical Server. See "Setting Historical Server start-up parameters" on page 20. Pay particular attention to the -D parameter, which specifies the Historical Server home directory. You must decide whether you want multiple instances of Historical Server to share the same home directory or maintain separate ones.

3 Set up connectivity information for Historical Server, adding entries for the new Historical Server. Use a port number for the new Historical Server that is unique on its machine.

4 Edit the connectivity information on the client machines, adding entries for the new Historical Server.

Configuring an additional Historical Server on Windows

The Server Configuration utility cannot configure a new Historical Server. To configure an additional Historical Server after initial installation:

1 Adding start-up parameters to the Windows Registry

2 Updating the Windows Registry services list

3 Adding connectivity information for Historical Server

4 Configuring Historical Server client machines

5 Creating a .bat file (optional)

Adding start-up parameters to the Windows Registry

To add Historical Server start-up parameters to the Registry:

1 Start the Registry Editor (*regedt32.exe*, usually in *winnt\system32*).

2 For Windows 4.0, select the window named:

```
\\HKEY_LOCAL_MACHINE
```

or, for Windows 3.5.1:

```
\\HKEY_USER
```

and select the appropriate user.

3 Save or print the existing settings before proceeding. From the Registry menu, select the Save Subtree As command or the Print command.

4 In the tree view, highlight:

```
\SOFTWARE\SYBASE\Server
```

5 From the Edit menu, choose Add Key.

6 In the resulting dialog box, in the Key Name box, enter the name of the Historical Server you are adding.

Leave the Class box blank.

7 In the tree view, highlight:

```
\SOFTWARE\SYBASE\Server\hsName
```

where *hsName* is the new Historical Server name.

8 Select Edit | Add Value.

9 In the resulting dialog box, in the Value Name box, enter:

```
ServerType
```

From the Data Type drop-down list, choose:

```
REG_SZ
```

In the resulting String dialog box, enter:

```
HISServer
```

10 Make sure the entry for the new Historical Server name remains highlighted.

11 Select Edit | Add Key.

12 In the Key Name box, enter:

```
Parameters
```

Leave the Class box blank.

13 In the tree view, highlight:

```
\SOFTWARE\SYBASE\Server\hsName\Parameters
```

where *hsName* is the new Historical Server name.

14 Select Edit | Add Value.

15 In the Value Name box, enter:

```
Arg0
```

From the Data Type drop-down list, choose:

```
REG_SZ
```

In the resulting String dialog box, enter:

```
-ShsName
```

where *hsName* is the Historical Server name.

16 Repeat steps 13 through 15 until you have entered all of the start-up parameters for Historical Server.

At a minimum, enter the following required parameters. The order does not matter; for example, -D does not have to be Arg1.

Value name	Datatype	String
Arg0	REG_SZ	-S*hsName* where *hsName* is the Historical Server name you used in step 5. Example: `-SHS_SERVER1`
Arg1	REG_SZ	-D*dataDirectoryName* where *dataDirectoryName* is the full path name of an *existing* directory where you want to store Historical Server data. All users who will create Historical Server recording sessions must have write access to this directory. Example: `-Dc:\SYBASE\data\hs.data`
Arg2	REG_SZ	-I*installationRootDir\ini\sql.ini* Example: `-Ic:\SYBASE\ini\sql.ini`
Arg3	REG_SZ	-I*installationRootDir\install\logName* where *logName* is the file name where you want to store Historical Server error messages. The server creates the file on start-up if it does not exist. Example: `-lc:\SYBASE\install\hs.log`

Updating the Windows Registry services list

To add Historical Server to the list of services in the Registry:

1 Start the Registry Editor (*regedt32.exe*, usually in *winnt\system32*).

2 For Windows 4.0, select the window named:

```
\\HKEY_LOCAL_MACHINE
```

or, for Windows 3.5.1:

```
\\HKEY_CURRENT_USER
```

and select the correct user.

3 Save or print the existing settings before proceeding. From the registry menu, select the Save Subtree As command or the Print command.

4 In the tree view, highlight:

```
\SYSTEM\CurrentControlSet\Services
```

5 Select Edit | Add Key.

6 In the Key Name box, enter:

```
SYBHIS_hsName
```

where *hsName* is the name of the Historical Server you just configured.

Leave the Class box blank.

7 In the tree view, highlight:

```
\SYSTEM\CurrentControlSet\Services\SYBHIS_hsName
```

8 Select Edit | Add Value as many times as necessary to enter all of the following values, datatypes, and strings to the `SYBHIS_`*hsName* key entry.

Value name	Datatype	String
DisplayName	REG_SZ	*Sybase HISServer _ hsName*
ErrorControl	REG_DWORD	01 (Select hex radio button)
Group	REG_SZ	(leave blank)
ImagePath	REG_ EXPAND_SZ	*rootInstlDir\bin\histsrvr.exe -ShsName -C*
ObjectName	REG_SZ	LocalSystem
Start	REG_DWORD	03 (Select hex radio button)
Type	REG_DWORD	010 (Select hex radio button)

9 Make sure the `SYBHIS_`*hsName* key entry remains highlighted.

10 Select Edit | Add Key.

11 In the Key Name box, enter:

```
Security
```

Leave the Class box blank.

12 In the tree view, highlight the following Security key entry:

```
\SOFTWARE\SYBASE\Server\hsName\Security
```

13 Select Edit | Add Value.

14 Enter the following values for value name, datatype, and binary editor:

Value name	Datatype	Binary editor
Security	REG_BINARY	01001480

15 Select Registry | Exit.

16 Restart the machine.

Adding connectivity information for Historical Server

This task assigns a port or network address to Historical Server. It also ensures that Historical Server can connect to one or more Adaptive Server/Monitor Server pairs.

This task differs depending on whether you rely on *sql.ini* files or a directory service for making client/server connections:

- If you are relying on *sql.ini* files for making client/server connections, check the server listings in the *sql.ini* file for the Historical Server machine. For Historical Server to run, this file must contain entries for *all* of the following servers:
  - Any Adaptive Server you want to monitor
  - Monitor Server paired with each Adaptive Server
  - Historical Server

  Use dsedit to add entries to a *sql.ini* file. To add these entries, you must know the following information:
  - Monitor Server and Adaptive Server names to which you want Historical Server to connect.
  - Port numbers or network addresses assigned to these servers when they were configured.

  To research this information, use dsedit on the machine where a server was configured to examine the appropriate *sql.ini* file.

  Note The Adaptive Server name cannot be an alias name. It must be the name that Monitor Server knows it by. For example, use the value you used in the -S parameter in the Monitor Server start-up command.

- If you are relying on a directory service for making client/server connections:
  - Make sure that the *libtcl.cfg* file on the machine where Historical Server was installed points to the appropriate directory service. Use ocscfg to check and update *libtcl.cfg* files.

- Add Historical Server to the appropriate directory service, using dsedit. You must know the Historical Server name to complete this step. The default name created by the installation process is in the format *machineName_hs*. For example, *smith_hs*.

See *Open Client/Server Configuration Guide for Desktop Platforms* for instructions on using ocscfg and dsedit.

Configuring Historical Server client machines

This task enables clients to connect to Historical Server. Historical Server clients are users who create recording sessions or playback sessions. Each client machine must be configured appropriately.

The configuration task differs depending on whether you are using *sql.ini* files or a directory service for making client/server connections.

- If you are using *sql.ini* files for making client/server connections, then update *all* of the *sql.ini* files used by Historical Server clients. The client *sql.ini* files must contain entries for:
  - Historical Server
  - The Monitor Server associated with each Adaptive Server listed
  - Any Adaptive Server that you want Historical Server to collect data for

  Use dsedit to add entries to a *sql.ini* file. To add these entries, you must know the following information:
  - Historical Server name
  - Monitor Server and Adaptive Server names to which you want Historical Server to connect
  - Port numbers or network addresses assigned to these servers when they were configured

  To research this information, use dsedit on the machine where a server is configured to examine the appropriate *sql.ini* file.
- If you are relying on a directory service for making client/server connections, make sure that the *libtcl.cfg* file on *all* Historical Server client machines points to the appropriate directory service.

  Use ocscfg to check and update *libtcl.cfg* files.

Creating a *.bat* file (optional)

This task allows you to start Historical Server by executing a batch file, rather than by using the services icons.

To prepare for Historical Server start-up from a batch file:

1 Start any text editor and open a new text file.

2 Enter the Historical Server start-up command, histsrvr, and all desired parameters. Use the full path name of the start-up command *histsrvr.exe*. See "Inferring start-up parameters from the Registry" on page 37 for precedence rules used by Historical Server to obtain start-up parameters. See "Setting Historical Server start-up parameters" on page 20 for explanations of start-up parameters.

An example line in the *.bat* file follows. The example has carriage returns inserted in the command. Do not put carriage returns in your *.bat* file. The entire file should be one line.

```
c:\sybase\bin\histsrvr.exe -Shs_server1
-Dc:\sybase\data\hs_data
-Ic:\sybase\ini\sql.ini
-lc:\sybase\data\hs.log
```

3 Save the file with a *.bat* extension. Sybase recommends that you use *RUN_hsName.bat*. For example:

```
RUN_hs_server1.bat
```

CHAPTER 3

Starting and Stopping Historical Server

Topics	Page
Starting and stopping Historical Server on UNIX platforms	33
Starting and stopping Historical Server on Windows	36

Starting and stopping Historical Server on UNIX platforms

This section describes how to start and stop Historical Server running on UNIX platforms.

Starting Historical Server on UNIX

On UNIX, you can start a configured Historical Server in two ways:

- Execute the histserver command from a UNIX shell prompt. If you use this method, you must type all appropriate parameters each time.
- Execute a script file that contains the histserver command and all appropriate parameters. If you followed the configuration instructions in Chapter 2, "Configuring Historical Server," you would start Historical Server using:

  ```
  install_dir/install/RUN_histServerName
  ```

  where *install_dir* is the Sybase root directory and *histServerName* is the name of the Historical Server you want to start.

Regardless of which method you use:

- Verify that the Adaptive Server to be monitored and its corresponding Monitor Server are running.

- Use the same account each time you start Historical Server, if you want the new instance to have access to previously recorded sessions. See "The operating system start-up account" on page 10 for more information.
- Set the $SYBASE environment variable to the root directory of the Sybase installation.

  The $SYBASE environment variable must contain the name of a directory that has the appropriate *locales* and *charsets* subdirectories for Historical Server. These subdirectories were created and populated automatically by the installation procedure.

  The $SYBASE environment variable also identifies the default location of the *interfaces* file. Use parameters to the histserver command to override the default location.

Historical Server displays the following message to indicate that start-up was successful:

```
Initialization is over. Ready to accept connections.
```

Historical Server writes messages to its log file during start-up. You can ignore these messages if start-up was successful. If start-up is not successful, examine the log file to research the problem.

The default path name for the Historical Server log file is *hs.log* in the current directory at the time of start-up. You can override this default path name with the -l parameter (the letter l) to the histserver command.

Stopping Historical Server on UNIX

This section describes:

- Who can shut down Historical Server
- Determining current activity on Historical Server
- Deferred versus immediate shutdown
- Detailed shutdown procedures

Who can shut down Historical Server

If the start-up command specifies a superuser account, then only the superuser account can stop Historical Server. If a superuser is not specified in the start-up command, any user can stop Historical Server.

The superuser is the one whose account was specified in the -U and -P parameters to the Historical Server start-up command.

Determining current activity on Historical Server

Before shutting down Historical Server, check current activity on Historical Server to determine if you want to issue a deferred shutdown or an immediate shutdown. A deferred shutdown lets all current activity complete before terminating Historical Server.

Historical Server activity might include fully defined and initiated recording sessions, uninitiated recording sessions with definitions still in progress, playback sessions in progress, and playback sessions being defined. Client connections can be from multiple machines and they might be monitoring various Adaptive Servers.

To determine current activity in Historical Server, connect to Historical Server with the isql utility and issue the hs_status activity command.

Deferred versus immediate shutdown

To close Historical Server, connect to it using the isql utility and issue one of the following commands:

- hs_shutdown – defers shutdown until all active recording sessions complete and any other active connections are closed. No new connections are accepted during this time.

  A deferred shutdown may block for an extended period of time, and typing Ctrl+C while the hs_shutdown command is blocked has no effect.

- hs_shutdown no_wait – shuts down Historical Server immediately and terminates any connections and active recording sessions.

In either case, the active recording sessions are shut down in a controlled manner. The Historical Server control file and the historical data files are available to Historical Server when it is restarted.

Detailed shutdown procedures

Shut down servers in the following order:

- Historical Server
- Monitor Server

- Adaptive Server

To stop Historical Server on a UNIX platform:

1 Connect to Historical Server. If you are using isql, the command is:

```
isql [ -Uhs_superuser_name -Phs_superuser_password ]
 -Shistorical_server
```

where:

- *superuser_name* is the name that was used with the -U parameter to the Historical Server start-up command. If -U was not used in the start-up command, any user can stop Historical Server, and the -U parameter in this isql command is optional.
- *superuser_password* is the password that was used with the -P parameter to the Historical Server start-up command. If -U was not used in the start-up command, any user can stop Historical Server, and this parameter is optional.
- *historical_server* is the name of the Historical Server you want to stop.

2 To determine current activity on Historical Server, issue the following command when the isql prompt appears:

```
1> hs_status activity
2> go
```

3 To shut down Historical Server, issue one of the following commands:

```
1> hs_shutdown
2> go
```

or:

```
1> hs_shutdown no_wait
2> go
```

Starting and stopping Historical Server on Windows

This section describes how to start and stop Historical Server running on Windows platforms. Topics are:

- Starting Historical Server on Windows
- Inferring start-up parameters from the Registry

Starting Historical Server on Windows

You can start Historical Server by using:

- The Windows Control Panel Services window.
- A batch (*.bat*) file containing the start-up command and parameters. The file name is *run_hsName.bat*, where *hsName* is the name of the Historical Server instance. Sybase recommends that you invoke the batch file from a command line shell rather than by double-clicking on it in File Manager. The command line shell captures start-up error messages, if any occur, whereas the File Manager does not.
- The start-up command typed directly from a command line shell. The name of the Historical Server executable file for Windows is *histsrvr.exe*.

When you use a *.bat* file or a command line command to start Historical Server, the server process is linked to your current login account. When you log off, the server shuts down.

For production systems, Sybase recommends that you start Historical Server using the Windows Control Panel Services Manager. When you start a server as a service, it persists across logins.

Inferring start-up parameters from the Registry

Start-up parameters for Historical Server are in the following Registry entry:

```
\SOFTWARE\SYBASE\SERVER\servername\Parameters
```

When you start Historical Server using the Control Panel, the server reads its start-up parameters from this Registry entry. If you start the server from the command line or by means of a batch file, the start-up parameters are taken from both the registry entry and from the command. If the same parameter appears in both places, the value specified in the command takes precedence over the value in the registry entry. If you do not specify any start-up parameters in the command, by default all of the Registry entry parameters are used.

See "Adding start-up parameters to the Windows Registry" on page 25 for information on editing the Registry entries.

Verifying that Historical Server is running

On Windows, check the status of Historical Server in the Windows Control Panel Services window.

Stopping Historical Server on Windows

This section describes:

- Who can shut down Historical Server on Windows
- Determining current activity on Windows
- Deferred versus immediate shutdown on Windows
- Detailed shutdown procedures on Windows
- Shutdown methods to avoid on Windows

Who can shut down Historical Server on Windows

If a superuser account is specified at start-up, only the superuser account can stop Historical Server. If a superuser is not specified in the start-up command, any user can stop Historical Server. Administrators with privileges to use the Control Panel Services window can also shut down Historical Server.

The superuser account is specified in the `-U` and `-P` parameters to the Historical Server start-up command.

Determining current activity on Windows

Before shutting down Historical Server, check current activity on Historical Server to determine if you want to issue a deferred shutdown or an immediate shutdown. A deferred shutdown lets all current activity complete before terminating Historical Server.

Historical Server activity might include fully defined and initiated recording sessions, uninitiated recording sessions with definitions still in progress, playback sessions in progress, and playback sessions being defined. Client connections can be from multiple machines and they might be monitoring various Adaptive Servers.

To determine current activity in Historical Server, connect to Historical Server with the isql utility and issue the hs_status activity command.

Deferred versus immediate shutdown on Windows

To close Historical Server, connect to it using the isql utility and issue one of the following commands:

- hs_shutdown – defers shutdown until all active recording sessions complete and any other active connections are closed. No new connections are accepted during this time.

  A deferred shutdown may block for an extended period of time, and typing Ctrl+C while the hs_shutdown command is blocked inside the isql has no effect.

- hs_shutdown no_wait – shuts down Historical Server immediately and terminates any connections and active recording sessions.

In either case, the active recording sessions are shut down in a controlled manner. The Historical Server control file and the historical data files are available to Historical Server when it is restarted.

Detailed shutdown procedures on Windows

Shut down servers in the following order:

- Historical Server
- Monitor Server
- Adaptive Server

To stop Historical Server on Windows, use any of these methods:

- Click the Stop button on the Windows Control Panel Services window.
- Issue the hs_shutdown command from isql.

- If start-up was launched from the command line or by means of a batch file, then Historical Server is associated with the login that started it. In these cases, Historical Server shuts down if you close the window in which it was started or if you log off of the system.

  If the controlled shutdown of Historical Server takes longer than a predetermined time interval (five seconds for closing the window or twenty seconds for logoff), the system displays a pop-up dialog box at regular intervals, asking you whether or not to terminate the process.

  Warning! Unless you respond with Wait each time the dialog box is presented, Historical Server shuts down in an uncontrolled way, risking data loss and file corruption.

To stop Historical Server using isql:

1 Connect to Historical Server using isql:

```
isql -Uhs_supersuer_name -Phs_superuser_password
-Shistorical_server
```

where:

- *superuser_name* is the name that was used with the -U parameter to the Historical Server start-up command. If -U was not used in the start-up command, any user can stop Historical Server, and this parameter is optional.
- *superuser_password* is the password that was used with the -P parameter to the Historical Server start-up command. If -U was not used in the start-up command, any user can stop Historical Server, and this parameter is optional.
- *historical_server* is the name of the Historical Server you want to stop.

2 To determine current activity on Historical Server, issue the following command when the isql prompt appears:

```
1> hs_status activity
2> go
```

3 When the isql prompt appears, issue one of the following commands:

```
1> hs_shutdown
2> go
```

or:

```
1> hs_shutdown no_wait
2> go
```

Shutdown methods to *avoid* on Windows

In some cases, shutting down your system without first shutting down Historical Server can cause uncontrolled shutdown. Unless Historical Server is experiencing a very high level of recording activity, its controlled shutdown should take less than 20 seconds. If 20 seconds or more are used for the controlled shutdown, however, the system may intervene and terminate Historical Server in an uncontrolled way.

Warning! To be safe, Sybase recommends that you manually stop Historical Server before you shut down the system.

Using the Kill Process button causes an uncontrolled shutdown.

If Microsoft Visual C++ Process Viewer tool is installed on your system, Sybase recommends that you do *not* use the Kill Process button in the Process Viewer window.

CHAPTER 4

Command Reference

This chapter describes the Historical Server command interface.

Command summary

Table 4-1 summarizes the Historical Server commands.

Table 4-1: Historical Server commands

Activity	Commands
Creating recording sessions	Use these commands to create a recording session: • hs_create_recording_session – defines the characteristics of a recording session. • hs_create_view – defines a view, which is a collection of data to record. • hs_create_filter – specifies filtering criteria on a data item. • hs_create_alarm – specifies a threshold value for a data item that triggers an alarm action. • hs_initiate_recording – ends recording session definition and enables recording to begin at its predefined start time. • hs_terminate_recording – stops a recording session.
Viewing recording session definitions	• hs_list – lists all of the views defined for a recording session and their data items.

Activity	Commands
Creating a playback session	Use these commands to create a playback session: • hs_create_playback_session – defines the characteristics of a playback session • hs_create_playback_view – specifies one or more views from the input recording sessions to include in the playback session. • hs_initiate_playback – ends playback session definition. Also starts playback to a file. • hs_playback_sample – plays back a sample to a client. • hs_terminate_playback – stops a playback session.
Administering Historical Server	Use these commands to manage Historical Server: • hs_delete_data – deletes the files containing the recorded data for specified recording sessions. • hs_status – displays the current status of Historical Server. • hs_shutdown – stops Historical Server.

Command syntax

The syntax for Historical Server commands is:

hs_*XXX arg1*, *arg2*, *arg3*, ..., *argn*

where *XXX* is the command name. For example, a typical command is hs_create_recording_session.

Parameters must be separated by commas. You can omit parameters that are optional or that have default values; however, you must use the value NULL for those parameters as a placeholder if other parameter values follow them. For example, the following command uses the default value for most parameters:

```
hs_create_playback_session null, null, raw, null,
null, client, null, null, null, null, 7
```

Command names, command keywords, the word null, data items, and statistic types are all case-insensitive. File names, view names, and other user-supplied names are case-sensitive.

If a parameter value contains embedded spaces (such as those in data items, statistic types, and date-time specifications), you must surround the value with quotes. Matched pairs of single-quote or double-quotes are valid delimiters.

If the parameter value contains an embedded quote that is the same as the character used to delimit the entire value, supply a pair of the quotes within the parameter value. Historical Server compresses the pair of quotes to a single character.

The word null within quotes is not a keyword.

You can enter Historical Server commands on multiple lines.

Command status and errors

All Historical Server commands return a status value of either zero or one.

- Zero indicates a successful execution.
- One indicates an error condition. If a status of one (“1”) is returned, an error message and its appropriate error code, severity level, and state are returned to the client.

Historical Server is an Open Server application that uses Client-Library™ to communicate with one or more pairs of Adaptive Server and Monitor Server. Any of these components may detect and report error conditions.

Historical Server also detects and reports error conditions, which it logs or reports or both to clients. Historical Server error codes are all five-digit numbers from 30000 through 30999. Open Server error codes are in the range between 16000 and 20000. For error codes in either of these ranges, the Historical Server assigns one of the following severity levels:

Table 4-2: Severity levels in command errors

Severity	Description
1	Informational message
2	Warning message
3	Fatal server error
4	Fatal process error
5	Operating system error

Errors detected by Client-Library, Adaptive Server, and Monitor Server use their own error numbering and severity-level schemes.

Script files as input to Historical Server

A convenient way to provide input to Historical Server is to enter the commands in a text file and use that file as input. The Historical Server input file is not an isql file, and does not necessarily conform to isql conventions. For example, Historical Server does not have a comments feature.

Appendix D, "Examples of Recording Session Views" provides examples of views that you might use as a start for your input files. These views also appear in the *views* file that was installed in the *sample/histserver* subdirectory in the installation directory. However, that the file includes explanatory text that you must remove before using the file as input to Historical Server.

Connecting to Historical Server

This section contains the following topics:

- Assumptions before connection
- How to connect
- Required permissions for Historical Server activities
- Mutually exclusive sessions

Assumptions before connection

Before you can connect to Historical Server:

- The Adaptive Server whose performance data you want to collect must be running.

- A Monitor Server that is monitoring the Adaptive Server must be running. See *Sybase Adaptive Server Enterprise Monitor Server User's Guide* for instructions on verifying whether an Adaptive Server/Monitor Server pair is running.
- The Historical Server must be running. See Chapter 3, "Starting and Stopping Historical Server" for instructions on verifying whether a Historical Server is running.
- Connection information must be configured correctly on both the Historical Server machine and its client machines. See Chapter 2, "Configuring Historical Server" for more information.

How to connect

Connect to Historical Server using a utility that offers an interactive interface to Sybase servers, such as SQL Advantage or isql. Follow the standard connection procedures for the utility. For example, in SQL Advantage, choose the Connect menu item. In isql, start isql using the Historical Server name as the value of the -S parameter.

Required permissions for Historical Server activities

Connect to Historical Server using a user login and password that has permissions for the activities you intend to perform, as described in Table 4-3. Depending on the permissions required, you might have to disconnect and reconnect with different login accounts to perform various activities.

