ADABAS SQL Server

Installation and Operations Manual for Windows NT

SOFTWARE AG

Manual Order Number: ESQ142-010WNT

This document applies to ADABAS SQL Server Version 1.4 for Windows NT and to all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Readers’ comments are welcomed. Comments may be addressed to the Documentation Department at the address on the back cover.

SOFTWARE AG documentation often refers to numerous hardware and software products by their trade names. In most cases, if not all, these designations are claimed as trademarks or registered trademarks by their respective companies.

TABLE OF CONTENTS

PREFACE

The ADABAS SQL Server is SOFTWARE AG's implementation of the ANSI/ISO Standard for the SQL database language.

The primary goal of the ADABAS SQL Server is to provide an ANSI/ISO compatible database language interface to SOFTWARE AG's database management system ADABAS.

Using This Manual – Some Basic Information

This manual is part of a set of four ADABAS SQL Server manuals. The *Reference Manual*, *Programmer's Guide* and the *Messages and Codes Manual* hold information valid for all platforms supported by the ADABAS SQL Server. *The Installation and Operations Manuals* are an exception to that rule and hold platform-dependent information only.

This manual describes the installation and operation of the ADABAS SQL Server on the Windows NT platform.

The following is a summary of the manual's chapters and their contents:

Chapter 1 gives a brief overview of the Installation and Operation manual and other manuals you may need to install and operate the ADABAS SQL Server.

Chapter 2 explains the prerequisites, the installation procedure and points of special interest with regard to installing the base ADABAS SQL Server software on the Windows NT platform.

Chapter 3 explains how to install the ADABAS SQL Server system on the Windows NT platform, including generation of server environments and loading of demonstration data.

Chapter 4 explains how to generate user applications, start, operate and terminate the ADABAS SQL Server.

Chapter 5 explains how to invoke and work with the ADABAS SQL Server Utilities.

Chapter 6 describes the various types of logging facilities and how to activate them.

Appendix A is valid for ALL supported platforms. It explains the parameter processing language (PPL) which allows the configuration of the various components of the ADABAS SQL Server.

Who Should Read This Manual

This manual is for those who plan to perform the installation of the ADABAS SQL Server and for those who manage or maintain the ADABAS SQL Server (such as Database Administrators and System Programmers), and SQL application developers.

Other Helpful Manuals

Other manuals you may need are:

- ADABAS SQL Server Reference Manual
- ADABAS SQL Server Programmer’s Guide
- ADABAS SQL Server Messages and Codes Manual
- ADABAS for Windows NT Set of Documentation
- ENTIRE NET-WORK for Windows NT Set of Documentation

INSTALLING AND SETTING UP THE ADABAS SQL SERVER FOR WINDOWS NT

This chapter contains general information which applies when installing and setting up the ADABAS SQL Server for Windows NT.

The information contained in this chapter is independent of hardware type assuming that its CPU is Intel-compatible.

This chapter covers the following topics:

- Installation Package
- Writing Conventions
- General Installation and Setup Overview
- Performing General Installation and Setup

Product-specific installation, configuration and installation verification are described after this general chapter.

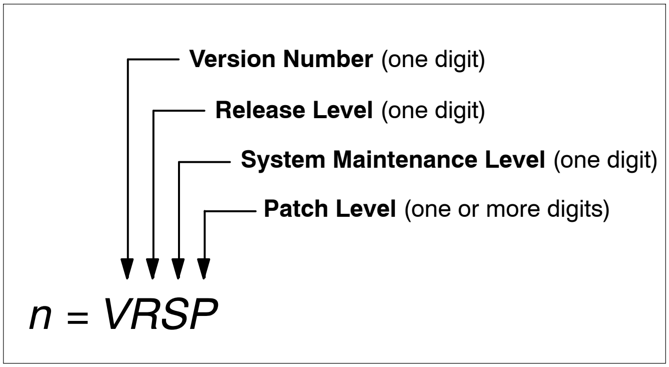
Installation Package

The installation package containing SOFTWARE AG products is available on ISO 9660 CD-ROM.

Writing Conventions

The following table describes the writing conventions used in this chapter.

Notation Example	Description
profile	Letters in **bold** indicate set strings which you cannot change, for example commands or certain file names.
`copy`	Letters in **`courier bold`** indicate that you must enter the information exactly as specified.
<*file-name*>	Lower-case letters in *italics* contained in angle brackets (< >) are used as placeholders to represent variable information which you must supply.
%*environment-variable-name*%	An environment variable name preceded and followed by a percent sign (%) stands for the string *contained* in the environment variable. For example, when the environment variable **ESQDIR** is set to C:\Program Files\Software AG\ADABAS SQL Server, **%ESQDIR%** stands for C:\Program Files\Software AG\ADABAS SQL Server.
vn	*vn* represents a product version number. *v* can be **v** for released versions, **b** for beta test versions and **r** for run-time versions. *n* consists of the following components:


Version Number (one digit)
Release Level (one digit)
System Maintenance Level (one digit)
Patch Level (one or more digits)
n = *VRSP*

General Installation and Setup Overview

The following is a summary of the steps required to set up the ADABAS SQL Server for Windows NT.