Table 4-3: Permissions required for Historical Server activities

Activity	Permissions required to perform the activity
Superuser activities: • Stop Historical Server. • Delete Historical Server files, including recording session data files. • Access files from private recording sessions.	Superuser permissions. To obtain superuser permissions, connect to Historical Server using the superuser account defined at start-up. The superuser account does not have to be a valid account in any Adaptive Server.

Activity	Permissions required to perform the activity
Create recording sessions.	Valid login and password in the Adaptive Server being monitored. Execute permission on the mon_rpc_connect stored procedure in the Adaptive Server being monitored. This permission is required to connect to Monitor Server.
Create playback sessions using public recording session files (from recording sessions whose *protection_level* parameter is **public**). Use hs_list command on public recording session files.	Valid login and password in the Adaptive Server being monitored.
Create playback sessions using private recording session files (from recording sessions whose *protection_level* parameter is **private**). Use the hs_list command on private recording session files.	The same login that created the private sessions, or superuser. In the latter case, the superuser account must be a valid login in the Adaptive Server being monitored.

A single Historical Server can connect to and collect data from multiple Adaptive Server/Monitor Server pairs. For example, from the same Historical Server connection, you could create two recording sessions, one gathering performance data for an Adaptive Server named server1 using a Monitor Server named server1_MON, and the other gathering data for an Adaptive Server named server2 using a Monitor Server named server2_MON. If you do not have a login account that is valid in both Adaptive Server instances, you would need to create these recording sessions in different connections to Historical Server, using different login accounts. The data files from the two recording sessions, however, could reside in the same directory.

The hs_create_recording_session command specifies the Monitor Server you want to connect to, which indirectly implies the Adaptive Server you want performance information about.

Mutually exclusive sessions

A Historical Server connection can be involved with only one session at a time. You can either be defining a recording session, or defining and executing a playback session, but you cannot mix together commands for multiple sessions. That is, if you start a recording session definition with the hs_create_recording_session command, you must finish the sequence of commands to define that session (or cancel the session) before you can start to define a playback session or a new recording session.

Defining a recording session is not the same thing as a recording session in progress. The user who defines a recording session is connected to and actively communicating with Historical Server during definition of a recording session. The user does not actively communicate with Historical Server during the actual recording. Any user connections are irrelevant during recording. For example, a user might connect to Historical Server at 3 p.m., define a recording session that starts at 10 p.m., and then disconnect from Historical Server. At 10 p.m., it is irrelevant whether the user is connected to Historical Server; the recording session takes place if Historical Server is running.

A playback session, however, requires the user connection during both playback definition and playback execution. To cancel playback execution and allow the user connection to perform some other communication, the user must issue the hs_terminate_playback command.

Starting and ending a recording session definition

The hs_create_recording_session command starts a recording session definition. After issuing this command, you further define the recording session by issuing other commands to create views, filters, and alarms for the recording session.

To end the recording session definition, use either:

- hs_initiate_recording – signals that the definition is complete and enables the recording session to start at the session's specified start time.
- hs_terminate_recording – cancels a recording session definition that is in progress.

Starting and ending a playback session

The hs_create_playback_session command starts a playback definition. After issuing this command, you use other commands to define the playback session and perform the playback.

After issuing a successful hs_create_playback_session command, you must successfully execute a hs_terminate_playback command to end the playback session. This command cancels the definition and ends playback.

Historical Server commands

The following pages describe the Historical Server commands:

- hs_create_alarm
- hs_create_filter
- hs_create_playback_session
- hs_create_playback_view
- hs_create_recording_session
- hs_create_view
- hs_delete_data
- hs_initiate_playback
- hs_initiate_recording
- hs_list
- hs_playback_sample
- hs_shutdown
- hs_status
- hs_terminate_playback
- hs_terminate_recording

hs_create_alarm

Description

Creates an alarm. An alarm is triggered when a data item value reaches a specified threshold value.

Syntax

hs_create_alarm *view_name*, *data_item_name*, *data_item_stat*, *alarm_action*, *alarm_action_data*, *alarm_value*, *occurence_threshold*.

Parameters

view_name

name of the view that contains the data item to which the alarm applies.

data_item_name

data item to which the alarm applies. If the data item contains embedded spaces, surround it with quotation marks.

data_item_stat

statistic type for *data_item_name*. The *data_item_name* and *data_item_stat* combination must exist in the view definition. Surround the statistic type with quotation marks.

alarm_action

action to take when an alarm condition occurs. Values are:

- log – logs messages when the alarm condition occurs. You specify the log file name in the *alarm_action_data* parameter.
- execute – executes a program or script file when the alarm condition occurs. You specify the file to execute in the *alarm_action_data* parameter.

alarm_action_data

specifies information required to carry out the *alarm_action*:

If alarm_action is	**Then alarm_action_data is**
log	Name of the file where Historical Server should log the alarm messages. The default is the Historical Server log file. If you specify a file that does not exist, Historical Server creates it. If you specify an existing file, the user who started Historical Server must have write permission for the file.

If alarm_action is	Then alarm_action_data is
execute	Name of the file to execute, optionally followed by a list of parameters separated by spaces. The file must exist and the user who started Historical Server must have execute permission for it. **Warning!** When an alarm condition occurs and *alarm_action* is execute, the specified file is executed by the account that started Historical Server, not by the account that created the alarm. This means that the access privileges of the person who starts Historical Server are available to the users who define alarms. A user who normally cannot execute a file may be able to execute it through an Historical Server alarm.

alarm_value
triggering value for the alarm.

occurance_threshold
number of times the threshold condition occurs before Hisotrical Server performs the threshold action..

Examples

1 This example creates an alarm for a view named Page I/O. The view contains the Page I/O data item with a Value for Session statistic type. The data is logged to the *page_io_alarm_file* file when a trigger value of 50 or greater is achieved.

```
hs_create_alarm PageIO,"Page I/O",
 "Value for Session",log,page_io_alarm_file,50
```

2 This example creates an alarm for a view named Page I/O. This alarm causes the */user/script1* script to be executed and passes the value 100 as its first parameter when a trigger value of 100 or greater is achieved on the Page I/O data item over six consecutive refreshes..

```
hs_create_alarm PageIO,"Page I/O",
 "Value for Session",execute,
 "/user/script1 100",100,6
```

Usage

- If *alarm_action* is execute, when the alarm condition occurs, the script or program executes in the background. The commands within the script or program are executed in the foreground and displayed if applicable.

For example, if you want the clock to display in an alarm condition, you should put *clock.exe* into a script file and specify the script file name when creating the alarm, rather than specifying *clock.exe* when creating the alarm.

- When an alarm condition occurs and *alarm_action* is log, Historical Server writes three lines to the log file. The first line contains:
  - Timestamp of the sample that triggered the alarm
  - Name of the data item on which the alarm was set
  - Statistic type of that data item
  - Value of the data item that triggered the alarm
  - Threshold value of the alarm
- The second line in the log file contains:
  - Session ID
  - View name
  - Alarm ID
- The third line in the log file contains:
  - Name of the Adaptive Server being monitored
  - Names and values of all key data items in the view for the row of sample data that triggered the alarm
- When an alarm condition occurs and *alarm_action* is execute, Historical Server runs the script or program specified in the *alarm_action_data* parameter. Historical Server invokes the program or file using the arguments specified in *alarm_action_data*, followed by additional positional arguments that can be accessed within the program. The positional arguments appended to the call by Historical Server represent the following information:
  - Session ID
  - Alarm ID
  - Sample timestamp
  - Data item on which the alarm was set
  - Statistic type of that data item
  - Value of the data item that triggered the alarm

  - Threshold value of the alarm
  - Name of the Adaptive Server being monitored
  - Number of key data items in the view
  - For each key data item:
    - Name of the data item
    - Value of that data item for the row of sample data that triggered the alarm
- When an alarm condition occurs and *alarm_action* is execute, the specified file is executed by the account that started Historical Server, not by the account that created the alarm. This means that the access privileges of the person who starts Historical Server are available to users who define alarms.

Warning! It is possible that a user who normally cannot execute a file can execute that file through an Historical Server alarm.

hs_create_filter

Description

Specifies filtering criteria on a data item. A filter limits the scope of collected data for a single data item in a view. Filters are optional.

Syntax

hs_create_filter *view_name*, *data_item_name*, *data_item_stat*, *filter_type*, *value_spec*

Parameters

view_name
name of the view that contains the data item to filter.

data_item_name
data item to which the filter applies. If the data item contains embedded spaces, surround it with quotation marks.

data_item_stat
statistic type for *data_item_name*. The *data_item_name* and *data_item_stat* combination must exist in the view definition. Surround the statistic type with quotation marks.

filter_type

type of filter to use. Valid values are:

- eq – passes values that match any of a list of values. That is, a value passes the filter if it equals the first filter value, or the second filter value, or any other value in the list. Specify the list of values in the *value_spec* parameter.
- neq – passes values that are not equal to any value in a list of values. That is, a value passes the filter if it is not equal to the first filter value, not equal to the second filter value, and not equal to any other value in the list. Specify the list of values in the *value_spec* parameter.
- range – passes values that fall inside a range of values. Specify the ranges in the *value_spec* parameter.
- top – passes a specified number of values that are the highest values received for the data item during a sample interval. For example, a top 10 filter passes the 10 rows of a sample that have the highest values for the data item.

value_spec

specifies values for the filter tests. The syntax depends on the value of *filter_type*:

If filter_type is	**value_spec syntax is**
eq or neq	*value1*[, *value2*]... See "Using wildcards" on page 57 for information about using a wildcard character (%) when *valuen* is a character string. See "Specifying filters on object name and procedure name" on page 57 for information on specifying a value for the Object Name and Procedure Name data items.

If filter_type is	value_spec syntax is
range	To specify values greater than or equal to a low bound: low,*value* To specify values less than or equal to a high bound: high,*value* To specify values greater than or equal to a low bound *and* less than or equal to a high bound: low,*value1*,high,*value2* See "Specifying filters on object name and procedure name" on page 57 for information on specifying a value for the Object Name and Procedure Name data items. If you are specifying both an upper and lower bound for either of these data items, each bound must have the same number of subcomponents.
top	*value* where *value* is the number of items to be passed. It must be greater than zero.

Examples

This example creates a filter for a view named Page I/O. The view contains the Page I/O data item with a "Value for Session" statistic type. The example limits the rows recorded for a sample to the 20 rows that have the highest total number of Page I/Os for the recording session to date.

```
hs_create_filter PageIO,"Page I/O","Value for
Session",top,20
```

Usage

Using multiple filters in a view

Filters limit the amount of data that Historical Server records. Each filter applies to one data item in the view. Each data item can have only one filter applied to it.

If multiple data items in a view have filters, Historical Server records data that satisfies all of the individual filters. Effectively, the Boolean AND operation is applied to the values of the various filters.

If the top filter is in effect for a data item and another filter is in effect for a second data item, some of the values passed by the top filter may be eliminated by the other filter. The result is that less that specified top *n* values are returned.

Using wildcards

The wildcard character is the percent sign (%). When *filter_type* is eq or neq, you can use the wildcard in *value_spec* for data items that return character strings.The wildcard matches any string of zero or more characters. The wildcard can appear anywhere within the filter value (beginning, end, or anywhere in between). Only a single instance of the wildcard can appear in *value_spec.*

An exception to the single instance rule for the wildcard pertains to the Object Name and Procedure Name data items. In those multi-component data items, you can use a single instance of the wildcard in each component. See "Specifying filters on object name and procedure name" on page 57 for more information.

When *filter_type* is range, you cannot use the wildcard character in *value_spec*.

Specifying filters on object name and procedure name

The Object Name and Procedure Name data items accept multi-component values in *value_spec* when *filter_type* is eq, neq, and range. These composite filter values can use the following name forms, where *object_name* is either a table name or a stored procedure name:

- *object_name*
- *owner_name.object_name*
- *database_name.owner_name.object_name*

If *filter_type* is eq or neq, each component can contain a single instance of the wildcard character (%). The wildcard can be used in place of the entire component. For example, a *value_spec* of `%.%.authors` for the *data_item* Object Name sets a filter that returns the authors table in all databases, regardless of owner. See "Using wildcards" on page 57 for more information about the wildcard character.

If *filter_type* is range with both upper and lower bounds, the two bounds must have the same number of subcomponents.

Some specific considerations for designing filters using these component data items follow:

- If *database_name* is a component of a composite filter value for an "Object Name" data item, a "Database Name" data item also must be present in the view.

- If *owner_name* is a subcomponent of any composite filter value for an "Object Name" data item, an "Owner Name" data item also must be present in the view.
- If *database_name* is a subcomponent of any composite filter value for a "Procedure Name" data item, a "Procedure Name" data item also must be present in the view.
- If *owner_name* is a component of any composite filter value for a "Procedure Name" data item, a "Procedure Owner Name" data item also must be present in the view.

hs_create_playback_session

Description

Defines the characteristics of a playback session. This command is the first step in creating a playback session.

Syntax

hs_create_playback_session *start_time*, *end_time*, *summarization_interval*, *allow_estimation*, *missing_data_option*, *target*, *directory_name*, *protection_level*, *sample_interval*, *script_type*, *delete_option*, *session_id* [, *session_id*...]

Parameters

start_time

specifies the date and time of the beginning of the recorded data to be played back. The default is to start playback from the beginning of the first session specified by *session_id*. Use the value NULL to accept the default.

The format for *start_time* is:

"year/month/day hour:minute[:second] [time zone]"

Depending on the *summarization_interval* and *missing_data_option* parameters, and depending on whether there is any data available at the time specified, playback might use data from a time later than that specified; however, playback does not use data from a time earlier than *start_time*.

The *start_time*, if specified, must be earlier than the end time of at least one of the sessions specified in *session_id*. The following time zone options are available:

Parameter value	Explanation
EST	U.S. eastern time zone, standard time.
EDT	U.S. eastern time zone, daylight saving time.
CST	U.S. central time zone, standard time.
CDT	U.S. central time zone, daylight saving time.
MST	U.S. mountain time zone, standard time.
MDT	U.S. mountain time zone, daylight saving time.
PST	U.S. Pacific time zone, standard time.
PDT	U.S. Pacific time zone, daylight saving time.
MET	Middle European time zone, standard time.
MET DST	Middle European time zone, daylight saving time.
WET DST	Western European (Greenwich) time zone, daylight saving time.
GMT	Greenwich mean time. This is equivalent to western European (Greenwich) time zone without regard to daylight saving time. All of the preceding time zone specifications, such as EST or EDT, can be supplied only in combination with dates and times when standard time or daylight savings time is in effect. GMT can be paired with any date and time specification.
GMT{+-} *hours_offset*	To specify any other time zone, where *hours_offset* is the number of hours that must be added to Greenwich mean time to derive the local time. The acceptable range of offset values is between +24 and -24 hours, inclusive. Fractional offsets such as +5.5 are valid.

The default *time_zone* is the local time zone of Historical Server.

end_time

specifies the date and time for the end of the recorded data to be played back. The default is to end playback at the end of the last session specified. Use the value NULL to accept the default.

The format of this parameter is the same as the *start_time* format. The *end_time*, if specified, must be later than the *start_time* of at least one input session.

summarization_interval

a *required* parameter (no default exists) that specifies the level of detail of the playback. Valid values are:

- raw – plays back data as it was collected, using the same sample intervals. Choose this option to view raw data as it was recorded. Also, this is the only option available for playing back snapshot data, such as current SQL statement data and status information on locks or processes. See Table C-2 on page 169 for a definitive list of the snapshot data items. (They are the ones with "no" in the "Allowed for Non-raw" column.)

  This option is valid only when *target* is client.

- actual – plays back data using the same sample intervals as the input recording sessions.

  This option allows you to specify some data item changes in the playback view. Also, Historical Server makes appropriate adjustments to the first and last samples when the recording session times do not align with the requested playback session times.

  Choose this option to add or change certain data items when summarization is not required. Also, when *target* is set to file, this option provides a way to concatenate the non-snapshot data in multiple recording sessions.

  This option is valid only when *target* is client.

- entire – plays back data for each input recording session summarized as a single sample. The sample interval is the timespan between the requested playback *start_time* and *end_time*.

  This option allows you to specify some data item changes in the playback view. Also, Historical Server makes appropriate adjustments to the data values to accurately reflect the requested playback session start and end times.

Choose this option to consolidate recorded data, rolling up details into overviews of activity over longer time periods.

sample_interval

plays back data summarized into sample intervals of the specified length. The parameter value is the sample interval length, specified as:

```
"S"
"M:S"
"H:M:S"
"D H:M:S"
```

where:

- *S* – is seconds.
- *M* – is minutes.
- *H* – is hours.
- *D* – is days.

All components are numeric and can be one or two digits. Some examples are:

```
"30" (specifies sample intervals of 30 seconds)
"10:0" (specifies sample intervals of 10 minutes)
"8:30:0" (specifies sample intervals of 8 1/2 hours)
"5 0:0:0" (specifies sample intervals of 5 days)
```

The first sample interval starts at *start_time*, and every sample, except possibly the last one, has the specified length.

This option allows you to specify some data item changes in the playback view. Also, Historical Server makes appropriate adjustments to the data values to accurately reflect the requested playback session length and playback sample intervals.

Choose this option to summarize data into any desired granularity. This type of summary can mediate deviations in activity and is useful for observing trends over time.

The actual, entire, and user-defined interval options have the following features in common:

- Data item changes in playback views – for all three options, the playback view can use statistic types different from those in the input view, and it can include some estimated and calculated data items not in the input view. It cannot include snapshot data, such as current SQL statement data. See Table C-2 on page 169 for a definitive list of data items and statistic types that can be included in playback views when these options are chosen.
- Data adjustments – for all three options, Historical Server performs appropriate mathematical computations to compensate for differences in:
  - the input recording sessions sample intervals and the requested playback sample intervals
  - the input recording sessionss session lengths and the requested playback start and end times

  To handle these differences, accumulated counts are prorated. Percentages and rates are weight-averaged, weighted by the number of seconds that each input sample contributes.

  See Appendix C, "Specifications for Defining Playback Views" for more information.

allow_estimation

specifies whether or not you want playback to estimate values for data items that cannot be calculated exactly. Valid values are:

- disallow (the default) – causes the hs_create_playback_view command to return an error if it encounters data items that require estimation.
- allow – causes playback to estimate values, if necessary, for certain data items. Some data items cannot be included in a playback view unless you allow estimation in the playback session.

This parameter is ignored if *summarization_interval* is raw.

See Table C-2 on page 169 to determine which data items require estimation.

missing_data_option

when *target* is client, this parameter specifies how the hs_playback_sample command treats periods of time when no data is available in the input sessions. Valid values are:

- skip (default) – when a time period contains no data, the hs_playback_sample command goes directly to the next time period containing data, rather than returning a sample having no data.
- show – the hs_playback_sample command returns a sample even for a time period where no data is available. During client playback, the column headers are returned with zero rows.

When *target* is file, gaps are not allowed.

target

specifies the target results of the playback session. Valid values are:

- client (default) – enables playback to the client.
- file – enables creation of a new session containing all the data specified by this hs_create_playback_session command and by subsequent hs_create_playback_view commands. Use the hs_initiate_playback command to create the new session. Setting *target* to file is not allowed if the *summarization_interval* parameter is raw or actual.

directory_name

when *target* is file, this parameter specifies the directory where the new files are created. This parameter is ignored when *target* is client. The default is the Historical Server home directory.

The operating system account that started Historical Server must have execute (search) and write permission on the specified directory.

protection_level

when *target* is file, this parameter specifies the permission level that the hs_initiate_playback command assigns to the data files for the newly created session. The protection level controls viewing the metadata in the control file (using the hs_list command) and creating playback sessions with the data (using the hs_create_playback_session command). Valid values are:

- private (default) – specifies that the new files will be password-protected. The files will be accessible only to the same account that created them or to the Historical Server superuser.
- public – gives unrestricted access to the data in the new session's files.
- null – implements the default value, private.

When *target* is client, this parameter is ignored.

script_type

when *target* is file, this parameter specifies whether you want the hs_initiate_playback command to create a script file for the newly created session. The script file contains SQL commands that create an Adaptive Server table for each playback view defined for the new session. Valid values are:

- no_script (default) – does not create a script file for the new session.
- sybase_script – creates a script file. The file is located in the Historical Server home directory. Its name is *sSessionId*, where:
  - *s* is a constant.
  - *SessionId* is assigned by Historical Server.
- null – implements the default value, no_script.

When *target* is client, this parameter is ignored.

delete_option

when *target* is file, this parameter specifies whether you want the hs_initiate_playback command to delete the input session files after successfully creating the new session. Valid values are:

- retain (default) – does not delete the files containing the data for the input sessions.
- delete – deletes all input session files, if creation of the new session is successful. The user must be either the Historical Server superuser or the owner of all the input sessions.
- null – implements the default value, retain.

When *target* is client, this parameter is ignored.

session_id[,session_id. . .]

specifies the unique identifiers for the input sessions to be played back. At least one *session_id* is ***required***. When more than one are used, they must be specified in the correct order according to their start times, with the earliest start time first. Playback skips over the sessions that are out of order. If *summarization_interval* is raw, then only one *session_id* is valid.

Historical Server assigns a session ID to a recording session when the recording session is defined. Use the hs_list command to find the session IDs that you want to play back.

If *target* is file, no gaps can exist between an input session's end time and the next input session's start time. Also, each session specified must have completed its recording.

If *target* is client, then each session must have started recording, but can be continuing to record when you submit the hs_create_playback_session command.

Examples

1 This example creates a playback session based on a single input session (session 7). The session is played back in its entirety, with no summarization or normalization, and without creating a new session.

```
hs_create_playback_session null, null, raw,
null, null, client, null, null, null, null, 7
```

2 This example creates a playback session based on three input sessions (sessions 4, 6, and 9). Only data collected between the time period from 9:00 to 5:00 is played back. The played back data is summarized at half-hour intervals. The playback is stored as a new session.

```
hs_create_playback_session "1996/5/3 9:00",
```

```
"1996/5/3 17:00", "0 00:30:00", disallow, skip,
file, null, public, no_script, retain, 4, 6, 9
```

Usage

- A connection to Historical Server can either be defining a recording session or defining and executing a playback session, but not both. After the successful execution of the hs_create_playback_session command, you must successfully execute a hs_terminate_playback command before starting to define a recording session or starting to define another playback session.
- The behavior of the playback session is different depending on the value of the *target* parameter:
  - If *target* is client, the client retrieves the data using the hs_playback_sample command. Playback can occur from recording sessions that are still in the process of recording.
  - If *target* is file, the client does not retrieve any data for this playback session; instead, Historical Server creates a new session and fills the data files for that new session with the data from this playback session. The accessibility of the new session is determined by the *protection_level* parameter. The user who invokes this command is the owner of the new session. The hs_initiate_playback command creates the new session.

    Playback is restricted to sessions that are finished recording as of when you submit the hs_create_playback_session command.
  - If *target* is file and multiple playback sessions are used as input, no time gaps are allowed. For example, if you collected data from 9 a.m. to 5 p.m. every day from Monday through Friday, you could not play back those five recording sessions to create a new, summarized, weekly session. However, if you eliminate the time gaps by collecting data from 9 a.m. to 9 a.m. every day from Monday through Friday, you could use the playback feature to create a new, summarized weekly session. Another way to eliminate the time gaps is to keep the 9 a.m. to 5 p.m. recording session, but add another set of recording sessions scheduled from 5 p.m. to 9 a.m. Use a longer sample interval for the off hours to reduce the volume of data collected.
  - The value of the *summarization_interval* parameter used in the hs_create_playback_session command affects the rules that the hs_create_playback_view command uses to define views.

hs_create_playback_view

Description

Specifies a view from the input recording sessions that you want to include in the playback session. Also specifies the data items in each view that you want to play back.

Syntax

hs_create_playback_view *view_name*,
[*data_item_name_1*, *data_item_stat_1*
[,*data_item_name_2*, *data_item_stat_2*]...]

Parameters

view_name

name of a view that you want to play back. The playback view name must match the name of a corresponding view in the input sessions. If there are two or more input sessions, then the view must exist in all of the input sessions and must contain the exact same data items and filters in all of the input sessions.

Only one playback view can be defined for a given view name. An attempt to create a second playback view with the same name fails.

Use the hs_list command to list the views in an input session and the data items in each view.

data_item_name_n

data items existing in the input view that you want to see in the playback view. (In some cases, you can introduce new data items not in the input views.) If the data item contains embedded spaces, surround it with quotation marks.

If you do not specify any data items, then the view is defined using all of the data items from the corresponding view in the input sessions. However, if a data item from that corresponding view is not valid in the playback view, an error results.

See Table C-2 on page 169 for more information about designing playback views.

data_item_stat_n

statistic types for the respective data items. The statistic types do not necessarily need to be the same as the statistic types used in the input view. See Table C-2 on page 169 to determine valid statistic types for data items in playback views.

Surround statistic types with quotation marks.

Examples

1 This example creates a playback view based on the view *device_view* from the input sessions. The playback view includes all of the data items from the input view, with the same combinations of data items and statistic types:

```
hs_create_playback_view device_view
```

2 This example creates a playback view that returns a timestamp and two data items from the input sessions:

```
hs_create_playback_view
device_view,"Timestamp","Value for
Sample","Device Name", "Value for Sample",
"Device Reads","Rate for Sample"
```

Usage

- The hs_create_playback_view command is valid only during the definition of a playback session; that is, it must be issued after a successful hs_create_playback_session command, but before a successful hs_initiate_playback command.
- A playback session must have at least one view.
- Appendix C, "Specifications for Defining Playback Views" explains the requirements for designing valid playback views, including valid combinations of data items and statistic types for playback views.

hs_create_recording_session

Description

Establishes a new recording session.

Syntax

hs_create_recording_session *monServerName*, *sample_interval* [,*dir_name*][,*start_time*][,*end_time*] [,*protection_level*][,*error_option*][,*script_type*], [*tab_delimited*]

Parameters

monServerName

name of the Monitor Server used to collect data from the Adaptive Server for which you want to collect historical data. The user name and password you used to connect to Historical Server must match a valid login account on the Adaptive Server being monitored by the Monitor Server you specify here.

sample_interval

determines the time between samples (in seconds). The shortest valid sample interval is one second; the longest valid sample interval is 86400 seconds, which is the number of seconds in one day.

dir_name

path name of the directory where the historical data files resulting from this recording session will reside. The operating system account that started Historical Server must have search (execute) and write permissions on this directory.

The default is the Historical Server home directory, specified in the -D parameter to the Historical Server start-up command.

start_time

date and time at which recording is to start, using the format:

```
year/month/day  hour:minute[:second]  [time zone]
```

The default is to start immediately. The following time zone options are available:

Parameter value	Explanation
EST	U.S. eastern time zone, standard time.
EDT	U.S. eastern time zone, daylight saving time.
CST	U.S. central time zone, standard time.
CDT	U.S. central time zone, daylight saving time.
MST	U.S. mountain time zone, standard time.
MDT	U.S. mountain time zone, daylight saving time.
PST	U.S. Pacific time zone, standard time.
PDT	U.S. Pacific time zone, daylight saving time.
-T	Middle European time zone, standard time.
MET DST	Middle European time zone, daylight saving time.
WET DST	Western European (Greenwich) time zone, daylight saving time
GMT	Greenwich mean time. This is equivalent to western European (Greenwich) time zone without regard to daylight saving time. All of the preceding time zone specifications, such as EST or EDT, can be supplied only in combination with dates and times when standard time or daylight savings time is in effect. GMT can be paired with any date and time specification.
GMT{+-} *hours_offset*	To specify any other time zone, where *hours_offset* is the number of hours that must be added to Greenwich mean time to derive the local time. The acceptable range of offset values is between +24 and -24 hours, inclusive. Fractional offsets such as +5.5 are valid.

If you do not specify a time zone, the local time zone of the Historical Server is used.

The *start_time* cannot be more that 31 days from the present time.

end_time
time at which recording is to stop. The default is to stop the recording session 24 hours after the *start_time*. If you specify an *end_time* and do not specify a *start_time*, the *start_time* defaults to the present.

protection_level
specifies whether the data files created in this recording session are password-protected or accessible to all users. The protection level controls viewing the metadata in the control file (using the hs_list command) and creating playback sessions with the data (using the hs_create_playback_session command).

- private (default) – specifies that the recorded data will be password-protected. The files will be accessible only by the same account that created them, or by the Historical Server superuser.
- public – specifies that access to the recorded data will be unrestricted.

error_option
specifies how you want Historical Server to handle non-fatal errors during a recording session. Valid values are:

- continue (default) – specifies that recording is to continue when non-fatal errors are detected. For example, when the Monitor Server fills all of its configured buffers while summarizing monitoring information, one or more non-fatal errors are sent to Historical Server, but it is still possible for Historical Server to collect the available information from the current sample and from future samples.
- halt – specifies that the recording session is to terminate when a non-fatal error is detected.

script_type
specifies whether you want Historical Server to create a script file that creates tables from your the recording session view definitions.

- sybase_script – creates a script file containing SQL commands that create an Adaptive Server table for each view in the recording session.
- no_script (the default) – does not create a script file.

tab_delimited

When adding this to the hs_create_recording_session, the default output file delimitoer HS is using will be ignored. Instead the TAB character will be used.

Examples

This example creates a recording session to capture data from the Monitor Sever called SERVER1_MON. The data is captured every 30 seconds and written to the data files in the */user/hist_dir* directory, starting now and ending on August 8, 1997 at 10:30 a.m. EDT (eastern time zone, daylight saving time). The resulting files have private restrictions. The recording session continues if non-fatal errors are detected. A script file that creates an Adaptive Server table for each recorded view is created.

```
hs_create_recording_session SERVER1_MON,30,
 /user/hist_dir,NULL,"97/08/08 10:30 EDT",
 private,continue, sybase_script
```

Usage

- Fatal errors include crashes and other connection failures with Adaptive Server or Monitor Server. A fatal error forces termination of the recording session, regardless of the value of *error_option*.
- You must issue this command before you define any views, alarms, or filters for the recording session. After you have issued this command and have defined at least one view, you can start recording with the hs_initiate_recording command.

hs_create_view

Description

Defines a collection of data items to be recorded during a session.

Syntax

hs_create_view *view_name*,
data_item_name_1, *data_item_stat_1*
[,*data_item_name_2*, *data_item_stat_2*]...

Parameters

view_name

user-defined name of the view. The view name must be constructed from the characters a – z, A – Z, 0 – 9, and the underscore character (_).

data_item_name_n

data items that participate in the view. If a data item contains embedded spaces, surround it with quotation marks.

See Table A-1 on page 107 for a list of available data items. See Table B-2 on page 128 for valid combinations of data items within views.

data_item_stat_n

statistic types of the respective data items. The statistic type must be surrounded with quotation marks. Valid statistic types are:

- "Value for Sample"
- "Value for Session"
- "Rate for Sample"
- "Rate for Session"
- "Avg for Sample"
- "Avg for Session"

Not all statistic types are valid for all data items. See Table B-3 on page 152.

Examples

This example creates a view called Page I/O that contains the Page I/O data item with a Value for Session statistic type and the Kernel Process ID data item with a Value for Sample statistic type.

```
hs_create_view PageIO,"Page I/O","Value for
Session","Kernel Process ID","Value for Sample"
```

Usage

- The order in which you specify the options determines the order in which the data items are stored in the recorded data file for the view.
- You must define at least one view for each recording session.
- If you intend to use the Historical Server playback features to examine data in a recording session, you might want to think about the playback view while designing the recording session view.

  For many data items, Historical Server can play back a data item using a different statistic type from the one used during recording. However, for some data items, Historical Server summarizes data by estimating values (as opposed to exact calculation), unless certain additional data items are included in the recorded data. See Table C-2 on page 169 to determine which data items are required for exact calculations.

hs_delete_data

Description

Deletes the historical monitoring files associated with one or more inactive recording sessions.

Syntax
hs_delete_data *low_session_id* [,*high_session_id*]

Parameters
low_session_id
unique identifier of the first recording session whose data is to be deleted.

high_session_id
unique identifier of the last in a range of sessions whose data is to be deleted. If omitted, only the session identified by *low_session_id* has its data deleted.

Examples
This example deletes historical monitoring files for any inactive recording sessions that have session IDs of 1 through 15.

```
hs_delete_data 1,15
```

Usage
- The only users who can delete historical data files are:
  - Historical Server superuser
  - Users who connected to Historical Server using the same user name and password that defined the recording session
- Only inactive recording session files are deleted. Files associated with active recording sessions are not deleted, even if their session IDs fall within the range specified in the hs_delete command.
- An inactive recording session is one that was terminated because the *end_time* was reached or because the hs_terminate_recording command was issued.

hs_initiate_playback

Description
Specifies that you are finished defining a playback session and ready to perform the playback. If the playback target is to a file, this command starts the playback. If the playback target is to a client, this command initializes playback so that the hs_playback_sample command with a default *step* returns the first sample in the playback session.

Syntax
hs_initiate_playback

Usage
- The hs_create_playback_session command and one or more successful hs_create_playback_view commands must precede the hs_initiate_playback command.

- If no views were created before the hs_initiate_playback command, then hs_initiate_playback returns an error. The user may still attempt an hs_create_playback_view command, or may cancel the playback session with the hs_terminate_playback command.

hs_initiate_recording

Description

Specifies that the definition of a recording session is complete and requests that recording be started or scheduled to start at the recording session's *start_time*.

Syntax

hs_initiate_recording

Usage

- The hs_create_recording_session command and one or more hs_create_view commands must precede the hs_initiate_recording command.
- One or more hs_create_alarm and hs_create_filter commands may also precede the hs_initiate_recording command.
- After you execute the hs_initiate_recording command, you may later cancel the session, but you cannot add any more views, filters, or alarms to the description of this recording session.

hs_list

Description

Lists information about past and present recording sessions.

Syntax

hs_list *level* [,*restriction*]

Parameters

level

specifies the level of detail returned for each recording session. Its value can be sessions, views, data_items, alarms, filters, or summarization_data_items.

restriction

Selects a subset of the data available about recording sessions for a given *level*.

If level is	Then restriction is

sessions	• One of the following:
	  • active – limits the returned data to currently active recording sessions of this Historical Server instance.
	  • inactive – selects only completed recording sessions.
	  • latest – selects only the recording session (if any) most recently initiated on the current client connection.
	• If *restriction* is omitted, all active and inactive recording sessions that you have permission to access are listed.

views	• A session ID whose views you want to list. • If *restriction* is omitted, the views of all recording sessions that you have permission to access are listed.
data_items, alarms, filters, or summarization_data_items	• A session ID and, optionally, a view name whose data items you want to list. The syntax is: *session_id* [,*view_name*] *session_id* limits the listing to data items that participated in a single session. If specified, *view_name* limits the listing to data items that appeared in a single view defined for the session. • If *restriction* is omitted, the data items that belong to all views defined for all active and inactive recording sessions that you have permission to access are listed.

Examples

1 This example lists all active sessions for this Historical Server instance:

```
hs_list sessions,active
```

2 This example lists all alarms that have been defined for data items in the Page I/O view of the session whose ID is 10:

```
hs_list alarms,10,PageIO
```

Usage

- The hs_list command returns the following fields as an integer datatype:
  - Session ID
  - Sample interval
  - Alarm count

    - Filter count
- The hs_list command returns the following field as a float datatype:
    - Alarm value
- All other fields are returned as character strings.
- If a session was recorded with the private protection level and the current user is not the Historical Server superuser, the current user's name and password are verified against the name and password of the user who recorded the session before that session is made visible.

Description of returned data

The data returned by hs_list depends on the value of *level*:

- sessions causes a single row to be returned for each session, with the following columns:
    - Session ID (a unique identifier for the session)
    - Status of the session:
        - *active*
        - *inactive*
        - *active remotely or inactive* – status of the session cannot be determined unambiguously. Sessions that are currently active in another instance of Historical Server that is using the same home directory as the current instance fall into this category. (For more information about running multiple instances of Historical Server and sharing control files, see "Configuring multiple instances of Historical Server" on page 23.) Sessions that have already ended because of an abnormal termination of the Historical Server instance that was running them, and whose intended end times have not yet been passed, are also in this category.
    - Name of the user who initiated the recording session
    - Name of the Adaptive Server monitored
    - Name of the Monitor Server used
    - Start date and time of the recording session
    - End date and time of the recording session
    - Directory containing the recorded data

- Sampling interval used

  If this value is 0, it means that the session was created by playback with a *summarization_level* of entire. (The entire session is represented in one sample, and there is no sample interval.)

- Error option used (continue or halt on non-fatal error)

- views produces one row for each view, with the following columns:
  - Session ID
  - View name
- data_items produces one row for each data item defined in a view, with the following columns:
  - Session ID
  - View name
  - Data item
  - Data item statistic type
  - Number of alarms defined for the data item
  - Number of filters defined for the data item
  - One of the following keywords:
    - recorded – indicates that the data is from the original recording session or preserved from that original session.
    - summarized – indicates that the data was summarized at a different sample interval from that of the original session.
    - estimated – indicates that the data was estimated, rather than calculated accurately, during summarization.
- alarms produces one row for each alarm defined for a data item in a view, with the following columns:
  - Session ID
  - View name
  - Data item
  - Data item statistic type
  - Alarm action (log or execute)

    - File name for log file or file to execute
    - Alarm value
- filters produces one row for each filter for a data item in a view, with the following columns:
    - Session ID
    - View name
    - Data item
    - Statistic type
    - Filter type (eq, neq, range, or top)
    - Value specified (returned as a single string)
- summarization_data_items produces one row for each combination of data item and statistic type that can be requested for non-raw playback from a view, with the following columns:
    - Session ID
    - View name
    - Data item
    - Statistic type
    - One of the following keywords:
        - recorded – indicates that the data for this data item, if played back, would be the data from the original recording session, or data preserved from that original session.
        - summarized – indicates that the data for this data item, if played back, would be available in summarized form.
        - estimated – indicates that the data for this data item, if played back, would be available only in estimated form.

hs_playback_sample

Description — Plays back a sample when the playback *target* is client. The *target* is defined in the hs_create_playback_session command.

Syntax — hs_playback_sample [*step* [, *retry_count*]]

Parameters

step

specifies the sample to send to the client, relative to the sample sent in the most recent hs_playback_sample command for the current playback session. Valid values are 0 or positive numbers.

The default is +1, which sends the next later sample. At the beginning, a *step* of +1 returns the first sample. Zero resends the most recently sent sample. A *step* of +2 sends the next later sample after the one that a *step* of +1 would have sent, and so on. An attempt to display a sample later than the last sample returns an error message.

The hs_initiate_playback command initializes playback so that the default (+1) sends the first sample in the playback session.

The definition of what constitutes a sample is affected by the *summarization_interval* and *missing_data_option* parameters to the hs_create_playback_session command.

If the playback session was created with a *summarization_interval* parameter of actual, entire, or a user-specified interval, and if any view in the playback session contains a data item with a statistic type of "Value for Session" or "Rate for Session", then the only permissible value for *step* is +1, the default.

retry_count

specifies the number of times that the Historical Server will retry, at one-second intervals, to read data from a historical data file of a recording session that is still recording at the time of playback.

The default is zero, that is, no retries are attempted and an error is returned when playback reaches an end-of-file condition while reading a historical data file.

Examples

1 This example plays back the next sample:

```
hs_playback_sample
```

2 This example skips over one sample and plays the next sample. If recording is ongoing, Historical Server retries up to 30 times to retrieve another sample if the end-of-file is reached.

```
hs_playback_sample 2, 30
```

Usage

Format of returned data

The hs_playback_sample command returns data in the form of tables. It returns one table for each playback view. The tables are arranged in the same order in which you submitted the hs_create_playback_view commands that created the views.

The columns in each table correspond to the data items in the corresponding playback view. The columns are arranged in the order in which the data items were listed when you defined the playback view. See Table B-3 on page 152 to determine the datatype of each column.

Each row represents a different combination of key data items in the view. If the view contains no key data items, then the table returns a single row reflecting server-level data.

When there is no activity to report, some views return a row with zero values and other views omit the row. The rules controlling whether or not a row with zero values appears in a view are:

- Views with server-level data items always return a row, even when there is no activity to report.
- Views that contain the key data items Process ID, Object ID, or Procedure ID omit the row when there is no activity to report. See "Views with Process ID" on page 80 for more explanation about when a Process ID is returned.
- Views that contain keys other than those listed in the previous item return rows even when there is no activity.

If playback is summarizing data, a row is returned for a combination of keys if *any* sample in *any* of the input sessions contains the same combination of keys.

If an integer data item overflows, Historical Server returns the largest valid number in the data item, and returns an information message to the client.

Views with Process ID

When a server process terminates, Adaptive Server can reuse its Process ID for a new process. Therefore, the Process ID data item is not guaranteed to uniquely identify a process. The Kernel Process ID data item, however, uniquely identifies a process.

Views that include Process ID return rows as follows:

- Recording session views (and hence, raw playback views) return a row only for Process IDs representing processes that exist at the end of a sample interval. If a server process terminates in the middle of a sample interval, a row is not returned for its Process ID.

- Summary playback views do not require the server process to exist for the duration of the playback sample. Summary views return rows for all server processes included in any of the input views. However, since Process ID is not guaranteed to be unique, the Kernel Process ID must be included in any summary view that includes Process ID to ensure uniqueness of the key. Otherwise, the summary could erroneously combine data from two different processes.

Gaps in data within the playback session

The *start_time* and *end_time* parameters to the hs_create_playback_session command define the period of time that the playback session covers. When the *target* parameter to the hs_create_playback_session command is client, there might be intervals of time between the playback session's start and end times for which no data is available in any of the input sessions.

Note When *target* is file, no gaps are allowed in the specified input sessions, so there will not be gaps in the playback session.

Gaps in available data might occur when:

- The input sessions specified for the playback session are not contiguous. For example, if a series of input sessions covered the time period between 9 a.m. and 5 a.m., Monday through Friday, data gaps would exist between 5 p.m. each evening and 9 a.m. the next morning. Larger gaps would exist between Friday evening and Monday morning.
- The playback session does summarization and has a start or end time beyond when data is available from the input sessions.

When the target of playback is a client, Historical Server handles gaps according to the value of the *missing_data_option* parameter to the hs_create_playback_session command:

- skip (the default) – when a time period contains no data, the hs_playback_sample command goes directly to the next time period containing data, rather than returning a sample having no data.
- show – the hs_playback_sample command returns a sample even for a time period where no data is available. No data is returned (the length of each data table returned is zero).

For the client to receive a timestamp in this case, there must be a playback view containing *only* the data item Timestamp or Timestamp Datim. A view containing any other data items returns zero rows.

The playback session's summarization level affects the number of empty samples returned during playback:

- if *summarization_level* is raw or actual, the entire gap is represented with one sample. For example, a gap in data between the hours of 5 p.m. and 9 a.m. would be represented by one sample.
- if *summarization_level* is a time interval, such as 1/2 hour (specified as `"0 00:30:00"`), the gap is represented with a sample for each time interval. For example, if the time interval is specified as 1/2 hour, a gap in data between the hours of 5 p.m. and 9 a.m. would be represented by 32 empty samples.
- if *summarization_level* is entire, gaps are ignored.

hs_shutdown

Description

Shuts down Historical Server.

Syntax

hs_shutdown [*wait_option*]

Parameters

wait_option

specifies whether termination should occur immediately or wait for active connections to close. Valid values are:

- no_wait – causes Historical Server to shut down immediately and terminates any connections (including those performing playback) and active recording sessions. The active recording sessions and playback sessions that are creating new sessions are shut down in a controlled manner, which prevents corruption of the control file or any historical data files.
- wait (the default) – defers shutdown until all active recording sessions (including those performing playback) complete and any other active connections are closed. No new connections are accepted during this time.

Examples

This example shuts down Historical Server immediately and terminates any connections and active recording sessions.

```
hs_shutdown no_wait
```

Usage

- If you use the hs_shutdown command without the no_wait option and then want to cancel the request, you can send an interrupt to Historical Server to cancel the shutdown request. A DB-Library application calls the dbcancel() function and a CT-Library application calls the ct_cancel() function to cause such an the interrupt. The isql command generates one of these calls when you send it an interrupt signal, which usually is done by typing Ctrl+C.
- You can request an immediate shutdown of Historical Server at any time by sending one of the following four signals:
  - SIGQUIT (quit)
  - SIGINT (interrupt)
  - SIGABORT (abort)
  - SIGTERM (terminate)

  Reception of one of these signals at any time (including while waiting for the hs_shutdown command to complete) is equivalent to issuing the hs_shutdown no_wait command.

Warning! Sybase strongly recommends that you do not use the kill signal (SIGKILL) to shut down Historical Server. The kill signal does not permit any controlled cleanup to be done and may result in data loss.

- If any active sessions were terminated during the shutdown, Historical Server writes information about the sessions to its log file. Active sessions are recording sessions or playback sessions in the process of creating a new recording session.
- If a superuser was specified in the Historical Server start-up command, the only user who can shut down Historical Server is the superuser. If a superuser was not specified, any user can shut down Historical Server.
- See Chapter 3, “Starting and Stopping Historical Server” for more information about stopping Historical Server.

hs_status

Description

Obtains status information.

Syntax

hs_status *option*

Parameters

option

one of the following specifications:

- directory – displays the name of the Historical Server home directory, specified by the -D parameter to the Historical Server start-up command.
- superuser – displays the name of the Historical Server superuser, specified by the -U parameter to the Historical Server start-up command (or NULL if this parameter was omitted).
- interfaces – displays the path name of the *interfaces* or *sql.ini* file in use by Historical Server.
- max_connections – displays the maximum number of concurrent client connections permitted.
- logfile – displays the name of the Historical Server log file.
- version – displays the version string and copyright information of Monitor Historical Server.
- activity – displays the following information about the current level of activity on Historical Server:

Number of connections	Connections to Historical Server, including the current connection
Number of active recording sessions	Sessions created with hs_create_recording_session and initiated with hs_initiate_recording. Includes sessions whose start times have not yet been reached.
Uninitiated recording sessions	Sessions created with hs_create_recording_session but no matching hs_initiate_recording was executed.
Number of active playback sessions	Sessions created with hs_create_playback_session and initiated with hs_initiate_playback.
Number of uninitiated playback sessions	Sessions created with hs_create_playback_session but no matching hs_initiate_playback was executed.

The activity information is relevant when stopping Historical Server because Historical Server does not shut down while other connections exist or any sessions are active, unless the no_wait option for the hs_shutdown command is explicitly requested.

Examples

This example displays the name of the Historical Server home directory:

```
hs_status directory
```

Usage

The hs_status command returns the max_connections field and all of the activity fields as integer datatypes. All other fields are returned as character strings.

hs_terminate_playback

Description

Terminates the definition of a playback session and the actual playback.

Syntax

hs_terminate_playback

Usage

- This command is valid only after a successful hs_create_playback_session command.
- Once you issue a successful hs_create_playback_session command, you must issue hs_terminate_playback before you can start defining any additional playback sessions or recording sessions.
- Unlike a recording session definition, playback session definitions are not stored in the Historical Server control file.

hs_terminate_recording

Description

Terminates the definition of a recording session, cancels the scheduled start of a recording session, or terminates a recording session in progress.

Syntax

hs_terminate_recording *session_id* [,*delete_option*]

Parameters

session_id

identifies the session you want to terminate. Historical Server assigns a session ID when a session is defined with hs_create_recording_session. Use the hs_list command to view session IDs.

delete_option

specifies whether to delete the files associated with the recording session, if recording is in progress.

- delete – deletes the files.
- retain (the default) – does not delete the files. This parameter is ignored when you are cancelling a recording session that was not initiated.

Examples

This example terminates the active recording session with identifier 5 and then deletes all files associated with session 5.

```
hs_terminate_recording 5,delete
```

Usage

- If you initiate a recording, you are the session owner.
- To terminate a recording session, you must be the session owner or the Historical Server superuser.
- The hs_terminate_recording_session command can cancel the definition of a recording session that was created in the current connection but never initiated. This allows you to define another recording session or to define a playback session using the current connection.
- If you attempt to terminate an inactive recording session, this command returns an error.