1. Check prerequisites.
2. Log in as an administrator.
3. Install the software.
4. Verify the directory structure and the Registry.
5. Install the ADABAS SQL Server Service (optional).
6. Install an ADABAS SQL Server System.

Performing General Installation and Setup

In this chapter the following is assumed:

- The root directory for SOFTWARE AG products is **C:\Program Files\Software AG**.

Step 1: Check Prerequisites.

To install and run the ADABAS SQL Server software you need:

1. An Intel or compatible PC running a Windows NT V4.0 (with Service Pack 3) or higher operating system.
2. ADABAS V3.1.1.11 or higher for Windows NT.
3. ENTIRE NET-WORK for Windows NT V2.3.1.2 or higher if client server communications are intended.
4. About 10MB of hard disk space for the ADABAS SQL Server software. Please be aware that additional hard disk space will be required by an ADABAS database in order to host at least one Server System catalog and error files. Please refer to the chapter **Installing The ADABAS SQL Server System** in this manual for further information.
5. A Windows NT user ID with administrator privileges.

Step 2:Log in as an Administrator

Log in using a user ID with administrator privileges.

Step 3: Install the Software.

Start the Windows NT Explorer and run the program setup.exe from the installation CD-ROM. Note that users who wish to control access to administrative functions should install the product on a disk containing an NTFS file system. This will allow use of the security tab in the file properties dialog to restrict access to executable files for specific users and/or groups.

Step 4:Verify the Directory Structure and the Windows NT Registry.

During the installation of the ADABAS SQL Server software, a new directory structure is created, and the Windows NT Registry is updated.

The following figure shows the general directory structure generated during installation and the environment variables which reference the specified directories.

```
\Program Files
    \Software AG
        \ADABAS SQL Server          %ESQDIR%
            \Vnnn                   %ESQDIR%\%ESQVERS%
                \bin
                    \bitmaps
                \demo
                \files
                \inc                %ESQDIR%\%ESQINC%
                \lib                %ESQDIR%\%ESQLIB%
\LocalNode
\LCLSRV_1
\LCLSRV_2
```

The variable %**ESQDIR%** is set to the path of the main directory of the ADABAS SQL Server, whose product code is ESQ. All files for this product are contained under the %**ESQDIR%** directory structure. The **LocalNode** path is created later when the first server is created.

Version-dependent components of the product, such as executables and libraries, are stored in the %**ESQVERS**% sub-directory. For example, all ADABAS SQL Server executables are found in the directory %**ESQDIR%\%ESQVERS%\bin**.

The following figure shows the general registry structure generated during software installation.

```
KEY HKEY_LOCAL_MACHINE
    ...
    KEY SOFTWARE
        KEY ...
        KEY Software AG
            KEY ...
            KEY ADABAS SQL Server
                KEY Servers
                KEY Vnnn
```

Step 5:Install the Server Service (Optional)

The ADABAS SQL Server Service “esqsvc” lets you run one or more ADABAS SQL servers so that it:

- is started automatically after Windows NT has been booted.
- remains active after you have logged off from Windows NT.

To install the ADABAS SQL Server Service, type **ESQSVC INSTALL** on a DOS Window command line.

Notes:

- You must have administrator rights in order to install the ADABAS SQL Server Service.
- In order to start an ADABAS SQL Server System, ENTIRE NET-WORK must be installed so that it’s CSCI component is available. Also, the ENTIRE NET-WORK user environment variables %SAGNW% and %SAGNODE% must be defined as system environment variables.

▸ **To start the ADABAS SQL Server Service**

□ Type **ESQSVC START** at a DOS Window command prompt.

Or double-click **Services** in the **Control Panel** and select ADABAS SQL Server Service in the Services list box and click the **Start** button.

▸ **To stop the ADABAS SQL Server Service and all ADABAS SQL Servers started by the service**

□ Type **ESQSVC STOP** at a DOS Window command prompt or

Double-click **Services** in the **Control Panel** and select ADABAS SQL Server Service in the Services list box and click the **Stop** button.

▸ **To remove (uninstall) the ADABAS SQL Server Service**

□ Type **ESQSVC REMOVE** at a DOS Window command prompt.

▸ **To set the startup mode of the ADABAS SQL Server Service**

□ Double-click **Services** in the **Control Panel** and select ADABAS SQL Server Service in the Services list box and click the **Startup** button. Select the desired **Startup Type** and click the **OK** button.