CHAPTER 5

Data Files and Output Options

Overview of Historical Server data files

The topics in this section are:

- Description of Historical Server files
- Permissions on files
- General file format

Description of Historical Server files

The Historical Server files are:

- Control file – all recording sessions for the same Historical Server use the same control file. For information about multiple Historical Server instances and control files, see "Configuring multiple instances of Historical Server" on page 23.
- Error message file – each recording session has its own error message file. The file is empty if no errors are logged for the session.

- Data file – each view in a recording session is represented by a data file. The data file holds all of the samples collected for the view during the session.
- Script file – when you define a recording session using the hs_create_recording_session command, the *script_type* parameter can specify creation of a SQL script file. The script file defines a Adaptive Server table for each view in the recording session.

You can use the bulk copy (bcp) utility to populate existing Adaptive Server tables. If you are a UNIX user, you can put subsets of the fields into separate Adaptive Server tables if you first invoke the cut utility to strip the desired columns from the file.

Permissions on files

All of the historical monitoring data files for a given recording session are stored in the same directory. This directory is, by default, the directory specified by the -D parameter to the Historical Server start-up command. You can override this default for a session when you create the session.

The user who starts Historical Server is the owner of the files that Historical Server creates. The owner has read and write permission on the historical monitoring files. If the data contained in an historical data file or error message file is public rather than private, all users and groups have read permission.

General file format

All files use newline characters as end-of-record markers. The fields in each file are variable-length and are separated by commas. All data is stored in ASCII format, which is compatible with the Sybase bulk copy (bcp) utility.

Control file

The *hs.ctl* control file resides in the Historical Server home directory. The control file maintains information about all recording sessions, past and present, known to Historical Server. See "The Historical Server control file and home directory" on page 9 for more details about how Historical Server uses the control file.

Use the Historical Server hs_list command to gain access to the information in the Historical Server control file. Do not edit the control file. You might inadvertently corrupt the file.

Regardless of the editor you use, do not open and then save this file. This is true especially if Historical Server is running on Windows. Unlike the other files created by Historical Server, the control file is not a standard-format Windows text file. Lines of text in the control file are terminated only with new-line characters, rather than the usual carriage-return/newline pairs. The editing program may corrupt the file by embedding unwanted carriage-return/newline pairs into the text.

The control file contains the following types of records:

- Header record
- Session control record
- Data item control record
- Alarm control record
- View control record
- Filter control record

Header record

The first record in the control file is a header record that contains the following six integer fields:

- An "update in progress" flag
- A byte offset in the file of the information being updated
- A unique session ID to be assigned by Historical Server to the next recording session
- A copy of the unique session ID for use in file corruption detection and recovery

- The session ID of the session whose control information was most recently written to the control file
- The number of session entries in the control file that are no longer valid because one or more data files or error message files they refer to were deleted

Following this header record are a number of records of different types that describe all of the recording sessions, past and present, known to Historical Server. These records appear in hierarchical sequential order in the control file, where the hierarchy of record types in top-down order is:

- Session control
- View control
- Data item control
- Alarm control and filter control

For example, a session that consisted of two views, each with two data items that had one alarm and one filter apiece, would be represented by the following hierarchy of control records, stored in the following order:

```
session 1
    view 1
        data item 1
            alarm 1
            filter 1
        data item 2
            alarm 2
           filter 2
    view 2
        data item 3
            alarm 3
            filter 3
        data item 4
            alarm 4
            filter 4
```

The contents of the various control record types are described in the next sections.

Session control record

The session control record contains:

- Record identifier (the word “session”).
- Status of the session. The status can be active, inactive, or invalid. A status of invalid applies if any data file or error message file belonging to the session has been deleted or if the session was terminated before its start time was reached.
- Byte offset to the next session record in the control file, from the start of this record.
- Session ID, a number that uniquely identifies the session among all sessions that appear in this control file. Session IDs are integers assigned sequentially in increasing order.
- User name of the client who requested that the recording be made.
- Start date and time of the recording session.

  The start and end times in the control file reflect as closely as possible when a recording session actually started and ended. These actual times might differ slightly from the *start_time* and *end_time* parameter values specified in the command. For example, *start_time* might be scheduled as 1 p.m., but because of a busy system, the session might actually start at 1:02 p.m.
- End date and time of the recording session.
- Name of the Adaptive Server monitored.
- Name of Monitor Server that participated in the recording session.
- Path name of the directory in which the monitoring files for the session are stored.
- Name of the error message file for the session.
- Sample interval, in seconds, used during recording.

  If this value is 0, it means that the session was created by playback with a *summarization_level* of entire. (The entire session is represented in one sample, and there is no sample interval.)
- Protection level in effect (public/private).
- Error option (continue/halt if nonfatal error during recording).
- Type of script file created (sybase_script/no_script).
- Encrypted password of the client that created the recording session.
- If the session was created as a result of playback, the keyword summary.

There is one session control record in the file for each recording session. The first view control record for the session follows immediately.

View control record

The view control record contains:

- Record identifier (the word “view”)
- View name
- Name of the data file for the view

One view control record exists for each view defined on a recording session. One or more data item control records that identify the data items in the view follow each control record. Alarm control and filter control records are interspersed between the data item control records as appropriate.

Data item control record

The data item control record contains:

- Record identifier (the word “dataitem”).
- Data item name.
- Statistic type.
- In sessions created from playback, the keyword estimated indicates a data item whose values were obtained by estimation from data items in other sessions.

One data item control record exists for each data item that appeared in the current view’s definition. If alarms or filters or both are applied to the data item, their control records follow that data item’s control record. All of the alarm control records for the data item precede the filter control records for the same data item.

Alarm control record

The alarm control record contains:

- Record identifier (the word “alarm”)

- Alarm action
- Alarm action data (file name, possibly followed by a list of parameters)
- Alarm datatype
- Alarm value

One alarm control record exists for each alarm that is defined for the current data item. All of the alarm control records for the data item are grouped ahead of the first filter control record for the data item.

Filter control record

The filter control record contains:

- Record identifier (the word “filter”)
- Filter type
- Filter datatype
- Value specification (one or more fields, separated by commas, in the same format as is used by the hs_create_filter command)

One filter control record exists for each filter that was applied to the current data item.

Data file

A data file contains the data for a single view recorded during a session. The file name is *dsessionid_viewnumber_viewname* where:

- *sessionid* is the unique identifier for the session.
- *viewnumber* is a number greater than or equal to one that reflects the order in which views were defined for the session.
- *viewname* is the user-supplied name for the view, truncated if necessary to conform to the file-naming conventions of the current platform. On the Windows platform, the file name is limited to eight characters.

One record in the file exists for each unique combination of data item key field values that occurred in each sample taken.

Each such record holds:

- Date and time of the sample

  This timestamp is based on the local time in effect at the Adaptive Server that was monitored

- Values of all of the data items sampled

If no data is returned for this view during a given sample interval, a placeholder record is written to the historical data file. This record contains the timestamp of the sample, followed by a comma-separated list of empty values. That is, the timestamp field is followed by as many commas as are needed to separate the remaining data items of the view, but no values are stored for those data items. This format is acceptable to the Bulk Copy utility, provided that the columns into which the data items are being copied will accept nulls.

Error message file

Any error messages that are received while recording samples for a session are stored in a separate error message file. This file's name is *eSessionId* where:

- *e* is a constant.
- *SessionId* is the unique identifier for the session.

Each record in this file contains the following fields:

- Date and time of the sample that occasioned the error message. This timestamp is based on the local time in effect at the Adaptive Server that was monitored. The timestamp field may be empty if the error prevented Historical Server from retrieving the sample timestamp.
- Error number
- Error severity
- Error state
- Error source:
  - 1 – unknown
  - 2 – Historical Server
  - 3 – CT-Library
  - 4 – Adaptive Server

    - 5 – Monitor Server
- Error message text

For more information about error message fields, refer to the *Reference Manual*.

Script file

If the value of the *script_type* parameter of the hs_create_recording_session command is sybase_script, a script file is created for the session. The name of the script file is *ssessionid* where *sessionid* is the unique identifier for the session.

Script file table names

The script file contains the text of create table commands, one command per view defined for the session. The name of each table is *sSessionId_viewnumber_viewname* where:

- *s* is a constant.
- *SessionId* is the unique identifier for the session.
- *viewnumber* is a number greater than or equal to one that reflects the order in which views were defined for the session.
- *viewname* is the user-supplied name for the views, truncated if necessary to conform to the limitations on the lengths of Adaptive Server identifiers.

Script file table column names

The first column defined for each table contains the sample timestamp field. It is always named “Timestamp” and has the datatime datatype.

The names of the remaining table columns are derived from the data items that they represent. Each column name is *dataitemname_stattype* where:

- *dataitemname* is a compressed form of the data item’s name.
- *stattype* is an abbreviated form of the data item’s statistic type.

Each column is assigned a datatype that is appropriate for the type of information that the data item represents. All columns except the sample timestamp column can contain null values.

Passing script file commands

You can pass the create table commands in the script file to Adaptive Server by using the isql utility's :r command. After the table is defined with the :r command, you can use the Bulk Copy utility to import the view data files into the tables. For a Bulk Copy example, see "Bulk copy example" on page 97.

Script use example

Use the following isql command to connect to the Adaptive Server in which you want to store the recorded monitoring data:

```
isql -Uusername -Ppassword -Sserver
```

where *username* is the name of a login account in that Adaptive Server, *password* is the password of that login account, and *server* is the Adaptive Server name.

Execute the following isql commands:

```
use database
go
```

where *database* is the name of the database that will contain the new tables.

```
:r scriptfile
go
```

where *scriptfile* is the path name of the script file that was created by the Historical Server.

A table now exists in the Adaptive Server for each view defined for the recording session.

Bulk copy example

This section describes how to pass historical data files to the Bulk Copy (bcp) utility to populate Adaptive Server tables.

To store recorded data in an Adaptive Server table by using bcp, you must first create a table for each view defined for the session. In the table, create a separate column for each of the data items that comprise the view. The first column for each table, which stores the date and time of the sample, must be of the datetime datatype.

If no data is returned for a view during a sample interval while Historical Server is recording, a placeholder record is written to the view's data file. This record contains the timestamp of the sample followed by a comma-separated list of empty values. For this reason, it is recommended that all but the first column be defined to allow nulls.

You may simplify the process of creating tables by having the Historical Server create a script file for you. Use the script_type parameter of the hs_create_recording_session command.

After you create the tables on Adaptive Server, you can use the bcp utility to import data from the view data files to their respective tables. Run the following command once for each data file for which you want to import the data into an Adaptive Server table:

```
bcp [[database_name.]owner.]table_name in
view_data_file -c [ -e errfile] [-U username]
 [-P password] [-S sqlserver] -t,
```

where:

- *database_name* is the name of the database where the tables to hold the imported data are located.
- *owner* is the name of the table's owner (presumedly your own user name in the database that contains the table).
- *table_name* is the name of the database table into which the data from the view's data file is to be imported.
- *view_data_file* is the name of the data file for the view.
- *errfile* is the name of an error file in which any rows that could not be transferred to the database table are to be stored.
- *username* is the login name to Adaptive Server.
- *password* is the password of the *username*.

- *sqlserver* is the name of Adaptive Server.
- *-t*, specifies the column terminators as being commas.

The default row terminator when using the -c option is a \n (new line) character used by data files. Therefore, you do not need to specify the -r option. If you do specify the record terminator option (-r), be sure to avoid the special significance the backslash (\) character has by including another backslash, i.e., -r\\n.

For more information about bcp, see the *Utilities Guide*.

Example

Suppose you want to record, once every second, the reads, writes, and total I/O per device on an Adaptive Server that is monitored by the SMS1100 Monitor Server. You can use the following steps to record the data, create a table to accept the data, and then use the bcp utility to import the data into the table. In this example, the view data file that is created by Historical Server for the view is *d1_1_device_io*.

1 Use isql to log in to Historical Server.

2 Record the data by using the following commands:

```
hs_create_recording_session SMS1100, 1
go
hs_create_view device_io,
"device name", "value for sample",
"device reads", "value for sample",
"device writes", "value for sample",
"device i/o", "value for sample"
go
hs_initiate_recording
go
```

3 Use isql to log in to Adaptive Server. Because the recording session contains only one view (“device_io”), create just one table in the pubs2 database to store data for that view by using the following Transact-SQL commands:

```
use pubs2
go
create table device_io
(
sample_time          datetime         not null,
```

```
device_name          varchar (255)    null,
device_reads_val     int              null,
device_writes_val    int              null,
device_io_val        int              null,
)
go
```

4 After the recording session is complete, you can use bcp to import data from the view's data file into the database table. Use the following commands:

```
bcp pubs2..device_io in d1_1_device_io -c
 -e d1_1_device_io.err -Uusername
 -Ppassword -Sserver -t
```

Cut utility example

If you are on a UNIX machine, you can put subsets of the fields from data files into separate Adaptive Server tables if you first invoke the cut utility to strip and save the desired columns from the file. Use the following command:

```
cut -flist -d, view_data_file > output_file_name
```

where:

- *list* is a comma-separated list of the field numbers that you want to cut and save from the original view data file. The first field has field number 1.
- *view_data_file* is the name of the file where all of the recording data is stored.
- *output_file_name* is the name of the file where the stripped data is to be stored.

After using cut, you can use the bcp utility to import the data in the *output_file_name* into an Adaptive Server table. Historical Server might return lines to the *view_data_file* that are similar to:

```
1995/3/25 10:32:39,master,0,0,0
1995/3/25 10:32:39,sysprocsdev,0,0,0
1995/3/25 10:32:39,ANewDBDevice,0,0,0
1995/3/25 10:32:39,ATestDevice,0,0,0
1995/3/25 10:32:40,master,0,0,0
1995/3/25 10:32:40,sysprocsdev,0,0,0
1995/3/25 10:32:40,ANewDBDevic,0,0,0
1995/3/25 10:32:40,ATestDevice,0,0,0
```

Suppose that you are interested in only the total I/O for each device during the sample. You could execute the following command to cut out the data you want by executing the following command:

```
cut -f1,2,5 -d, d1_1_device_io > d1_1_device.io.new
```

where field 1 is the date and time of the sample, 2 is the second field, which is the device name, and 5 is the fifth field, which is the device I/O value.

Now you can import the *d1_1_device.new* file into an Adaptive Server table as described in the previous section, "Bulk copy example" on page 97.

For more information about the cut utility, refer to the UNIX reference pages on your system.

Enabling Historical Server to output monitoring data to a database

In addition to saving Historical Server output to a file, you can send the output from the Historical Server directly to a database. When you save your data to a database, it is easier to analyze it.

When data is stored in a database, output from each Historical Server view is stored in a seperate table. Changes to the Historical Server view definition will be automatically reflected in the table definition.

Setting up the receiving Adaptive Server

Before you enable Historical Server to send it's monitoring data to an Adaptive Server, you must select that Adaptive Server. Once you select an Adaptive Server to receive Historical Server monitoring data, you must:

1. Create a database for the Historical Server to use for storing monitoring data. The default for the name of the database is hs_monitoring. If you choose to have a different name for the database, create that named database, then change the database name in the hs_directload.sql script.
2. Run the install script hs_directload.sql on the database. The install script creates two catalog tables, sessions and views, and the stored procedure sp_hs_dboutput.

Starting Historical Server

In order to enable Historical Server to send it's monitoring data to an Adaptive Server database, you must specify the following:

- The destination for the monitoring data from Historical Server to a specific Adaptive Server and database, rather than a flat file.
- The user name and password to connect to the output server if they are not the same as those used by the Historical Server to connect to the monitored server. If the user name and the password are not specified, they default to the ones specified in the -U and -P parameters on the command line.

The target Adaptive Server and database for the historical data must be available when Historical Server is started.

The command line syntax is:

histserver -U<*user name*>
- P<*password*> -D<*output dir*>
-I<*log file*> -I<*interfaces file*>
[-d<*delimiter*>] [-O<*ASE name*>] [-o<*DATABASE name*>] [-f]
-u<*outputASE user name*> - p<*outputASEpassword*>

- -O *ASE name* – the name of the target Adaptive Server.
- -o *DATABASE name* – the database name to which the monitoring data is sent, if it is not hs_monitoring, identify the database name on this option.
- -u *outputASE user name* – login name for the connection to the Adaptive Server with the output.
- -p *outputASEpassword* – password for the login name for the output Adaptive Server.
- -f – must be used when the -O option is specified if you want Historical Server to send data to files in the output directory as well as the database.

You must have access to Historical Server and update permissions to the target database.

In the following example, the -O and -o parameters are specified and the output database is called hs_monitoring:

```
$SYBASE/ASE-12_5/bin/histserver -D$SYBASE -Sajax_hs
-l$SYBASE/ajax_hs.log -Usa -Pnorthstar -Oajax
-ohs_monitoring
```

Viewing the data

Data storage

The data from Historical Server is in the same format as defined when the view was created on the server. To store the data, Historical Server creates two system tables:

- Sessions table provides a record of every recording session that used the output database.
- Views table lists the views that were used by each recording session.
- In addition to one table for each view that is output to the database, there are two system tables created in the Historical Server output database. These tables are:
  - The sessions table provides a record of every recording session that has used the output database.
  - The views table lists the views that were used by each recording session.

The following example shows the possible contents of the sessions table table:

```
1> select * from sessions
2> go

DbSessionId HsSessionId HsName HsAseName StartDate EndDate
-----------------------------------------------------------------------------
0           0     ajax_hs    ajax    Jul 22 2005 11:26AM    Jul 22 2005 11:45AM
1           14    ajax_hs    ajax    Jul 23 2005 12:25PM    Jul 23 2005  6:10AM
2           15    ajax_hs    ajax    Jul 24 2005 11:41AM    Jul 24 2005 11:59AM
3           16    ajax_hs    ajax    Jul 25 2005 11:56AM    Jul 25 2005 12:09PM
4           19    ajax_hs    ajax    Jul 26 2005 10:59AM    Jul 27 2005 11:25AM
5           23    ajax_hs    ajax    Jul 27 2005 11:26AM    Jul 27 2005 11:36AM
```

The following example shows the possible contents of the views table:

```
1> select * from views
2> go

DbSessionId ViewName
----------- ------------------------------
1           stored_procs
1           connections
2          stored_procs_shutdown_test
```

```
2          connections
3          stored_procs_shutdown_test
3          connections
4          connections
5          stored_procedure_activity
5          connections
5          process_activity
```

The structures of the view data output tables are similar to the structures of the output data files, where:

- First column is the monitoring session ID.
- Second column is the monitored server name.
- Third column is a date/timestamp.
- Subsequent columns per data item specified in the view definition.

This structure, for the date and timestamp, and the data items, is identical to the structure defined in the DDL scripts Historical Server provides when a bulk copy is executed on Historical Server data files into another Adaptive Server.

Note viewname is the name for the view, truncated if necessary to conform to the file-naming conventions of the current platform. On Windows platforms, the file name is limited to thirty characters.

In the following example, the query reports the sessions that were active on July 27, 2005 and the views that were used by those sessions:

```
1> select HsAseName, HsName, StartDate, EndDate, ViewName
2> from sessions, views
3> where sessions.DbSessionId = views.DbSessionId
4> and StartDate >= 'July 27, 2005'
5> and EndDate < 'July 28, 2005
6> go

HsAseName HsName    StartDate            EndDate              ViewName
------------------------------------------------------------------------------
tribble  tribble_hs Jul 27 2005 10:59AM  Jul 27 2005 11:00AM  connections
tribble  tribble_hs Jul 27 2005 11:00AM  Jul 27 2005 11:05AM  sproc_activity
tribble  tribble_hs Jul 27 2005 11:00AM  Jul 27 2005 11:05AM  connections
tribble  tribble_hs Jul 27 2005 11:17AM  Jul 27 2005 11:24AM  sproc_activity
tribble  tribble_hs Jul 27 2005 11:17AM  Jul 27 2005 11:24AM  connections
tribble  tribble_hs Jul 27 2005 11:25AM  Jul 27 2005 11:25AM  sproc_activity
tribble  tribble_hs Jul 27 2005 11:25AM  Jul 27 2005 11:25AM  connections
```

```
tribble  tribble_hs Jul 27 2005 11:26AM  Jul 27 2005 11:36AM  sproc_activity
tribble  tribble_hs Jul 27 2005 11:26AM  Jul 27 2005 11:36AM  connections
tribble  tribble_hs Jul 27 2005 11:26AM  Jul 27 2005 11:36AM  process_activity
```

The following is an example of a Historical Server view:

```
hs_create_view process_activity,
"Login Name", "Value for Sample",
"Process ID", "Value for Sample",
"Kernel Process ID", "Value for Sample",
"Connect Time", "Value for Session",
"Page I/O", "Value for Session",
"CPU Time", "Value for Session",
"Current Process State", "Value for Sample"
```

The query below shows the contents of the database table generated by the process_activity view illustrated above.

```
1> select * from process_activity
3> go

DbId CPUTime ProcessID KernelProcessID ConnectTime PageIO ProcessState LogName
-------------------------------------------------------------------------------
5    0.128447  17      3211313         61          28531    6          jsmith
5    0.128447  17      3211313         121         28531    6          jsmith
5    0.032169  19      3276850         60          32072    6          byoung
5    0.128447  17      3211313         182         28531    6          jsmith
5    0.099188  18      3342387         61          28759    6          dcharles
5    0.032169  19      3276850         121         33283    6          byoung
5    0.128447  17      3211313         242         28531    6          jsmith
5    0.105076  18      3342387         121         24373    6          dcharles
5    0.032169  19      3276850         181         33283    6          byoung
5    0.128447  17      3211313         303         28531    6          jsmith
```

Deleting Historical Server sessions

When you delete the recoding session using the command hs_delete_data, the default behavior is to only delete the output files corresponding to that session and to leave the entries intact in the output tables.

If you want to remove the session data from the output tables, you should enter:

```
hs_deletedata database_also, sessionId
```

This command removes all the corresponding entries in the Session and ViewPerSession tables.

Reporting errors

Historical Server reports errors in the monitoring data output saving process in the same manner as it does when outputting to flat files. You are notified of the errors encountered through the session error files and the HS log file. Errors relating to the database output mode can include:

- If the target Adaptive Server does not exist, Historical Server reports a failure.
- If the target database specified on the command-line does not exist, Historical Server reports a failure.
- If Historical Server cannot write to the database specified on the commandline because it has run out of disk space, Historical Server reports a failure.
- If Historical Server has to add columns to an existing table in order to use it for its monitoring output, it will log a warning message.

APPENDIX A

Data Items

Table of data items and definitions

Table A-1 describes the data items available for inclusion in recording session views. The table lists data items in alphabetical order and provides the following information about each one:

- A definition
- An Adaptive Server release dependency designation
- A result or key designation

Table A-1: Data items and definitions

Data item	Description
Application Execution Class Version: 11.5 and later	Configured execution class, if any, for a given application name. Because of override features, the configured execution class does not necessarily reflect the priority and engine group at runtime. The following notations are used in the monitor: • Blank – no execution class is configured for this application. • Execution class name – the execution class configured for the application in general, without consideration for specific logins. (That is, an execution class is configured for the application with null scope.) • Execution class name followed by asterisk (*) – in addition to the execution class configured for the application in general, additional execution classes are configured for specific logins. (That is, an execution class is configured for the application with null scope and at least one additional execution class is configured for the application with a specific scope.) • An asterisk (*) without execution class – no execution class is configured for the application in general, but execution classes are configured for specific logins using this application. (That is, at least one execution class is configured for the application with a specific scope.) Type: Result
Application Name Version: 11.0 and later	Name of application for which other statistics are being accumulated. Views that contain Application Name report only on processes that are active as of the end of the sample period. Application name is mutually exclusive with Process ID in a view. Type: Key

Data item	Description
Blocking Process ID Version: 11.0 and later	ID of the process that holds a lock that another process is waiting for, if any. A returned value of zero means that the other process is not blocked. Type: Result
Cache Efficiency Version: 11.0 and later	The number of cache hits per second per megabyte of a particular data cache. Type: Result
Cache Hit Pct Version: 11.0 and later	The fraction of the page reads for objects bound to a particular data cache that are satisfied from the cache computed from the following formula: *cache_hits* / (*cache_hits* + *cache_misses*) * 100 Type: Result
Cache Hits Version: 11.0 and later	The number of times a page read was satisfied from a particular data cache. Type: Result
Cache ID Version: 11.0 and later	The ID of a data cache in Adaptive Server version 11.0 or later. Particular database tables and indexes may be bound to a specific data cache, or all objects in a database may be bound to the same data cache. No object may be bound to more than one data cache. Type: Key
Cache Misses Version: 11.0 and later	Number of times a page read was satisfied from disk rather than from a particular data cache. This data item includes failed attempts to locate pages in the data caches during page allocation. Therefore, the number of physical page reads reported may be overstated. If this occurs, the percentage of data cache misses reported by Cache Hit Pct is understated. Type: Result
Cache Name Version: 11.0 and later	The name of a data cache. Particular database tables and indexes may be bound to a specific data cache, or all objects in a database may be bound to the same data cache. No object may be bound to more than one cache. Type: Key

Data item	Description
Cache Prefetch Efficiency Version: 11.0 and later	A percentage comparing the count of pages being reused in large I/O buffers (the denominator) to the number of those pages that were ever referenced by Adaptive Server. When a buffer is reused, all of the pages in it are counted as reused. Buffers are reused when there are no free buffers in the pool to accept a new physical read from a database device. The number of reused pages referenced by Adaptive Server divided by the result of the number of pages per buffer multiplied by the number of reused buffers indicates the effectiveness of large I/O fetches. Regardless of how many buffer pools are configured in a named data cache, Adaptive Server only uses two of them. It uses the 2K buffer pool and the pool configured with the largest-sized buffers. Prefetch effectiveness does not apply to the 2K buffer pool, since 2K grabs are not considered large I/O. Therefore, prefetch effectiveness applies to the largest buffer pool in the cache. For example, if a data cache has pools of buffers sized 2K, 8K, and 16K, the 8K pool is not used, and this metric reflects the effectiveness of large I/O in the 16K buffer pool. If the ratio is large, then prefetching is effective; otherwise, prefetching is not providing much benefit. This may suggest that a buffer pool should be eliminated (or it may imply that a clustered index on some table is fragmented, and that the index should be dropped and re-created). Type: Result
Cache Refer and Reuse Version: 11.0 and later	The number of pages in buffers that were both referenced and reused. This count is employed when determining the efficiency of prefetching buffers (see Cache Prefetch Efficiency). This data item, unlike Cache Prefetch Efficiency, includes activity in the default 2K buffer pool. See Cache Prefetch Efficiency for a definition of buffer reuse. Type: Result
Cache Reuse Version: 11.0 and later	The number of pages in buffers that were reused. A large value indicates a high rate of turnover (of buffers in this memory pool), and suggests that the pool may be too small. A zero value suggests that this memory pool may be too large. This data item, unlike Cache Prefetch Efficiency, includes activity in the default 2K buffer pool. See Cache Prefetch Efficiency for a definition of buffer reuse. Type: Result
Cache Reuse Dirty Version: 11.0 and later	The number of times that a buffer that was reused had changes that needed to be written. A non-zero value indicates that the wash size is too small. See Cache Prefetch Efficiency for a definition of buffer reuse. Type: Result
Cache Size Version: 11.0 and later	The size of a data cache, in megabytes. Type: Result

Data item	Description
Cache Spinlock Contention Version: 11.0 and later	The fraction of the requests for a data cache's spinlock that were forced to wait. *spinlock_waits* / *spinlock_requests* Type: Result
Code Memory Size Version: 11.0 and later	Amount of memory in bytes allocated to Adaptive Server. Type: Result
Connect Time Version: 11.0 and later	Number of seconds elapsed since the process was started or since the session was started, whichever is smaller. Type: Result
CPU Busy Percent Version: 11.0 and later	Percentage of total server CPU ticks that the Adaptive Server CPU was busy. Type: Result
CPU Percent Version: 11.0 and later	If used in a view with Process ID, this is the percentage of time a single process was in the Running state over the time all processes were in the Running state. If used in a view with Application Name, this is the percentage of time the set of processes running a given application were in the Running state over the time all processes were in the Running state. Type: Result
CPU Time Version: 11.0 and later	If used in a view with no keys, this data item is the total CPU busy time, in seconds, on the server. If used with keys, this data item is the busy time, in seconds, that was used by each process, application, or engine. Process ID and Application Name are mutually exclusive. Type: Result
CPU Yields Version: 11.0 or later	Number of times that the Adaptive Server yielded to the operating system. Type: Result
Current Application Name Version: 11.0 and later	The name of the application that is currently executing on a particular process. Type: Result
Current Engine Version: 11.0 and later	Number of the Adaptive Server engine on which the process was running most recently. Type: Result
Current Execution Class Version: 11.5 and later	The name of the execution class under which a process is currently running. Type: Result
Current Process State Version: 11.0 and later	Current state of a process. See Process State for definitions of the possible states. Type: Result

Data item	Description
Current Stmt Batch ID Version: 11.5 and later	The ID of a particular query batch being executed on a particular process. Type: Result
Current Stmt Batch Text Version: 11.5 and later	The text of a particular query batch being executed for a particular process. This text may be only an initial substring of the complete text in a query batch. The amount of text stored in this field is determined by the Adaptive Server max SQL text monitored configuration parameter. Type: Result
Current Stmt Batch Text Byte Offset Version: 11.5 and later	The byte offset to the beginning of a statement within the query batch or stored procedure being executed for a particular process. If both: Current Stmt Procedure Database ID is equal to 0 and Current Stmt Procedure ID is equal to 0 then the statement is the currently executing SQL statement in the query batch. Otherwise, the statement is the currently executing SQL statement in the stored procedure uniquely identified by these two IDs. Type: Result
Current Stmt Batch Text Enabled Version: 11.5 and later	Reports whether Adaptive Server (version 11.5 and later) is saving the SQL text of the currently executing query batches and, if so, how much. Value of 0 = saving SQL text disabled Value of 1 or more = maximum number of bytes of batch text per server process that can be saved. Type: Result
Current Stmt Context ID Version: 11.5 and later	The ID that uniquely identifies a stored procedure invocation within a particular query batch being executed for a particular process. Type: Result
Current Stmt CPU Time Version: 11.5 and later	The amount of time (in seconds) that the currently executing SQL statement has spent in the running state. Type: Result
Current Stmt Elapsed Time Version: 11.5 and later	The amount of time (in seconds) that the currently executing SQL statement has been running. Type: Result
Current Stmt Line Number Version: 11.5 and later	The number of the line (within a query batch or stored procedure) that contains the beginning of the currently executing SQL statement for a particular process. The currently executing SQL statement is in the query batch if both Current Stmt Procedure Database ID is equal to 0 and Current Stmt Procedure ID is equal to 0. Otherwise, the currently executing SQL statement is in the stored procedure uniquely identified by these two IDs. Type: Result

Data item	Description
Current Stmt Locks Granted After Wait Version: 11.5 and later	Number of lock requests by the currently executing SQL statement that were granted after waiting. Type: Result
Current Stmt Locks Granted Immediately Version: 11.5 and later	Number of lock requests by the currently executing SQL statement that were granted immediately or were not needed (because sufficient locking was already held by the requestor). Type: Result
Current Stmt Locks Not Granted Version: 11.5 and later	Number of lock requests by the currently executing SQL statement that were denied. Type: Result
Current Stmt Logical Reads Version: 11.5 and later	Number of data page reads satisfied from cache or from device reads by the currently executing SQL statement. Type: Result
Current Stmt Number Version: 11.5 and later	The number of the statement (appearing in a query batch or stored procedure) that is the currently executing SQL statement for a particular process. The currently executing SQL statement is in the query batch if both Current Stmt Procedure Database ID is equal to 0 and Current Stmt Procedure ID is equal 0. Otherwise, the currently executing SQL statement is in the stored procedure uniquely identified by these two IDs. A value of 0 indicates partial result data for the currently executing SQL statement. In other words, this SQL statement began executing before monitoring began. Performance metrics are available but numbers reflect only the time period since the start of monitoring. Type: Result
Current Stmt Page I/O Version: 11.5 and later	Number of combined logical page reads and page writes by the currently executing SQL statement. Type: Result
Current Stmt Physical Reads Version: 11.5 and later	Number of data page reads by the currently executing SQL statement that could not be satisfied from the data cache. Type: Result
Current Stmt Procedure Database ID Version: 11.5 and later	The database ID of the stored procedure (including triggers, a special kind of stored procedure) that contains the currently executing SQL statement for a particular process. If the currently executing SQL statement is not contained in a stored procedure, this ID is 0. Type: Result

Data item	Description
Current Stmt Procedure Database Name Version: 11.5 and later	The database name of the stored procedure (including triggers, a special kind of stored procedure) that contains the currently executing SQL statement for a particular process. If the currently executing SQL statement is not contained in a stored procedure, this name is "**NoDatabase**". Type: Result
Current Stmt Procedure ID Version: 11.5 and later	The ID of the stored procedure (including triggers, a special kind of stored procedure) that contains the currently executing SQL statement for a particular process. If the currently executing SQL statement is not contained in a stored procedure, this ID is 0. Type: Result
Current Stmt Procedure Name Version: 11.5 and later	The name of the stored procedure (including triggers, a special kind of stored procedure) that contains the currently executing SQL statement for a particular process. If the currently executing SQL statement is not contained in a stored procedure, this name is "**NoObject**". Type: Result
Current Stmt Procedure Owner Name Version: 11.5 and later	The owner name of the stored procedure (including triggers, a special kind of stored procedure) that contains the currently executing SQL statement for a particular process. If the currently executing SQL statement is not contained in a stored procedure, this name is "**NoOwner**". Type: Result
Current Stmt Procedure Text Version: 11.5 and later	The text of a particular stored procedure (including triggers, a special kind of stored procedure) being executed for a particular process. If both Current Stmt Procedure Database ID is equal to 0 and Current Stmt Procedure ID is equal to 0 then a stored procedure is not currently executing and this text is a null-terminated empty string (""). If the text is not available (because this stored procedure was compiled and its text was discarded, or because the text is stored in an encrypted format), then this text is a null-terminated empty string (""). Type: Result
Current Stmt Query Plan Text Version: 11.5 and later	The text of the query plan for a particular query being executed for a particular process. If the text is not available (because the Adaptive Server has removed this plan from its catalog of query plans), then this text is a null-terminated empty string (""). Type: Result
Current Stmt Start Time Version: 11.5 and later	The date and time, in the time zone of Adaptive Server, when the currently executing SQL statement began running. If this SQL statement began running before monitoring began, then this result is the date and time that activity was first encountered for this statement. Type: Result

Data item	Description
Current Stmt Text Byte Offset Version: 11.5 and later	The byte offset to the beginning of a statement within the query batch or stored procedure being executed for a particular process. If both Current Stmt Procedure Database ID is equal to 0 and Current Stmt Procedure ID is equal to 0, then the statement is the currently executing SQL statement in the query batch. Otherwise, the statement is the currently executing SQL statement in the stored procedure uniquely identified by those two IDs. Type: Result
Database ID Version: 11.0 and later	Unique identification of a database. Type: Key
Database Name Version: 11.0 and later	Name of the database. Type: Result
Deadlock Count Release: 11.0 and later	Number of deadlocks. Type: Result
Demand Lock Version: 11.0 and later	A character string (Y or N) that indicates whether or not a lock that has been upgraded to demand lock status. Type: Result
Device Hit Percent Version: 11.0 and later	The fraction of device requests is computed by multiplying the quotient of device hits divided by device misses plus device misses by 100. Type: Result
Device Hits Version: 11.0 and later	Number of times access to a device was granted. Type: Result
Device I/O Version: 11.0 and later	Combination of device reads and device writes. Type: Result
Device Misses Version: 11.0 and later	Number of times access to a device had to wait. Type: Result
Device Name Version: 11.0 and later	Name of a database device defined in Adaptive Server. Type: Key
Device Reads Version: 11.0 and later	Number of reads made from a device. Type: Result
Device Writes Version: 11.0 and later	Number of writes made to a device. Type: Result

Data item	Description
Elapsed Time Version: 11.0 and later	The time increment, in seconds, either from one data refresh to the next (sample) or from the creation of the view to the present (session). Type: Result
Engine Number Version: 11.0 and later	Number of the Adaptive Server engine. Type: Key
Host Name Version: 11.0 and later	The name of the host computer that established a particular connection. Type: Result
Index Logical Reads Release: 11.0 and later	Number of index page reads satisfied from cache and from database devices. Type: Result
Index Physical Reads Version: 11.0 and later	Number of index page reads that could not be satisfied from the data cache. Type: Result
Kernel Process ID Version: 11.0 and later	An Adaptive Server process identifier that remains unique over long periods of time. Type: Key
Kernel Structures Memory Size Version: 11.0 and later	Amount of memory in bytes allocated to the kernel structures. Type: Result
Large I/O Denied Version: 11.0 and later	The number of times the buffer manager did not satisfy requests (of the optimizer) to load data into a buffer in this data cache by fetching more than one contiguous page from disk at a time. Type: Result
Large I/O Performed Version: 11.0 and later	The number of times the buffer manager satisfied requests (of the optimizer) to load data into a buffer in this data cache by fetching more than one contiguous page from disk at a time. Type: Result
Large I/O Requested Version: 11.0 and later	The number of times the optimizer made requests (of the buffer manager) to load data into a buffer in this data cache by fetching more than one contiguous page from disk at a time. Type: Result
Lock Count Version: 11.0 and later	Number of locks. Type: Result
Lock Hit Percent Version: 11.0 and later	Percentage of successful requests for locks. Type: Result

Data item	Description
Lock Result Version: 11.0 and later	Result of a logical lock request. Lock result values are: • 1 – granted immediately. • 2 – not needed; requestor already held a sufficient lock. • 3 – waited; requestor waited. • 4 – did not wait; lock was not available immediately and the requestor did not want the lock request to be queued. • 5 – deadlock; requestor selected as deadlock victim. • 6 – interrupted; the lock request was interrupted by attention condition. Type: Key
Lock Results Summarized Version: 11.0 and later	Lock results summarized at a granted or not granted level. • 1 – the lock result summary granted is composed of the lock results: granted, not needed, and waited. • 2 – the lock result summary not granted is composed of the lock results: did not wait, deadlock, and interrupted. Type: Key
Lock Status Version: 11.0 and later	Current status of a lock which includes lock status values: • 1– held and blocking. • 2 – held and not blocking. • 3 – requested and blocked. • 4 – requested and not blocked. Type: Key
Lock Status Count Version: 11.0 and later	Number of locks in each lock status. This is a snapshot value. Type: Result

Data item	Description
Lock Type Version: 11.0 and later	Adaptive Server protects tables or data pages currently used by active transactions by locking them. Adaptive Server employs the following lock types: • 1 – exclusive table lock. • 2 – shared tablelock. • 3 – exclusive intent lock. • 4 – shared intent lock. • 5 – exclusive page lock. • 6 – shared page lock. • 7 – update page lock. • 8 – exclusive row lock. • 9 – shared row lock. • 10 – update row lock. Type: Key
Locks Being Blocked Count Version: 11.0 and later	Number of locks being blocked by this process that holds this “held_and_blocking” lock. Type: Result
Locks Granted Immediately Version: 11.5 and later	Number of locks that were granted immediately, without having to wait for another lock to be released. Type: Result
Locks Granted After Wait Version: 11.5 and later	Number of locks that were granted after waiting for another lock to be released. Type: Result
Locks Not Granted Version: 11.5 and later	Number of locks that were requested but not granted. Type: Result
Log Contention Percent Version: 11.0 and later	The percentage of times, of the total times when a user log cache was flushed into the transaction log, that it had to wait for the log semphore. A high percentage may indicate that the user log cache size should be increased. Type: Result
Logical Page Reads Version: 11.0 and later	Number of data page reads per unit of time, whether satisfied from cache or from a database device. Type: Result
Login Name Version: 11.0 and later	Login name associated with Adaptive Server processes. Type: Result

Data item	Description
Most Active Device I/O Version: 11.0 and later	Number of combined reads and writes against the device with the most activity during a given time interval. Type: Result
Most Active Device Name Version: 11.0 and later	Name of the device with the largest number of combined reads and writes during a given time interval. Type: Result
Net Bytes Received Version: 11.0 and later	Number of network bytes received. Type: Result
Net Bytes Sent Version: 11.0 and later	Number of network bytes sent. Type: Result
Net Default Packet Size Version: 11.0 and later	Default network packet size. Type: Result
Net I/O Bytes Version: 11.0 and later	Total number of network bytes sent and received. Type: Result
Net Max Packet Size Version: 11.0 and later	Configured maximum size for a network packet. Type: Result
Net Packet Size Received Version: 11.0 and later	Average size of network packets received. Type: Result
Net Packet Size Sent Version: 11.0 and later	Average size of network packets sent. Type: Result
Net Packets Received Version: 11.0 and later	Number of network packets received. Type: Result
Net Packets Sent Version: 11.0 and later	Number of network packets sent. Type: Result
Number of Engines Version: 11.0 and later	Number of engines configured for Adaptive Server. Type: Result
Number of Processes Version: 11.0 and later	Number of processes currently running on Adaptive Server, or, if used with the key Application Name, the number of processes currently running a given application. Type: Result

Data item	Description
Object ID Version: 11.0 and later	ID of a database object. The object returned is a database table, a stored procedure, or a temporary table. Object IDs might be negative numbers. The object IDs that Adaptive Server assigns to temporary tables can be positive or negative. Type: Key
Object Name Version: 11.0 and later	Database object name. The string **TempObject** is reported for temporary tables. Type: Result
Object Type Version: 11.0 and later	Type of database object: • 0 – none. • 1 – stored procedure (including triggers). • 2 – table. Type: Result
Owner Name Version: 11.0 and later	Name of an objects's owner. Type: Result
Page Cache Size Version: 11.0 and later	Amount of memory in bytes allocated for the page cache. Type: Result
Page Hit Percent Version: 11.0 and later	Percentage of times that a data page read could be satisfied from cache without requiring a physical page read. Type: Result
Page I/O Version: 11.0 and later	Combined total of logical page reads and page writes. Type: Result
Page Number Version: 11.0 and later	Data page number for a particular lock or lock request. Type: Key
Page Writes Version: 11.0 and later	Number of pages written to a database device. Type: Result
Physical Page Reads Version: 11.0 and later	Number of data page reads that could not be satisfied from the data cache. Type: Result
Procedure Buffer Size Version: 11.0 and later	Amount of memory in bytes allocated for the procedure buffer. Type: Result
Procedure CPU Time Version: 11.0 and later	Number of seconds of CPU time spent executing a stored procedure. Type: Result

Data item	Description
Procedure Database ID Version: 11.0 and later	Database ID of the active stored procedure. Type: Key
Procedure Database Name Version: 11.0 and later	Database name of the active stored procedure. Type: Key
Procedure Elapsed Time Version: 11.0 and later	Number of seconds elapsed during a stored procedure execution. All statistic types valid with this data item report time in units of seconds. For example, "Procedure Elapsed Time", "Average for Session" reports the average number of elapsed seconds per procedure execution. Type: Result
Procedure Execution Class Version: 11.5 and later	Configured execution class, if any, for a given stored procedure. Type: Result
Procedure Execution Count Version: 11.0 and later	Number of times a stored procedure, or a line in a stored procedure, was executed. Type: Result
Procedure Header Size Version: 11.0 and later	Amount of memory in bytes allocated for the procedure header. Type: Result
Procedure Hit Percent Version: 11.0 and later	Percentage of times that a procedure execution found the procedure's query plan in procedure cache and available for use. Type: Result
Procedure ID Version: 11.0 and later	Active stored procedure. Active indicates the top-level stored procedure was called. Type: Key
Procedure Line Number Version: 11.0 and later	Stored procedure line number. Type: Key
Procedure Line Text Version: 11.0 and later	Entire text of the stored procedure. Type: Result
Procedure Logical Reads Version: 11.0 and later	Number of requests to execute a stored procedure, whether satisfied from procedure cache or with a read from *sysprocedures*. Type: Result
Procedure Name Version: 11.0 and later	Name of the active stored procedure. Type: Result.

Data item	Description
Procedure Owner Name Version: 11.0 and later	Name of the owner of the active stored procedure. Type: Result
Procedure Physical Reads Version: 11.0 and later	Number of requests to execute a stored procedure, for which a read from *sysprocedures* was necessary. Type: Result
Procedure Statement Number Version: 11.0 and later	Statement number within a stored procedure. A single stored procedure line can contain one or more statements. Type: Key
Process ID Version: 11.0 and later	Adaptive Server process identification number. Views that contain Process ID only report on processes that are active as of the end of the sample period. Process ID is mutually exclusive with Application Name in a view. Type: Key
Process State Version: 11.0 and later	Process state: • 0 – None. • 1 – alarm sleep.Waiting on an alarm. • 2 – background. Adaptive Server process executing. • 3 – bad status. Undetermined error condition. • 4 – infected. Tagged by Adaptive Server as unprocessable. • 5 – lock sleep. Waiting on a lock acquisition. • 6 – received sleep. Waiting on a network read. • 7 – runnable. Waiting to run according to priority and availability of CPU. • 8 – running. Executing. • 9 – send sleep. Waiting on a network send. • 10 – sleeping. Paused for any other reason not listed here, such as: waiting on device I/O (physical reads) or waiting for client activity. • 11 – stopped. Process terminated. • 12 – terminating. Process terminating. • 13 – unknown. Process state undeterminable. • 14 – remote I/O. Waiting on a remote (OMNI) server to complete an operation. • 15 – synch sleep. Waiting to synchronize with some other server process(es) that are working in parallel to execute a given query. Type: Key
Process State Count Version: 11.0 and later	Number of processes in a particular state. Type: Result

Data item	Description
Rows Deleted Version: 11.0 and later	Number of rows deleted from a database table. Type: Result
Rows Deleted Deferred Version: 11.0 and later	Number of rows deleted from a database table in deferred mode. Type: Result
Rows Deleted Direct Version: 11.0 and later	Number of rows deleted from a database table in direct mode. Type: Result
Rows Inserted Version: 11.0 and later	Insertions into a database table. Type: Result
Rows Inserted Clustered Version: 11.0 and later	Insertions into database tables with clustered indexes. Type: Result
Rows Inserted Heap Version: 11.0 and later	Insertions into database tables without clustered indexes. Type: Result
Rows Updated Version: 11.0 and later	Updates to a database table. Type: Result
Rows Updated Deferred Version: 11.0 and later	Updates that require two steps to complete. First, records for deleting the existing entry and inserting the new entry are written to the log, but only the deletes are actually performed on the data pages. In the second step, the log is rescanned and the insert operations performed on the data pages. Type: Result
Rows Updated Direct Version: 11.0 and later	The sum of expensive, in-place, and not-in-place updates (everything except updates deferred). Type: Result
Rows Updated Expensive Version: 11.0 and later	A type of direct update that deletes a row from its original location and inserts it in a new location.
Rows Updated In Place Version: 11.0 and later	A type of direct update that does not require rows to move on a data page. Type: Result
Rows Updated Not In Place Version: 11.0 and later	A type of direct update that does not require the updated row to move, but because the length of the updated row changes, other rows on the data page are moved. Also known as cheap updates. Type: Result

Data item	Description
Select Statements Version: 11.0 and later	Number of SELECT or OPEN CURSOR statements. Type: Result
Server Structures Size Version: 11.0 and later	Amount of memory in bytes allocated for the Adaptive Server structures. Type: Result
SQL Server Name Version: 11.0 and later	Name of the Adaptive Server that is being monitored as specified by the -S parameter in the start-up command for the Monitor Server being used. Type: Result
SQL Server Version Version: 11.0 and later	Version of the Adaptive Server that is being monitored. For more information, see the global *@@version* variable in the *Transact-SQL User's Guide*. Type: Result
Thread Exceeded Max Version: 11.5 and later	The number of times a query plan was adjusted at runtime because of attempting to exceed the configured limit of threads in the server-wide worker thread pool in Adaptive Server version 11.5 and later. Type: Result
Thread Exceeded Max Percent Version: 11.5 and later	The percentage of time a query plan was adjusted at runtime because of attempting to exceed the configured limit of threads in the server-wide worker thread pool in Adaptive Server version 11.5 and later. Type: Result
Thread Max Used Version: 11.5 and later	The maximum number of threads from the server-wide worker thread pool that were concurrently in use on the server. Type: Result
Time Waited on Lock Version: 11.0 and later	Amount of time (in seconds) that a lock request has been waiting. Type: Result
Timestamp Version: 11.0 and later	In recording session views and in playback views when *summarization_level* is raw, the date and time on the Adaptive Server when the recording session data was gathered. In playback views, when *summarization_level* is actual, entire, or a user-defined interval, the time is converted to the time zone of Historical Server. For more information, see the getdate() function in the *Transact-SQL User's Guide*. Type: Result
Timestamp Datim Version: 11.0 and later	In recording session views and in playback views when *summarization_level* is raw, the date and time on the Adaptive Server when the recording session data was gathered, returned in a CS_DATETIME structure. For more information, see the getdate() function in the *Transact-SQL User's Guide*. In playback views, when *summarization_level* is actual, entire, or a user-defined interval, the time is converted to the time zone on Historical Server. Type: Result

Data item	Description
Transactions Version: 11.0 and later	Total number of committed Transact-SQL statement blocks delimited by a begin transaction and commit transaction statement. Type: Result

APPENDIX B

Specifications for Defining Recording Session Views

This appendix describes rules and considerations for designing recording session views.

Definition of key and result

Data items are either keys or results.

- A key data item uniquely identifies the rows returned in a view. For example, in a view that defines per-process data, the Process ID data item identifies each process.
- A result data item identifies the information to be returned concerning each key value.

By using a mix of keys and results to define recording session views, you control the data that Historical Server returns.

You can combine key data items within a view to narrow the scope of the returned data. With the inclusion of each successive key, envision adding the word “per” to a view definition. For example, page I/Os “per” database table “per” process. (Only certain key combinations are valid for each data item. See Table B-2 on page 128.)

To return server-wide data, define a view with no keys. In a view with no keys, the result always contains exactly one row. (Only certain data items are valid at the server level. See Table B-2 on page 128.)

Table B-1 provides examples of views using different key combinations.

Table B-1: View examples using various key combinations

View definition	Results

Page I/O	Since there are no keys in the view, the result is page I/O for the whole server. For example:
	`Page I/O`
	`--------`
	`145`

Process ID (key), Login Name, Page I/O	The result is page I/O per process. For example:
	`Process ID  Login Name  Page I/O`
	`----------  ----------  ----------`
	`1  sa  45`
	`5  joe  100`

Process ID (key), Database ID (key), Object ID (key), Database Name, Object Name, Page I/O	The result is page I/O per database table per process. For example:
	`Process ID  Database ID  Object ID  Databse Name  Object Name  Page I/O`
	`----------  ----------  ----------  ----------  ----------  ----------`
	`1  5  208003772  pubs2  authors  10`
	`1  5  336004228  pubs2  titles  35`
	`5  5  208003772  pubs2  authors  100`

Designing recording session views

To define a view:

- Choose valid combinations of key and result data items using Table B-2 on page 128.
- Choose valid statistic types for each of the data items using Table B-3 on page 152.

Using the Process ID

When a server process terminates, Adaptive Server can reuse its Process ID for a new process. Therefore, the Process ID data item is not guaranteed to uniquely identify a process. The Kernel Process ID data item, however, uniquely identifies a process.

To create a non-raw playback view that shows per-process data, the recording session view and the playback view must include both the Process ID and Kernel Process ID data items. If playback is raw, using only Process ID is allowed.

Views that include Process ID return rows as follows:

- Recording session views and raw playback views.

  These views return a row only for Process IDs that exist at the end of a sample interval. If a server process terminates in the middle of a sample interval, a row is not returned for its Process ID.

- Playback views when the session's *summarization_level* parameter is actual, entire, or user-defined intervals.

These views return rows for all server processes included in any of the input samples. However, since process IDs are not guaranteed to be unique, the Kernel Process ID data item must also be included to ensure uniqueness of the key. Otherwise, the view could erroneously summarize two different processes.

Using the application name

Historical Server accumulates performance data per application by summing the data for all processes having the same application name. It collects performance data only for processes that exist as of the end of the sample interval. See "Using the Process ID" on page 126 for more information.

Empty rows versus no rows in views

When there is no activity to report, some data items cause an empty row (that is, a row with zero values for result data items) to appear in a view, and other data items cause the row to be omitted. The rules controlling whether empty rows appear in a view are:

- Server-level data items always return a row, even when there is no activity to report.
- Views that contain the key data item Process ID or Application Name only report on processes that are active as of the end of the sample period.
- Views that contain the key data items Object ID or Stored Procedure ID omit the row when there is no activity to report during the sample period.

- Views that contain keys other than those listed in the previous bullets return rows when there is no activity.

Table of valid key and result data item combinations

There are restrictions on the data items that can be combined within a view. The restrictions are based upon the relationships between data items. For example, it makes no sense to ask for "CPU Busy Percent" per "Page Number"; therefore, it is not allowed.

Table B-2 shows valid combinations of result and key data items for views. The table lists result data items in alphabetical order. The basic rules for determining whether a result data item can be included in a recording session view are:

- The view must include *all* of the required keys listed for the result data item.
- The view *can* include the optional keys listed for the result data item. However, not all of the optional keys listed for a data item are guaranteed to work together.
- The view *cannot* include any other keys besides those listed as required or optional. (A few exceptions are described in the Notes column.)
- Data Available at Server Level? – indicates whether this data item can be used in a view without any keys to obtain summary or static information about Adaptive Server.
- A result data item that has no required keys can be used in server-level views. Server-level views have no keys and report summary or static information about Adaptive Server.
- Footnotes contain additional information about using the result data items in views.

Table B-2: Valid key and result combinations

Data item	Required and optional keys
Application Execution Class [2]	*Valid at server level?* No *Required keys:* Application Name
Application Name *(KEY)*[3]	

Data item	Required and optional keys
Blocking Process ID	*Valid at server level?* No *Required keys:* • Process ID • Database ID • Object ID • Lock Status • Page Number *Optional keys:* Lock Type
Cache Efficiency	*Valid at server level?* No *Required keys:* Cache ID
Cache Hit Pct	*Valid at server level?* No *Required keys:* Cache ID
Cache Hits	*Valid at server level?* No *Required keys:* Cache ID
Cache ID *(KEY)* [4]	
Cache Misses	*Valid at server level?* No *Required keys:* Cache ID
Cache Name[4]	*Valid at server level?* No *Required keys:* Cache ID
Cache Prefetch Efficiency	*Valid at server level?* No *Required keys:* Cache ID

Data item	Required and optional keys
Cache Refer and Reuse	*Valid at server level?* No *Required keys:* Cache ID
Cache Reuse	*Valid at server level?* No *Required keys:* Cache ID
Cache Reuse Dirty	*Valid at server level?* Yes *Required keys:* None *Optional keys:* Cache ID
Cache Size	*Valid at server level?* No *Required keys:* Cache ID
Cache Spinlock Contention	*Valid at server level?* No *Required keys:* Cache ID
Code Memory Size	*Valid at server level?* Yes *Required keys:* None
Connect Time	*Valid at server level?* No *Required keys:* Process ID

Data item	Required and optional keys
CPU Busy Percent	*Valid at server level?* Yes *Required keys:* None *Optional keys:* Engine Number
CPU Percent[3]	*Valid at server level?* No *Required keys:* Process ID *or* Application Name *Optional keys:* Engine Number
CPU Time[3]	*Valid at server level?* Yes *Required keys:* None *Optional keys:* • Process ID *or* • Application Name • Engine Number
CPU Yields	*Valid at server level?* Yes *Required keys:* None *Optional keys:* Engine Number
Current Application Name	*Valid at server level?* No *Required keys:* Process ID
Current Engine	*Valid at server level?* No *Required keys:* Process ID

Data item	Required and optional keys
Current Execution Class[2]	*Valid at server level?* No *Required keys:* Process ID
Current Process State	*Valid at server level?* No *Required keys:* Process ID
Current Stmt Batch ID[2]	*Valid at server level?* No *Required keys:* Process ID
Current Stmt Batch Text[2]	*Valid at server level?* No *Required keys:* Process ID
Current Stmt Batch Text Byte Offset[2]	*Valid at server level?* No *Required keys:* Process ID
Current Stmt Batch Text Enabled[2]	*Valid at server level?* Yes *Required keys:* None
Current Stmt Context ID[2]	*Valid at server level?* No *Required keys:* Process ID
Current Stmt CPU Time[2]	*Valid at server level?* No *Required keys:* Process ID
Current Stmt Elapsed Time[2]	*Valid at server level?* No *Required keys:* Process ID

Data item	Required and optional keys
Current Stmt Line Number[2]	*Valid at server level?* No *Required keys:* Process ID
Current Stmt Locks Granted After Wait[2]	*Valid at server level?* No *Required keys:* Process ID
Current Stmt Locks Granted Immediately[2]	*Valid at server level?* No *Required keys:* Process ID
Current Stmt Locks Not Granted[2]	*Valid at server level?* No *Required keys:* Process ID
Current Stmt Logical Reads	*Valid at server level?* No *Required keys:* Process ID
Current Stmt Number [2]	*Valid at server level?* No *Required keys:* Process ID
Current Stmt Page I/O[2]	*Valid at server level?* No *Required keys:* Process ID
Current Stmt Page Writes[2]	*Valid at server level?* No *Required keys:* Process ID
Current Stmt Physical Reads[2]	*Valid at server level?* No *Required keys:* Process ID

Data item	Required and optional keys
Current Stmt Procedure Database ID[2]	*Valid at server level?* No *Required keys:* Process ID
Current Stmt Procedure Database Name[2]	*Valid at server level?* No *Required keys:* Process ID
Current Stmt Procedure ID[2]	*Valid at server level?* No *Required keys:* Process ID
Current Stmt Procedure Name[2]	*Valid at server level?* No *Required keys:* Process ID
Current Stmt Procedure Owner Name[2]	*Valid at server level?* No *Required keys:* Process ID
Current Stmt Procedure Text[2]	*Valid at server level?* No *Required keys:* Process ID
Current Stmt Query Plan Text[2]	*Valid at server level?* No *Required keys:* Process ID
Current Stmt Start Time[2]	*Valid at server level?* No *Required keys:* Process ID
Current Stmt Text Byte Offset[2]	*Valid at server level?* No *Required keys:* Process ID

Data item	Required and optional keys
Database ID *(KEY)*	
Database Name[1]	*Valid at server level?* No *Required keys:* Database ID
Deadlock Count	*Valid at server level?* Yes *Required keys:* None
Demand Lock	*Valid at server level?* No *Required keys:* • Process ID • Database ID • Object ID • Lock Status • Page Number *Optional keys:* Lock Type
Device Hit Percent	*Valid at server level?* Yes *Required keys:* None *Optional keys:* Device Name
Device Hits	*Valid at server level?* Yes *Required keys:* None *Optional keys:* Device Name

Data item	Required and optional keys
Device I/O	*Valid at server level?* Yes *Required keys:* None *Optional keys:* Device Name
Device Misses	*Valid at server level?* Yes *Required keys:* None *Optional keys:* Device Name
Device Name *(KEY)*	
Device Reads	*Valid at server level?* Yes *Required keys:* None *Optional keys:* Device Name
Device Writes	*Valid at server level?* Yes *Required keys:* None *Optional keys:* Device Name
Elapsed Time	*Valid at server level?* Yes *Required keys:* None
Engine Number *(KEY)*	
Host Name	*Valid at server level?* No *Required keys:* Process ID

Data item	Required and optional keys

Index Logical Reads	*Valid at server level?*
	Yes
	Required keys:
	None
	Optional keys:
	• Process ID
	• Database ID
	• Object ID
	• Engine Number
	• Key combination:
	  • Procedure Database ID
	  • Procedure ID

Index Physical Reads	*Valid at server level?*
	Yes
	Required keys:
	None
	Optional keys:
	• Process ID
	• Database ID
	• Object ID
	• Engine Number
	• Key combination:
	  • Procedure Database ID
	  • Procedure ID

Kernel Process ID[1]	*Valid at server level?* No *Required keys:* Process ID
Kernel Structures Memory Size	*Valid at server level?* Yes *Required keys:* None

Data item	Required and optional keys
Large I/O Denied	*Valid at server level?* No *Required keys:* Cache ID
Large I/O Performed	*Valid at server level?* No *Required keys:* Cache ID
Large I/O Requested	*Valid at server level?* No *Required keys:* Cache ID
Lock Count	*Valid at server level?* Yes *Required keys:* None *Optional keys:* • Process ID • Lock Type • Lock Result • Lock Results Summarized
Lock Hit Percent	*Valid at server level?* Yes *Required keys:* None
Lock Result *(KEY)*	
Lock Results Summarized *(KEY)*	
Lock Status *(KEY)*	
Lock Status Count	*Valid at server level?* No *Required keys:* Lock Status
Lock Type *(KEY)*	

Data item	Required and optional keys
Locks Being Blocked Count	*Valid at server level?* No *Required keys:* • Process ID • Database ID • Object ID • Lock Status • Lock Type • Page Number

Locks Granted After Wait[3]	*Valid at server level?*
	Yes
	Required keys:
	None
	Optional keys:
	• Process ID *or*
	• Application Name
	• Key Combination:
	  • Database ID
	  • Object ID
	• Key Combination:
	  • Procedure Database ID
	  • Procedure ID

Locks Granted Immediately[3]	*Valid at server level?*
	Yes
	Required keys:
	None
	Optional keys:
	• Process ID *or*
	• Application Name
	• Key Combination:
	  • Database ID
	  • Object ID
	• Key Combination:
	  • Procedure Database ID
	  • Procedure ID

Data item	Required and optional keys

Locks Not Granted[3]	*Valid at server level?*
	Yes
	Required keys:
	None
	Optional keys:
	• Process ID *or*
	• Application Name
	• Key Combination:
	  • Database ID
	  • Object ID
	• Key Combination:
	  • Procedure Database ID
	  • Procedure ID

Log Contention Percent	*Valid at server level?* Yes *Required keys:* None

Logical Page Reads	*Valid at server level?*
	Yes
	Required keys:
	None
	Optional keys:
	• Process ID
	• Engine Number
	• Key combination:
	  • Database ID
	  • Object ID
	• Key combination:
	  • Procedure Database ID
	  • Procedure ID

Login Name[1]	*Valid at server level?* No *Required keys:* Process ID

Data item	Required and optional keys
Most Active Device I/O	*Valid at server level?* Yes *Required keys:* None
Most Active Device Name	*Valid at server level?* Yes *Required keys:* None
Net Bytes Received	*Valid at server level?* Yes *Required keys:* None
Net Bytes Sent	*Valid at server level?* Yes *Required keys:* None
Net Default Packet Size	*Valid at server level?* Yes *Required keys:* None
Net I/O Bytes	*Valid at server level?* Yes *Required keys:* None
Net Max Packet Size	*Valid at server level?* Yes *Required keys:* None
Net Packet Size Received	*Valid at server level?* Yes *Required keys:* None
Net Packet Size Sent	*Valid at server level?* Yes *Required keys:* None

Data item	Required and optional keys
Net Packets Received	*Valid at server level?* Yes *Required keys:* None
Net Packets Sent	*Valid at server level?* Yes *Required keys:* None
Number of Engines	*Valid at server level?* Yes *Required keys:* None
Number of Processes[2]	*Valid at server level?* Yes *Required keys:* None *Optional keys:* Application Name
Object ID *(KEY)*	
Object Name[1]	*Valid at server level?* No *Required keys:* • Database ID • Object ID
Object Type[1]	*Valid at server level?* No *Required keys:* • Database ID • Object ID
Owner Name[1]	*Valid at server level?* No *Required keys:* • Database ID • Object ID

Data item	Required and optional keys
Page Cache Size	*Valid at server level?* Yes *Required keys:* None

Page Hit Percent	*Valid at server level?*
	Yes
	Required keys:
	None
	Optional keys:
	• Process ID
	• Engine Number
	• Key combination:
	  • Database ID
	  • Object ID
	• Key combination:
	  • Procedure Database ID
	  • Procedure ID

Page I/O	*Valid at server level?*
	Yes
	Required keys:
	None
	Optional keys:
	• Process ID
	• Engine Number
	• Key combination:
	  • Database ID
	  • Object ID
	• Key combination:
	  • Procedure Database ID
	  • Procedure ID

Page Number*(KEY)*	

Data item	Required and optional keys

Page Writes	*Valid at server level?*
	Yes
	Required keys:
	None
	Optional keys:
	• Process ID
	• Engine Number
	• Key combination:
	  • Database ID
	  • Object ID
	• Key combination:
	  • Procedure Database ID
	  • Procedure ID

Physical Page Reads	*Valid at server level?*
	Yes
	Required keys:
	None
	Optional keys:
	• Process ID
	• Engine Number
	• Key combination:
	  • Database ID
	  • Object ID
	• Key combination:
	  • Procedure Database ID
	  • Procedure ID

Procedure Buffer Size	*Valid at server level?* Yes *Required keys:* None

Data item	Required and optional keys
Procedure CPU Time	*Valid at server level?* No *Required keys:* • Procedure Database ID • Procedure ID *Optional keys:* • Process ID • Procedure Stmt Number • Procedure Line Number
Procedure Database ID *(KEY)*	
Procedure Database Name[1]	*Valid at server level?* No *Required keys:* Procedure Database ID
Procedure Elapsed Time	*Valid at server level?* No *Required keys:* • Procedure Database ID • Procedure ID *Optional keys:* • Process ID • Procedure Stmt Number • Procedure Line Number
Procedure Execution Class[2]	*Valid at server level?* No *Required keys:* • Procedure Database ID • Procedure ID

Data item	Required and optional keys
Procedure Execution Count	*Valid at server level?* No *Required keys:* • Procedure Database ID • Procedure ID *Optional keys:* • Process ID • Procedure Stmt Number • Procedure Line Number
Procedure Header Size	*Valid at server level?* Yes *Required keys:* None
Procedure Hit Percent	*Valid at server level?* Yes *Required keys:* None
Procedure ID *(KEY)*	
Procedure Line Number *(KEY)*	
Procedure Line Text	*Valid at server level?* No *Required keys:* • Procedure Database ID • Procedure ID
Procedure Logical Reads	*Valid at server level?* Yes *Required keys:* None
Procedure Name[1]	*Valid at server level?* No *Required keys:* • Procedure Database ID • Procedure ID

Data item	Required and optional keys
Procedure Owner Name[1]	*Valid at server level?* No *Required keys:* • Procedure Database ID • Procedure ID
Procedure Physical Reads	*Valid at server level?* Yes *Required keys:* None
Procedure Statement Number *(KEY)*	
Process ID *(KEY)*[3]	
Process State *(KEY)*	
Process State Count	*Valid at server level?* No *Required keys:* Process State
Rows Deleted	*Valid at server level?* Yes *Required keys:* None
Rows Deleted Deferred	*Valid at server level?* Yes *Required keys:* None
Rows Deleted Direct	*Valid at server level?* Yes *Required keys:* None
Rows Inserted	*Valid at server level?* Yes *Required keys:* None

Data item	Required and optional keys
Rows Inserted Clustered	*Valid at server level?*
	Yes
	Required keys:
	None
Rows Inserted Heap	*Valid at server level?*
	Yes
	Required keys:
	None
Rows Updated	*Valid at server level?*
	Yes
	Required keys:
	None
Rows Updated Deferred	*Valid at server level?*
	Yes
	Required keys:
	None
Rows Updated Direct	*Valid at server level?*
	Yes
	Required keys:
	None
Rows Updated Expensive	*Valid at server level?*
	Yes
	Required keys:
	None
Rows Updated In Place	*Valid at server level?*
	Yes
	Required keys:
	None
Rows Updated Not In Place	*Valid at server level?*
	Yes
	Required keys:
	None
Select Statements	*Valid at server level?*
	Yes
	Required keys:
	None

Data item	Required and optional keys
Server Structures Size	*Valid at server level?* Yes *Required keys:* None
SQL Server Name	*Valid at server level?* Yes *Required keys:* None
SQL Server Version	*Valid at server level?* Yes *Required keys:* None
Thread Exceeded Max[2]	*Valid at server level?* Yes *Required keys:* None
Thread Exceeded Max Percent[2]	*Valid at server level?* Yes *Required keys:* None
Thread Max Used[2]	*Valid at server level?* Yes *Required keys:* None
Time Waited on Lock	*Valid at server level?* No *Required keys:* • Process ID • Database ID • Object ID • Lock Status • Page Number *Optional keys:* Lock Type

Data item	Required and optional keys
Timestamp	*Valid at server level?*
	Yes
	Required keys:
	None
Timestamp Datim	*Valid at server level?*
	Yes
	Required keys:
	None
Transactions	*Valid at server level?*
	Yes
	Required keys:
	None

1. These data items infer names from IDs or status types. They are valid in any view containing the required keys, regardless of the other keys that might also be in the view. (For example, Database Name is valid in any view containing the key Database ID; therefore, it would be valid in a view whose keys are Database ID and Process ID.)

2. These data items are available only if you are monitoring Adaptive Server version 11.5 and later.

3. Process ID and Application Name cannot be used in the same view.

4. In versions earlier than 11.5, Cache Name was a key. Currently, Cache Name is a result. A new key, Cache ID, replaces the key Cache Name. For compatibility, Cache Name remains valid as a key. However, for efficiency and for forward compatibility, use Cache ID as the key in new views.

Examples of valid combinations

- Login Name, Process ID, Page I/O
- Process ID, CPU Percent
- Login Name, Process ID
- Deadlock Count, CPU Busy Percent
- Transactions, Rows Deleted, Rows Inserted, Rows Updated
- Current Engine, Login Name, Process ID
- Current Engine, Login Name, Process ID, CPU Percent
- Login Name, Process ID, Page I/O, Connect Time

Examples of invalid combinations

- Current Engine, Process ID, Lock Status Count

  Lock Status Count is not available per process.

- Login Name, Blocking Process ID

  Need to add the other required key data items.

- Net I/O Bytes, Demand Lock

  Demand Lock requires keys and Net I/O Bytes is not valid with any key data items.

- Current Engine, Login Name, Deadlock Count

  Deadlock Count is not compatible with any key data items, but keys are required for Current Engine and Login Name.

Table of valid statistic types for data items

There are six valid statistic types:

- “Value for Sample”
- “Value for Session”
- “Rate for Sample”
- “Rate for Session”
- “Avg for Sample”
- “Avg for Session”

However, all six are not valid for all data items. Table B-3 shows the statistic types that are valid with each data item. In the table:

- A blank cell indicates that the statistic type is not valid with that data item.
- The valid statistic types are indicated by showing the datatype of the resulting data when that statistic type is specified:
  - long and ENUMS indicate integers
  - double indicates floating point numbers
  - charp indicates character strings

Table B-3: Data items and their valid statistic types

Statistic type data item	Value for sample	Value for session	Rate for sample	Rate for session	Avg for sample	Avg for session
Application Execution Class	charp					
Application Name	charp					
Blocking Process ID	long					
Cache Efficiency	double	double				
Cache Hit Pct	double	double				
Cache Hits	long	long	double	double		
Cache ID	long					
Cache Misses	long	long	double	double		
Cache Name	charp					
Cache Prefetch Efficiency	double	double				
Cache Refer and Reuse	long	long	double	double		
Cache Reuse	long	long	double	double		
Cache Reuse Dirty	long	long	double	double		
Cache Size	double					
Cache Spinlock Contention	double	double				
Code Memory Size	long					
Connect Time	long	long				
CPU Busy Percent	double	double				
CPU Percent	double	double				
CPU Time	double	double				
CPU Yields	long	long	double	double		

Statistic type data item	Value for sample	Value for session	Rate for sample	Rate for session	Avg for sample	Avg for session
Current Application Name	charp					
Current Engine	long					
Current Execution Class	charp					
Current Process State	ENUMS					
Current Stmt Batch ID	long					
Current Stmt Batch Text	charp					
Current Stmt Batch Text Byte Offset	long					
Current Stmt Batch Text Enabled	long					
Current Stmt Context ID	long					
Current Stmt CPU Time	double	double				
Current Stmt Elapsed Time	double	double				
Current Stmt Line Number	long					
Current Stmt Locks Granted Immediately	long	long	double	double		
Current Stmt Locks Granted After Wait	long	long	double	double		
Current Stmt Locks Not Granted	long	long	double	double		

Statistic type data item	Value for sample	Value for session	Rate for sample	Rate for session	Avg for sample	Avg for session
Current Stmt Logical Reads	long	long	double	double		
Current Stmt Number	long					
Current Stmt Page I/O	long	long	double	double		
Current Stmt Page Writes	long	long	double	double		
Current Stmt Physical Reads	long	long	double	double		
Current Stmt Procedure Database ID	long					
Current Stmt Procedure Database Name	charp					
Current Stmt Procedure ID	long					
Current Stmt Procedure Name	charp					
Current Stmt Procedure Owner Name	charp					
Current Stmt Procedure Text	charp					
Current Stmt Query Plan Text	charp					
Current Stmt Start Time	charp					
Current Stmt Text Byte Offset	long					
Database ID	long					

Statistic type data item	Value for sample	Value for session	Rate for sample	Rate for session	Avg for sample	Avg for session
Database Name	charp					
Deadlock Count	long	long				
Demand Lock	charp					
Device Hit Percent	double	double				
Device Hits	long	long	double	double		
Device I/O	long	long	double	double		
Device Misses	long	long	double	double		
Device Name	charp					
Device Reads	long	long	double	double		
Device Writes	long	long	double	double		
Elapsed Time	long	long				
Engine Number	long					
Host Name	charp					
Index Logical Reads	long	long	double	double		
Index Physical Reads	long	long	double	double		
Kernel Process ID	long					
Kernel Structures Memory Size	long					
Large I/O Denied	long	long	double	double		
Large I/O Performed	long	long	double	double		
Large I/O Requested	long	long	double	double		
Lock Count	long	long	double	double		

Statistic type data item	Value for sample	Value for session	Rate for sample	Rate for session	Avg for sample	Avg for session
Lock Hit Percent	double	double				
Lock Result	ENUMS					
Lock Results Summarized	ENUMS					
Lock Status	ENUMS					
Lock Status Count	long	long	double	double		
Lock Type	ENUMS					
Locks Being Blocked Count	long					
Locks Granted After Wait	long	long	double	double		
Locks Granted Immediately	long	long	double	double		
Locks Not Granted	long	long	double	double		
Log Contention Percent	double	double				
Logical Page Reads	long	long	double	double		
Login Name	charp					
Most Active Device I/O	long	long	double	double		
Most Active Device Name	charp					
Net Bytes Received	long	long	double	double		
Net Bytes Sent	long	long	double	double		
Net Default Packet Size	long					
Net I/O Bytes	long	long	double	double		

Statistic type data item	Value for sample	Value for session	Rate for sample	Rate for session	Avg for sample	Avg for session
Net Max Packet Size	long					
Net Packet Size Received	double	double	double	double		
Net Packet Size Sent	double	double	double	double		
Net Packets Received	long	long	double	double		
Net Packets Sent	long	long	double	double		
Number of Engines	long					
Number of Processes	long					
Object ID	long					
Object Name	charp					
Object Type	ENUMS					
Owner Name	charp					
Page Cache Size	long					
Page Hit Percent	double	double				
Page I/O	long	long	double	double		
Page Number	long					
Page Writes	long	long	double	double		
Physical Page Reads	long	long	double	double		
Procedure Buffer Size	long					
Procedure CPU Time	double	double			double	double
Procedure Database ID	long					
Procedure Database Name	charp					

Statistic type data item	Value for sample	Value for session	Rate for sample	Rate for session	Avg for sample	Avg for session
Procedure Elapsed Time	double	double			double	double
Procedure Execution Class	charp					
Procedure Execution Count	long	long	double	double		
Procedure Header Size	long					
Procedure Hit Percent	double	double				
Procedure ID	long					
Procedure Line Number	long					
Procedure Line Text	charp					
Procedure Logical Reads	long	long	double	double		
Procedure Name	charp					
Procedure Owner Name	charp					
Procedure Physical Reads	long	long	double	double		
Procedure Statement Number	long					
Process ID	long					
Process State	ENUMS					
Process State Count	long					
Rows Deleted	long	long	double	double		
Rows Deleted Deferred	long	long	double	double		

Statistic type data item	Value for sample	Value for session	Rate for sample	Rate for session	Avg for sample	Avg for session
Rows Deleted Direct	long	long	double	double		
Rows Inserted	long	long	double	double		
Rows Inserted Clustered	long	long	double	double		
Rows Inserted Heap	long	long	double	double		
Rows Updated	long	long	double	double		
Rows Updated Deferred	long	long	double	double		
Rows Updated Direct	long	long	double	double		
Rows Updated Expensive	long	long	double	double		
Rows Updated In Place	long	long	double	double		
Rows Updated Not In Place	long	long	double	double		
Select Statements	long	long	double	double		
Server Structures Size	long					
SQL Server Name	charp					
SQL Server Version	charp					
Thread Exceeded Max	long	long	double	double		

Statistic type data item	Value for sample	Value for session	Rate for sample	Rate for session	Avg for sample	Avg for session
Thread Exceeded Max Percent	double	double				
Thread Max Used	long					
Time Waited on Lock	long					
Timestamp	charp					
Timestamp Datim	datim					
Transactions	long	long	double	double		

APPENDIX C

Specifications for Defining Playback Views

This appendix describes rules and considerations for designing playback views.

Summarization level details

All playback views in a session play back with the same summarization level. You define the summarization level of playback using the *summarization_level* parameter to the hs_create_playback_session command. The summarization levels are:

- raw
- actual
- entire
- user-defined intervals

Raw playback

This option plays back data as it was collected, using the same sample intervals. Choose this option to view raw data as it was recorded. This is the only option available for playing back snapshot data, such as current SQL statement data and per-process status data. See Table C-2 on page 169 for a definitive list of the snapshot data items. (They are the ones with "No" in the "Allowed for Non-raw" column.)

This option is valid only when *target* is client.

Historical Server performs no processing on the data. The time of each playback sample matches exactly the time on the sample from the input session. The data is exactly the same as in the input session.

Raw playback includes only those recording session samples that fall entirely between the playback *start_time* and *end_time*. For example, if the playback session *start_time* is 3 p.m., and an input recording session has 10-minute samples with one starting at 2:55 p.m., the first sample in the playback is the one that starts at 3:05 p.m.

If playback *start_time* is later than the input recording session start time, cumulative session values are played back as they were collected, with no adjustments made. For example, if an input recording session started at 1 p.m., but the playback *start_time* is 3 p.m., the data item "Device I/O" "Total for Session" would reflect the total I/O since 1 p.m.

Actual playback

This option plays back data using the same sample intervals as the input recording sessions. The playback view can use statistic types different from those in the recorded data, and it can include some estimated and calculated data items not in the original view. It cannot include snapshot data, such as process or lock status data and current SQL statement data. See Table C-2 on page 169 for a definitive list of data items that can be converted or added to a playback view.

Choose this option to add or change certain data items when summarization is not required.

This option is valid only when *target* is client.

Historical Server returns samples whose time intervals correspond to those of samples in the input sessions. However, samples might be truncated to comply with the *start_time* or *end_time* specified, or if overlaps occur across input sessions.

If the requested playback start or end times fall in the middle of a sample interval, Historical Server prorates counted values and performs weighted averages on percentages and rates. The weights are the number of seconds that each input sample contributes.

For example, if the playback session *start_time* is 3 p.m., and an input recording session has 10-minute samples with one starting at 2:55 p.m., the first sample in the playback is the one that started at 2:55 p.m., with all values prorated 50 percent to reflect half a sample.

If playback *start_time* is later than the input recording session start time, cumulative session values are played back prorated to reflect the partial session. For example, if an input recording session started at 1 p.m., but the playback *start_time* is 3 p.m., the data item "Device I/O" "Total for Session" would be a prorated value for I/O since 3 p.m.

Entire playback

This option plays back data for all input recording sessions summarized as a single sample. For example, seven input recording sessions would result in a single playback sample. The sample interval is the timespan between the requested playback *start_time* and *end_time*.

The playback view can use statistic types different from those in the recorded data, and it can include some estimated and calculated data items not in the original view. It cannot include snapshot data, such as process or lock status data and current SQL statement data. See Table C-2 on page 169 for a definitive list of data items that can be converted or added to a playback view.

Choose this option to consolidate recorded data, rolling up details into overviews of activity over longer time periods.

If the requested playback start or end times fall in the middle of the input recording session, Historical Server prorates counted values and performs weighted averages on percentages and rates. The weights are the number of seconds that each input sample contributes.

For example, if the playback session *start_time* is 3 p.m. and the *end_time* is 6 p.m., and an input recording session started at 1 p.m. and ended at 9 p.m., Historical Server calculates data item values for the hours between 3 p.m. and 6 p.m. It prorates cumulative (session) counts and uses weighted averages for cumulative percentages and rates if the start and end times specified for playback do not match the start and end times of the input sessions.

Playback with user-defined intervals

This option plays back data summarized into sample intervals of the specified length. The parameter value is the sample interval, specified as:

"*S*"
"*M*:*S*"
"*H*:*M*:*S*"
"*D H*:*M*:*S*"

where:

- *S* is seconds.
- *M* is minutes.
- *H* is hours.
- *D* is days.

All components are numeric and can be one or two digits. Some examples are:

```
"30" (specifies sample intervals of 30 seconds)
"10:0" (specifies sample intervals of 10 minutes)
"8:30:0" (specifies sample intervals of 8 1/2 hours)
"5 0:0:0" (specifies sample intervals of 5 days)
```

The first sample interval starts at *start_time*, and every sample, except possibly the last one, has the specified length. If the requested playback start or end times fall in the middle of a sample interval in the original recording session, Historical Server prorates the recorded data values for the playback sample.The last sample might be shorter, if necessary, to end at the *end_time* specified.

Choose this option to summarize data into any desired granularity. This type of summary can mediate deviations in activity and is useful for observing trends over time.

The playback view can use statistic types different from those in the recorded data, and it can include some estimated and calculated data items not in the original view. See Table C-2 on page 169 for a definitive list of data items that can be converted or added to a playback view.

For each sample interval, Historical Server prorates counted values and performs weighted averages on percentages and rates to align the data with the requested playback sample interval. The weights are the number of seconds that each input sample contributes.

Summary of summarization intervals

Table C-1 summarizes the playback features offered by each of the *summarization_interval* values.

Table C-1: Summarization interval features

Feature	Raw	Actual	Entire	Defined interval
All recorded data items are available for playback.	Yes	No	No	No
Timestamp allowed in playback view.	Yes	Yes	Yes	Yes
Changed statistic types allowed in playback view.	No	Yes	Yes	Yes
Playback to a file (to create a summarized recording session) allowed.	No	No	Yes	Yes
Calculated or estimated data items not in input view allowed in playback view.	No	Yes	Yes	Yes
Session (cumulative) data prorated when playback starts in the middle of an input session.	No	Yes	Yes	Yes
Sample data prorated when playback starts in the middle of an input sample.	No – the sample is omitted	Yes	Yes	Yes
Summarization.	No	No	Yes	Yes
Standardized sample intervals, with appropriate data adjustments.	No	No	No	Yes

Designing playback views

This section describes the following topics:

- Rules for specifying input sessions
- Relationship of input views to playback views
- Rules for defining raw playback views
- Rules for defining non-raw playback views

Rules for specifying input sessions

When you are using multiple input sessions to create a playback session to a file, no time gaps are allowed in input sessions. That is, when *target* is file, no gaps can exist between an input session's end time and the next input session's start time.

For example, if you collected data from 9 a.m. to 5 p.m. every day from Monday through Friday, you could not play back those five recording sessions to create a new, summarized, weekly session. However, if you eliminate the time gaps by collecting data from 9 a.m. to 9 a.m. every day from Monday through Friday, you could use the playback feature to create a new, summarized weekly session. Another way to eliminate the time gaps is to keep the 9 a.m. to 5 p.m. recording session, but add another set of recording sessions scheduled from 5 p.m. to 9 a.m. Use a longer sample interval for the off hours to reduce the volume of data collected.

Relationship of input views to playback views

Use the hs_create_playback_session command to create a playback session. Then use one or more hs_create_playback_view commands to add views to the playback session. A playback session must contain at least one view. It can contain more than one view.

In the hs_create_playback_session command, you specify the input session IDs that you want to include in the playback. The input views are the views that were defined for the input recording sessions. A playback view defines which data items in the corresponding input view you want to play back.

Playback views are derived from the input views. The name of a playback view must match the name of a view in the input sessions. In the case of multiple input sessions, the view must exist in *all* of the input sessions, and the view definitions in all of the input sessions must contain exactly the same list of data items and the same list of filters.

Rules for defining views

The rules for defining playback views are different depending on whether the *summarization_interval* defined for the playback session is raw or non-raw.

Rules for defining raw playback views

When the *summarization_interval* for the playback session is raw:

- All key data items in the input view *must* be included in the playback view. The column labeled "Key" in Table C-2 indicates which data items are keys. The only valid statistic type for a key is "Value for Sample."
- The result data items in the input view are optional in the playback view.
- The statistic types for all data items in the playback view must be the same as those used for the input view.
- Any playback view can include these data items:

  "Timestamp" "Value for Sample"
  "Timestamp Datim" "Value for Sample"
  "Elapsed Time" "Value for Sample"
  "Elapsed Time" "Value for Session"

If you want to play back all data items in the input view and do not need to add Timestamp, use the hs_create_playback_view command without the *data_item_name_n* and *data_item_stat_n* parameters. The default when you omit these parameters is to define the playback view using all of the data items from the input view of the same name.

Rules for defining non-raw playback views

When *summarization_interval* is entire, actual, or a summary interval:

- All key data items in the input view *must* be included in the playback view. The column labeled "Key" in Table C-2 on page 169 indicates which data items are keys. The only valid statistic type for keys is "Value for Sample".

- Any playback view can include these data items:

  "Timestamp" "Value for Sample"
  "Timestamp Datim" "Value for Sample"
  "Elapsed Time" "Value for Sample"
  "Elapsed Time" "Value for Session"

- The column labeled "Valid for Non-raw?" in Table C-2 indicates which result data items are allowed in a non-raw playback view. The column labeled "Conditions for Inclusion" represents conditions under which the data item is allowed.

- If the "Conditions for Inclusion" column indicates "Full", the full range of statistic types valid for that data item is available to the playback view, regardless of the statistic type used in the input view. Use Table B-3 on page 152 to determine valid statistic types for each data item.

- If the "Conditions for Inclusion" column indicates "Estimated", then both of the following conditions must be true if you want to include the data item in the playback view:

  - The statistic type for the data item must be "Value for Sample" in the input view, *and*

  - The playback session must be defined to allow estimations. (In the hs_create_playback_session command, the *allow_estimation* parameter must be allow.)

  The full range of statistic types valid for the data item is available for the playback view, regardless of the statistic type used in the input view.

- If the "Conditions for Inclusion" column indicates "Calculated", then one of the following conditions must be true if you want to include the data item in the playback view:

  - The data items listed in the column labeled "Other Data Items Needed for Calculation" exist in the input view. If this is true, Historical Server calculates values during playback, *or*

  - If the data items required for calculation do not exist in the input view, Historical Server can estimate values during playback. For this to happen, the conditions for estimating, described previously, must be true.

  The full range of statistic types valid for the data item is available for the playback view, regardless of the statistic type used in the input view.

Table of data item requirements for playback views

Table C-2 describes the requirements for including data items in a playback view. Read the previous section for explanations of the columns.

Table C-2: Data item requirements for playback views

Data item (keys in an input view are required in a playback view)	Valid for non-raw?	Conditions for inclusion
Application Execution Class	Valid	Value for Sample
Application Name *(KEY)*		
Blocking Process ID	No	
Cache Efficiency	Valid	Estimated
Cache Hit Pct	Valid	Calculated from: • Cache Hits • Cache Misses
Cache Hits	Valid	Full
Cache ID *(KEY)*[1]		
Cache Misses	Valid	Full
Cache Name[1]	Valid	Value for Sample
Cache Prefetch Efficiency	Valid	Estimated
Cache Refer And Reuse	Valid	Full
Cache Reuse	Valid	Full
Cache Reuse Dirty	Valid	Full
Cache Size	Valid	Value for Sample
Cache Spinlock Contention	Valid	Estimated
Code Memory Size	Valid	Value for Sample
Connect Time	Valid	Full
CPU Busy Percent	Valid	Estimated
CPU Percent	Valid	Estimated
CPU Time	Valid	Full
CPU Yields		
Current Application Name		
Current Engine	No	
Current Execution Class	No	
Current Process State	No	
Current Stmt Batch ID	No	
Current Stmt Batch Text	No	
Current Stmt Batch Text Byte Offset	No	
Current Stmt Batch Text Enabled	No	

Data item (keys in an input view are required in a playback view)	Valid for non-raw?	Conditions for inclusion
Current Stmt Cache Reads	No	
Current Stmt Context ID	No	
Current Stmt CPU Time	No	
Current Stmt Elapsed Time	No	
Current Stmt Line Number	No	
Current Stmt Locks Granted Immediately	No	
Current Stmt Locks Granted After Wait	No	
Current Stmt Locks Not Granted	No	
Current Stmt Max Text Config Size	No	
Current Stmt Max Text Default Size	No	
Current Stmt Max Text Run Size	No	
Current Stmt Number	No	
Current Stmt Page I/O	No	
Current Stmt Page Writes	No	
Current Stmt Physical Reads	No	
Current Stmt Procedure Database ID	No	
Current Stmt Procedure Database Name	No	
Current Stmt Procedure ID	No	
Current Stmt Procedure Name	No	
Current Stmt Procedure Owner Name	No	
Current Stmt Procedure Text	No	
Current Stmt Query Plan Text	No	
Current Stmt Start Time	No	
Current Stmt Text Byte Offset	No	
Database ID *(KEY)*	Valid	
Database Name	Valid	Value for Sample
Deadlock Count	Valid	Full
Demand Lock	No	
Device Hit Percent	Valid	Calculated from: • Device Hits • Device Misses
Device Hits	Valid	Full
Device I/O	Valid	Full
Device Misses	Valid	Full
Device Name *(KEY)*	Valid	Value for Sample
Device Reads	Valid	Full

Data item (keys in an input view are required in a playback view)	Valid for non-raw?	Conditions for inclusion
Device Writes	Valid	Full
Elapsed Time[2]	Valid	Full
Engine Number *(KEY)*	Valid	Value for Sample
Host Name	No	
Index Logical Reads	Valid	Full
Index Physical Reads	Valid	Full
Kernel Process ID *(KEY)*	Valid	Value for Sample
Kernel Structures Memory Size	Valid	Value for Sample
Large I/O Denied	Valid	Full
Large I/O Performed	Valid	Full
Large I/O Requested	Valid	Full
Lock Count	Valid	Full
Lock Hit Percent	Valid	Estimated
Lock Result *(KEY)*	Valid	Value for Sample
Lock Results Summarized *(KEY)*	Valid	Value for Sample
Lock Status *(KEY)*	Valid	Value for Sample
Lock Status Count	No	
Lock Type *(KEY)*	Valid	Value for Sample
Locks Being Blocked Count	No	
Locks Granted Immediately	Valid	Full
Locks Granted After Wait	Valid	Full
Locks Not Granted	Valid	Full
Log Contention Percent	Valid	Estimated
Logical Page Reads	Valid	Full
Login Name	Valid	Value for Sample
Most Active Device I/O	No	
Most Active Device Name	No	
Net Bytes Received	Valid	Full
Net Bytes Sent	Valid	Full
Net Default Packet Size	Valid	Value for Sample
Net I/O Bytes	Valid	Full
Net Max Packet Size	Valid	Value for Sample

Data item (keys in an input view are required in a playback view)	Valid for non-raw?	Conditions for inclusion
Net Packet Size Received	Valid	Calculated from: • Net Packets Received • Net Bytes Received
Net Packet Size Sent	Valid	Calculated from: • Net Packets Sent • Net Bytes Sent
Net Packets Received	Valid	Full
Net Packets Sent	Valid	Full
Number of Engines	Valid	Value for Sample
Number of Processes	Valid	Estimated
Object ID *(KEY)*	Valid	Value for Sample
Object Name	Valid	Value for Sample
Object Type	Valid	Value for Sample
Owner Name	Valid	Value for Sample
Page Cache Size	Valid	Value for Sample
Page Hit Percent	Valid	Calculated from: • Logical Reads • Physical Reads
Page I/O	Valid	Full
Page Number *(KEY)*	Valid	Value for Sample
Page Writes	Valid	Full
Physical Page Reads	Valid	Full
Procedure Buffer Size	Valid	Value for Sample
Procedure CPU Time		
Procedure Database ID *(KEY)*	Valid	Value for Sample
Procedure Database Name	Valid	Full
Procedure Elapsed Time	Valid	Requires: Procedure Execution Count See “Using Procedure Elapsed Time and Procedure CPU Time” in this document for more information.
Procedure Execution Class	No	
Procedure Execution Count	Valid	Full

Data item (keys in an input view are required in a playback view)	Valid for non-raw?	Conditions for inclusion
Procedure Header Size	Valid	Value for Sample
Procedure Hit Percent	Valid	Calculated from: • Procedure Logical Reads • Procedure Physical Reads
Procedure ID *(KEY)*	Valid	Value for Sample
Procedure Line Number *(KEY)*	Valid	Value for Sample
Procedure Line Text	Valid	Value for Sample
Procedure Logical Reads	Valid	Full
Procedure Name	Valid	Value for Sample
Procedure Owner Name	Valid	Value for Sample
Procedure Physical Reads	Valid	Full
Procedure Statement Number *(KEY)*	Valid	Value for Sample
Process ID *(KEY)*	Valid	Value for Sample
Process State *(KEY)*	Valid	Value for Sample
Process State Count	No	
Rows Deleted	Valid	Full
Rows Deleted Deferred	Valid	Full
Rows Deleted Direct	Valid	Full
Rows Inserted	Valid	Full
Rows Inserted Clustered	Valid	Full
Rows Inserted Heap	Valid	Full
Rows Updated	Valid	Full
Rows Updated Deferred	Valid	Full
Rows Updated Direct	Valid	Full
Rows Updated Expensively	Valid	Full
Rows Updated In Place	Valid	Full
Rows Updated Not In Place	Valid	Full
Select Statements	Valid	Full
Server Structures Size	Valid	Value for Sample
SQL Server Name	Valid	Value for Sample
SQL Server Version	Valid	Value for Sample
Thread Exceeded Max	Valid	Full
Thread Exceeded Max Percent	Valid	Full
Thread Max Used	No	

Data item (keys in an input view are required in a playback view)	Valid for non-raw?	Conditions for inclusion
Time Waited on Lock	No	
Timestamp[2]	Valid	Value for Sample
Timestamp Datim[2]	Valid	Value for Sample
Transactions	Valid	Full

1. In versions earlier than 11.5, Cache Name was a key. Currently, Cache Name is a result. A new key, Cache ID, replaces the key Cache Name. For compatibility, Cache Name remains valid as a key. However, for efficiency and for forward compatibiltiy, use Cache ID as the key in new views.

2. These data items can be included in any playback view even if not present in the input view.

Additional information about some data items

This section provides additional information about using some data items in views.

Using “Timestamp”, “Timestamp Datim”, and “Elapsed Time”

The following data items can always be included in a playback view, even if they were not in the input view:

“Timestamp” “Value for Sample”
“Timestamp Datim” “Value for Sample”
“Elapsed Time” “Value for Sample”
“Elapsed Time” “Value for Session”

You can create a playback view consisting of *only* one or more of these time-related data items. To do this, you need a valid view name from the input session, which means that there must be some view in the input sessions that you do not intend to play back.

When *summarization_level* is raw, “Timestamp” and “Timestamp Datim” values are the same as those in the original data file, which is the time as reported by Adaptive Server when the recording was originally made. When *summarization_level* is actual, entire, or a user-defined interval, “Timestamp” and “Timestamp Datim” values reflect the time zone on the Historical Server system where the summarization was made.

Using Process ID

When a server process terminates, Adaptive Server can reuse its Process ID for a new process. Therefore, the Process ID data item is not guaranteed to uniquely identify a process. The Kernel Process ID data item, however, uniquely identifies a process.

To create a non-raw playback view that shows per-process data, the input view and the playback view must include both the Process ID and Kernel Process ID data items. If playback is raw, using only Process ID is allowed.

Views that include Process ID return rows as follows:

- Recording session views and raw playback views.

  These views return a row only for process IDs that exist at the end of a sample interval. If a server process terminates in the middle of a sample interval, a row is not returned for its process ID.

- Playback views when the session's *summarization_level* parameter is actual, entire, or user-defined intervals.

  These views return rows for all server processes included in any of the input views. However, since the Process IDs are not guaranteed to be unique, the Kernel Process ID data item must also be included to ensure uniqueness of the key. Otherwise, the view could erroneously summarize two different processes.

Using Procedure Elapsed Time and Procedure CPU Time

This section discusses how to use the data items that can have the "Avg for Sample" and "Avg for Session" statistic types. The data items are:

- Procedure Elapsed Time
- Procedure CPU Time

If the playback view uses these data item names with the "Value for Sample" or "Value for Session" statistic types, the input view must have the same statistic type.

If the playback view uses these data item names with the "Avg for Sample" or "Avg for Session" statistic types:

- The returned value is *calculated* if the input view includes the additional data item Procedure Execution Count.
- The returned value is *estimated* if all of the following are true:

- Procedure Execution Count does not exist in the input view, and
- Procedure Elapsed Time exists in the input view with the statistic type "Avg for Sample", and
- The playback session allows estimations. That is, in the hs_create_playback_session command, the *allow_estimation* parameter was set to allow.

APPENDIX D

Examples of Recording Session Views

The appendix contains examples of valid views for Historical Server. These views also appear in the *views* file that was installed in the *sample/histserver* subdirectory under the installation directory.

You may find that some of these views collect exactly the information you need, while others can serve as templates for building the views that you need.

Some of the sample views differ from one another only in the time interval over which the data is accumulated (either duration of the recording session, or just the most recent sample interval). Other views contain similar data items but in different orders. The order in which data items appear in a view is significant because the data is sorted according to the key fields. The first key field that appears in a view's definition acts as the primary sort key, the second key field is the secondary sort key, and so on.

Do *not* use the *views* file directly as input to isql, for the following reasons:

- It is unlikely that you would want to define all of the views contained in this file for a single recording session.
- The file contains comment lines intermingled with lines that contain the actual text of the view definitions. Historical Server does not understand comment lines. As you cut and paste the view definitions, discard the comment lines.

Cache performance summary

This view represents the overall effectiveness of Adaptive Server caches during the most recent sample interval. It shows the percentage of data page reads that are satisfied from the Adaptive Server data caches, as well as the percentage of requests for procedure executions that are satisfied from the Adaptive Server procedure cache.

```
hs_create_view cache_perf_sum,
"Page Hit Percent", "Value for Sample",
"Procedure Hit Percent", "Value for Sample"
```

Database object lock status

This view shows the status of locks on database objects that are held or being requested by Adaptive Server processes, as of the end of the most recent sample interval. Each lock is identified by the name and ID of the object being locked, the name and ID of the database that contains that object, and the page number to which the lock applies (if it is a page lock).

Each Adaptive Server process associated with the lock is also identified by its login name, Process ID, and Kernel Process ID. The type of lock is presented, together with the current status of the lock and an indication of whether or not this is a demand lock. If the lock is being requested by the process, the amount of time that this process waited to acquire the lock and the Process ID of the process that already holds the lock are shown. If, instead, the process already holds the lock, the count of other processes waiting to acquire that lock is shown.

```
hs_create_view object_lock_status,
"Database ID", "Value for Sample",
"Database Name", "Value for Sample",
"Object ID", "Value for Sample",
"Object Name", "Value for Sample",
"Page Number", "Value for Sample",
"Login Name", "Value for Sample",
"Process ID", "Value for Sample",
"Kernel Process ID", "Value for Sample",
"Lock Type", "Value for Sample",
"Lock Status", "Value for Sample",
"Demand Lock", "Value for Sample",
"Time Waited on Lock","Value for Sample",
"Blocking Process ID", "Value for Sample",
"Locks Being Blocked Count", "Value for Sample"
```

Database object page I/O

This view shows objects in the Adaptive Server databases and the page I/Os associated with them. It shows the Adaptive Server database name and ID and the object names and IDs within each database. This view shows for each object the associated logical page reads, physical page reads, and page writes for both the most recent sample interval and for the recording session.

```
hs_create_view object_page_io,
"Database ID", "Value for Sample",
```

```
"Database Name", "Value for Sample",
"Object ID", "Value for Sample",
"Object Name", "Value for Sample",
"Logical Page Reads", "Value for Sample",
"Physical Page Reads", "Value for Sample",
"Page Writes", "Value for Sample",
"Logical Page Reads", "Value for Session",
"Physical Page Reads", "Value for Session",
"Page Writes", "Value for Session"
```

Data cache activity for individual caches

This view is available only for SQL Server version 11.0.x and Adaptive Server version 11.5 and later. It contains information about the performance of individual data caches.

For each named cache configured in the Adaptive Server, and also for the default data cache, this view collects the cache's name and the percentage of page reads for objects bound to the cache that were satisfied from the cache since the start of the recording session.

This view also presents a measure of the efficiency of the cache's space utilization, and the percentage of times when an attempt to acquire the cache's spinlock was forced to wait, since the start of the session. The actual number of cache hits and misses for the session are also collected.

```
hs_create_view data_cache_sum,
"Cache Name", "Value for Sample",
"Cache Hit Pct", "Value for Session",
"Cache Efficiency", "Value for Session",
"Cache Spinlock Contention", "Value for Session",
"Cache Hits", "Value for Session",
"Cache Misses", "Value for Session"
```

Data cache statistics for recording session

This view represents the overall effectiveness of the combined data caches of the Adaptive Server since the start of the recording session. It shows the percentage of requests for page reads that were satisfied from cache for the recording session.

The view also shows the number and rate of logical page reads, physical page reads, and page writes for the recording session.

```
hs_create_view session_page_cache_stats,
"Page Hit Percent", "Value for Session",
"Logical Page Reads", "Value for Session",
"Logical Page Reads", "Rate for Session",
"Physical Page Reads", "Value for Session",
"Physical Page Reads", "Rate for Session",
"Page Writes", "Value for Session",
"Page Writes", "Rate for Session"
```

Data cache statistics for sample interval

This view represents the overall effectiveness of the combined data caches of the Adaptive Server for the most recent sample interval. It shows the percentage of requests for page reads that were satisfied from cache for the most recent sample interval.

The view also shows the number and rate of logical page reads, physical page reads, and page writes for the most recent sample interval.

```
hs_create_view sample_page_cache_stats,
"Page Hit Percent", "Value for Sample",
"Logical Page Reads", "Value for Sample",
"Logical Page Reads", "Rate for Sample",
"Physical Page Reads", "Value for Sample",
"Physical Page Reads", "Rate for Sample",
"Page Writes", "Value for Sample",
"Page Writes", "Rate for Sample"
```

Device I/O for recording session

This view represents the I/O activity that occurred on Adaptive Server database devices since the start of the recording session. It identifies each device by name. Device I/O levels are presented in two ways: as counts of total device I/Os, reads and writes since the start of the session, and also as overall rates of total I/Os, reads and writes per second since the session began.

```
hs_create_view session_device_io,
"Device Name", "Value for Sample",
"Device Reads", "Value for Session",
"Device Writes", "Value for Session",
"Device I/O", "Value for Session",
"Device Reads", "Rate for Session",
"Device Writes", "Rate for Session",
"Device I/O", "Rate for Session"
```

Device I/O for sample interval

This view represents the I/O activity that occurred on the Adaptive Server database devices during the most recent sample interval. It identifies each device by name. Device I/O levels are presented in two ways: as counts of total device I/Os, reads and writes during the most recent sample interval, and also as rates of total I/Os, reads and writes per second during the sample interval.

```
hs_create_view sample_device_io,
"Device Name", "Value for Sample",
"Device I/O", "Value for Sample",
"Device Reads", "Value for Sample",
"Device Writes", "Value for Sample",
"Device I/O", "Rate for Sample",
"Device Reads", "Rate for Sample",
"Device Writes", "Rate for Sample"
```

Device I/O performance summary

This view represents reads and writes to database devices by Adaptive Server, accumulated from the start of the recording session. It shows the overall rate of reads and writes to database devices since the start of the session, as well as the most active database device for that time period and the rate of reads and writes to the most active device.

```
hs_create_view device_perf_sum,
"Device I/O", "Rate for Session",
"Most Active Device Name", "Value for Session",
"Most Active Device I/O", "Rate for Session"
```

Engine activity

This view shows the level of activity for each active Adaptive Server engine during the most recent sample interval. For each engine, the percentage of the sample interval, when that engine used the CPU, is presented.

Also shown are the number of logical page reads, physical page reads, and page writes that were generated by that engine during the sample interval.

```
hs_create_view engine_activity,
"Engine Number", "Value for Sample",
"CPU Busy Percent", "Value for Sample",
"Logical Page Reads", "Value for Sample",
"Physical Page Reads", "Value for Sample",
"Page Writes", "Value for Sample"
```

Lock performance summary

This view represents the total number of locks of each type requested and granted during the most recent sample interval.

```
hs_create_view lock_perf_sum,
"Lock Type", "Value for Sample",
"Lock Results Summarized", "Value for Sample",
"Lock Count", "Value for Sample"
```

Network activity for recording session

This view represents the network activity over all of the Adaptive Server network connections since the start of the recording session. It shows the default packet size, the maximum packet size, and both average packet sizes sent and received since the start of the session. The view shows the number of packets sent, the number of packets received, and the rate at which packets were sent and received. It also shows the number of bytes sent and the number of bytes received and the rate at which bytes were sent and received.

```
hs_create_view session_network_activity,
"Net Default Packet Size", "Value for Sample",
"Net Max Packet Size", "Value for Sample",
"Net Packet Size Sent", "Value for Session",
"Net Packet Size Received", "Value for Session",
"Net Packets Sent", "Value for Session",
"Net Packets Received", "Value for Session",
"Net Packets Sent", "Rate for Session",
"Net Packets Received", "Rate for Session",
"Net Bytes Sent",  "Value for Session",
"Net Bytes Received", "Value for Session",
"Net Bytes Sent", "Rate for Session",
"Net Bytes Received", "Rate for Session"
```

Network activity for sample interval

This view represents the network activity over all of the Adaptive Server network connections during the most recent sample interval. It shows the default packet size, the maximum packet size, and both average packet sizes sent and received for the sample interval. The view shows the number of packets sent, the number of packets received, and the rate at which packets were sent and received. It also shows the number of bytes sent and the number of bytes received and the rate at which bytes were sent and received.

```
hs_create_view sample_network_activity,
"Net Default Packet Size", "Value for Sample",
"Net Max Packet Size", "Value for Sample",
"Net Packet Size Sent", "Value for Sample",
"Net Packet Size Received", "Value for Sample",
"Net Packets Sent", "Value for Sample",
"Net Packets Received", "Value for Sample",
"Net Packets Sent", "Rate for Sample",
```

```
"Net Packets Received", "Rate for Sample",
"Net Bytes Sent", "Value for Sample",
"Net Bytes Received", "Value for Sample",
"Net Bytes Sent", "Rate for Sample",
"Net Bytes Received", "Rate for Sample"
```

Network performance summary

This view represents the rate of Adaptive Server activity over all its network connections during the most recent sample interval. It shows the number of bytes per second that were received by and sent by the Adaptive Server during the interval.

```
hs_create_view network_perf_sum,
"Net Bytes Received", "Rate for Sample",
"Net Bytes Sent", "Rate for Sample"
```

Page I/O

This view enables you to determine which tables experienced the highest number of page reads over the duration of the recording session. It also allows you to track activity against tables at different times of day because page reads are also accumulated on a per-sample basis. You can set the sample interval for the recording session to whatever granularity is most useful for establishing activity trends at different times of day.

You may want to accumulate only total page reads for the entire recording session. In that case, you can set the sample interval to nearly the length of the recording session, and drop “Logical Page Reads”, “Value for Sample” from the view.

```
hs_create_view page_ios,
"Database ID", "Value for Sample",
"Database Name", "Value for Sample",
"Object ID", "Rate for Sample",
"Logical Page Reads", "Value for Sample",
```

```
"Logical Page Reads", "Rate for Session"
```

Note The count of the page I/Os collected by this view for a given database table includes I/Os against indexes on the table, in addition to those performed against the data pages of the table itself.

Procedure cache statistics for recording session

This view represents the effectiveness of the procedure cache of the Adaptive Server since the start of the recording session. It shows the percentage of requests for stored procedure executions that were satisfied by the procedure cache.

The view also shows the number of logical reads and physical reads of stored procedures since the start of the session, and the overall rate of logical and physical reads of stored procedures since the start of the session.

```
hs_create_view session_procedure_cache_stats,
"Procedure Hit Percent", "Value for Session",
"Procedure Logical Reads", "Value for Session",
"Procedure Logical Reads", "Rate for Session",
"Procedure Physical Reads", "Value forSession",
"Procedure Physical Reads", "Rate for Session"
```

Procedure cache statistics for sample interval

This view represents the effectiveness of the procedure cache of the Adaptive Server for the most recent sample interval. It shows the percentage of requests for stored procedure executions that were satisfied by the procedure cache for the most recent sample interval.

The view also shows the number of logical reads and physical reads of stored procedures during the most recent sample interval, and the rate of logical and physical reads of stored procedures for the most recent sample interval.

```
hs_create_view sample_procedure_cache_stats,
"Procedure Hit Percent", "Value for Sample",
"Procedure Logical Reads", "Value for Sample",
```

```
"Procedure Logical Reads", "Rate for Sample",
"Procedure Physical Reads", "Value for Sample",
"Procedure Physical Reads", "Rate for Sample"
```

Procedure page I/O

This view represents page I/Os that occurred while running stored procedures during the most recent sample interval. For each stored procedure that generated page I/Os during the sample interval, it shows the stored procedure name and ID, together with the name and ID of the database that contains the procedure. If page I/Os were produced when no stored procedure was active, those I/Os are associated with procedure ID and database ID values of zero.

This view also shows, on a per stored procedure level, the total page I/Os, the percentage of page I/O requests that could be satisfied by the Adaptive Server data caches, and the number of logical page reads, physical page reads, and page writes generated while executing the stored procedures during the most recent sample interval.

```
hs_create_view procedure_page_cache_io,
"Procedure Database Name", "Value for
Sample","Procedure Database ID", "Value for
Sample","Procedure Name", "Value for Sample","Procedure
ID", "Value for Sample","Page I/O", "Value for
Sample","Page Hit Percent", "Value for Sample","Logical
Page Reads", "Value for Sample","Physical Page Reads",
"Value for Sample","Page Writes", "Value for Sample"
```

Process activity

This view shows the CPU utilization, page I/Os, and current process state for all processes in the Adaptive Server.

For the most recent sample interval, the login name, Process ID and Kernel Process ID are given for each process, together with its current process state.

The view also presents each process's connect time, total page I/Os and CPU utilization time, accumulated since the start of the recording session.

```
hs_create_view process_activity,
```

```
"Login Name", "Value for Sample",
"Process ID", "Value for Sample",
"Kernel Process ID", "Value for Sample",
"Connect Time", "Value for Session",
"Page I/O", "Value for Session",
"CPU Time", "Value for Session",
"Current Process State", "Value for Sample"
```

Process database object page I/O

This view represents the page I/Os by database object for each Adaptive Server process. It shows the login name, Process ID, and Kernel Process ID for each process that had page I/Os during the most recent sample interval.

For each such process and for each database object it accessed, the view also shows the object name, object ID, and the object's database name and ID, plus the page I/Os associated the object.

The view also shows the total page I/Os, the percentage of page I/O requests that could be satisfied by the Adaptive Server cache, and the number of logical page reads, physical page reads, and page writes for the most recent sample interval.

```
hs_create_view process_object_page_io,
"Login Name", "Value for Sample",
"Process ID", "Value for Sample",
"Kernel Process ID", "Value for Sample",
"Database Name", "Value for Sample",
"Database ID", "Value for Sample",
"Object Name", "Value for Sample",
"Object ID", "Value for Sample",
"Object Type", "Value for Sample",
"Page I/O", "Value for Sample",
"Page Hit Percent", "Value for Sample",
"Logical Page Reads", "Value for Sample",
"Physical Page Reads", "Value for Sample",
"Page Writes", "Value for Sample"
```

Process detail for locks

This view shows the status of locks held by or being requested by Adaptive Server processes as of the end of the most recent sample interval. Each lock is identified by the login name, Process ID, and Kernel Process ID of the Adaptive Server process associated with the lock, as well as the name and ID of the object being locked, the name and ID of the database that contains that object, and the page number to which the lock applies (if it is a page lock). The current status of each lock is presented, as is an indication of whether or not this is a demand lock. If the lock is being requested by the process, the amount of time that this process waited to acquire the lock and the Process ID of the process that already holds the lock are shown. If instead the process already holds the lock, the count of other processes waiting to acquire that lock is shown.

```
hs_create_view process_detail_locks,
"Login Name", "Value for Sample",
"Process ID", "Value for Sample",
"Kernel Process ID", "Value for Sample",
"Database Name", "Value for Sample",
"Database ID", "Value for Sample",
"Object Name", "Value for Sample",
"Object ID", "Value for Sample",
"Page Number", "Value for Sample",
"Lock Status", "Value for Sample",
"Demand Lock", "Value for Sample",
"Time Waited on Lock", "Value for Sample",
"Blocking Process ID", "Value for Sample",
"Locks Being Blocked Count", "Value for Sample"
```

Process detail page I/O

This view represents the page I/Os for each Adaptive Server process in detail. The login name, Process ID, Kernel Process ID, current process state and current engine are shown for each Adaptive Server process as of the end of the most recent sample interval. The view shows the percentage of page I/O requests that could be satisfied by the Adaptive Server data caches, both for the sample interval and since the start of the session.

This view also presents the number of logical page reads, physical page reads, and page writes accumulated since the start of the recording session.

```
hs_create_view process_detail_io,
"Login Name", "Value for Sample",
"Process ID", "Value for Sample",
"Kernel Process ID", "Value for Sample",
"Current Process State", "Value for Sample",
"Current Engine", "Value for Sample",
"Connect Time", "Value for Session",
"CPU Time", "Value for Session",
"Page Hit Percent", "Value for Sample",
"Page Hit Percent", "Value for Session",
"Logical Page Reads", "Value for Session",
"Physical Page Reads", "Value for Session",
"Page Writes", "Value for Session"
```

Process locks

This view shows the count of lock requests for every process in the Adaptive Server that generated lock requests during the most recent sample interval.

```
hs_create_view process_lock,
"Login Name", "Value for Sample",
"Process ID", "Value for Sample",
"Kernel Process ID", "Value for Sample",
"Lock Count", "Value for Sample"
```

Process page I/O

This view represents the page I/Os summarized on a per Adaptive Server process level for the most recent sample. It shows the login name, Process ID, and Kernel Process ID for each process in the Adaptive Server that generated page I/Os during the interval.

This view also shows, on a per process level, the total page I/Os, the percentage of page I/O requests that could be satisfied by the Adaptive Server data caches, and the number of logical page reads, physical page reads, and writes for the most recent sample interval.

```
hs_create_view process_page_io,
"Login Name", "Value for Sample",
"Process ID", "Value for Sample",
```

```
"Kernel Process ID", "Value for Sample",
"Page I/O", "Value for Sample",
"Page Hit Percent", "Value for Sample",
"Logical Page Reads", "Value for Sample",
"Physical Page Reads", "Value for Sample",
"Page Writes", "Value for Sample"
```

Process state summary

This view shows the number of processes that were in each process state at the end of the most recent sample interval.

```
hs_create_view process_perf_sum,
"Process State", "Value for Sample",
"Process State Count", "Value for Sample"
```

Process stored procedure page I/O

This view represents the page I/Os associated with stored procedure executions by Adaptive Server processes. It shows the login name, Process ID, and Kernel Process ID for each process that generated page I/Os during the sample interval. For each process and stored procedure that generated page I/Os, it shows the name and ID of the database that contains the stored procedure, as well as the name and ID of the procedure itself.

The view also shows the total page I/Os, the percentage of page I/O requests that could be satisfied from data caches, and the number of logical page reads, physical page reads, and page writes for the most recent sample interval.

```
hs_create_view process_procedure_page_io,
"Login Name", "Value for Sample",
"Process ID", "Value for Sample",
"Kernel Process ID", "Value for Sample",
"Procedure Database Name", "Value for Sample",
"Procedure Database ID", "Value for Sample",
"Procedure Name", "Value for Sample",
"Procedure ID", "Value for Sample",
"Page I/O", "Value for Sample",
"Page Hit Percent", "Value for Sample",
"Logical Page Reads", "Value for Sample",
```

```
"Physical Page Reads", "Value for Sample",
"Page Writes", "Value for Sample"
```

Server performance summary

This view represents overall Adaptive Server performance. It shows the number of lock requests per second, the percentage of the sample interval when the Adaptive Server was busy, the number of transactions processed per second, and the number of times the Adaptive Server detected a deadlock during the most recent sample interval.

```
hs_create_view server_perf_sum,
"Lock Count", "Rate for Sample",
"CPU Busy Percent", "Value for Sample",
"Transactions", "Rate for Sample",
"Deadlock Count", "Value for Sample"
```

Stored procedure activity

This view shows stored procedure activity at the level of procedure statement. Each statement of any stored procedure that was executed during the most recent sample interval is identified by the name and ID of the database that contains the procedure, the name and ID of the procedure, the relative number of the statement within the procedure, and the line of the procedure's text on which the statement begins.

The view includes the number of times each statement was executed, both during the most recent sample interval and since the start of the recording session. It also contains the average elapsed time needed to execute the statement, both for the sample interval and for the recording session so far.

```
hs_create_view procedure_activity,
"Procedure Database ID", "Value for Sample",
"Procedure Database Name", "Value for Sample",
"Procedure ID", "Value for Sample",
"Procedure Name", "Value for Sample",
"Procedure Line Number", "Value for Sample",
"Procedure Statement Number", "Value for Sample",
"Procedure Execution Count", "Value for Sample",
```

```
"Procedure Execution Count", "Value for Session",
"Procedure Elapsed Time", "Avg for Sample",
"Procedure Elapsed Time", "Avg for Session"
```

Transaction activity

This view details the transaction activity that occurred in Adaptive Server, both for the sample interval and the recording session.

```
hs_create_view transaction_activity,
"Transactions", "Value for Sample",
"Rows Deleted", "Value for Sample",
"Rows Inserted", "Value for Sample",
"Rows Updated", "Value for Sample",
"Rows Updated Directly", "Value for Sample",
"Transactions", "Value for Session",
"Rows Deleted", "Value for Session",
"Rows Inserted", "Value for Session",
"Rows Updated", "Value for Session",
"Rows Updated Directly", "Value for Session",
"Transactions", "Rate for Sample",
"Rows Deleted", "Rate for Sample",
"Rows Inserted", "Rate for Sample",
"Rows Updated", "Rate for Sample",
"Rows Updated Directly", "Rate for Sample",
"Transactions", "Rate for Session",
"Rows Deleted", "Rate for Session",
"Rows Inserted", "Rate for Session",
"Rows Updated", "Rate for Session",
"Rows Updated Directly", "Rate for Session"
```

Index

Symbols

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W