If you want an ADABAS SQL Server System to start automatically when the ADABAS SQL Server Service starts, use the ADABAS SQL Server **Administrator** to select a Server from the Server list, then click on the **Start as Service** option from the **Server** Menu. This information is then stored in the registry under the Servers key.

Once the **Start as Service** option has been selected for a particular ADABAS SQL Server System, the server will be started as a service each time the server is started via the ADABAS SQL Server **Administration Tool**.

Note:
*Starting an ADABAS SQL Server System via the **esqstart** command will NOT start the ADABAS SQL Server System as a service.*

You can stop an ADABAS SQL Server System that has been started as a service by using the **Stop** option from the **Server** Menu of the ADABAS SQL SERVER **Administrator**.

Step 6:Install an ADABAS SQL Server System

You have completed the installation steps for the ADABAS SQL Server software. Now proceed to the chapter **Installing The ADABAS SQL Server System** and install an ADABAS SQL Server System.

2

INSTALLING THE ADABAS SQL SERVER SYSTEM

This chapter describes preparing for and installing the ADABAS SQL Server System and Demonstration System.

Installation Activities Overview

Step 1 Check prerequisites

Step 2 Verify directory structure and registry structure

Step 3 Install the ADABAS SQL Server System

Step 4 Install the ADABAS SQL Server Demonstration System (optional)

Installation Procedure

Step 1: Check Prerequisites

Disk Space

In order to install an ADABAS SQL Server System using the default values, you will need approximately the following disk space:

- Windows NT disk space
  - A directory for the server
  - About 100KB for various miscellaneous files.
- ADABAS disk space:
  - First catalog file
    ASSO 1108 x 2K blocks
    DATA 100 x 4K blocks
  - Second catalog file
    ASSO 93 x 2K blocks
    DATA 100 x 4K blocks
  - Third catalog file
    ASSO 201 x 2K blocks
    DATA 100 x 4K blocks
  - Error file
    ASSO 33 x 2K blocks
    DATA 40 x 2K blocks

The three catalog files and the error files must be assigned file numbers in the range 1 – 255.

The three catalog files must be numbered sequentially, e.g. file numbers 4 – 6.

The error file may be shared between different ADABAS SQL Server Systems.

The catalog files cannot be shared between different ADABAS SQL Server Systems.

When sizing the ADABAS database using the catalog and error file sizes given above, you should allow for future growth requirements.

Other Software

In order to create or use an ADABAS SQL Server System in a Windows NT environment, ADABAS V3.1.1.11 or higher must be installed, and the ADABAS database chosen to host your Server System must be active and reachable.

In order to operate an ADABAS SQL Server System in a client/server mode, ENTIRE NET-WORK for Windows NT V2.3.1.2 or higher must be installed. If remote client/server communication connections are to be made, the network must be started and connected on both the server node and client node.

Note:
The LU parameters in ENTIRE NET-WORK should be increased to 64 KB.

In order to access an ADABAS SQL Server System via ODBC, ADABAS ODBC Client V2.1 or higher must be installed on the client node, and the ADABAS SQL Server System must be defined to the client using the ODBC Administrator utility.

Step 2: Verify Directory Structure and the Registry Structure.

The ADABAS SQL Server directory structure shown below was generated during the installation of the product (**Chapter 2, Step 3: Performing General Installation and Setup).** The **LCLSRV_x** subdirectories and all of the files contained are generated during the execution of **Step 3: Install the ADABAS SQL Server System** later in this chapter. The environment variables shown in the picture below are created automatically during the installation procedure.

```
\Program Files
    \Software AG
        \ADABAS SQL Server          %ESQDIR%
            /Vnnn                   %ESQDIR%\%ESQVERS%
                \bin
                    \bitmaps
                \demo
                \files
                \inc                %ESQINC%
                \lib                %ESQLIB%
            \LocalNode
                \LCLSRV_x
```

%ESQDIR%

This directory contains all currently installed versions of the ADABAS SQL Server product together with the environments for the individual local servers.

%ESQDIR%\%ESQVERS%

This directory is the top level for various components belonging to a particular ADABAS SQL SERVER software version.

%ESQDIR%\%ESQVERS\bin

This directory contains the executable images, dlls, batch procedures and bitmaps for the ADABAS SQL Server product.

%ESQDIR%\%ESQVERS\bin\bitmaps

This directory contains the bitmaps for the Basic Interactive Facility GUI Tool.

%ESQDIR%\%ESQVERS%\demo

This directory contains the scripts and source files needed to build the SAGTOURS demonstration system, which can be used for the installation verification of the ADABAS SQL Server.

%ESQDIR%\%ESQVERS%\files

This directory contains files which are required for installation or user template files.

%ESQDIR%\%ESQVERS\ (%ESQINC%)

This directory contains include files.

%ESQDIR\%ESQVERS\ (%ESQLIB%)

This directory contains libraries which are required for linking user programs using the ADABAS SQL Server software.

%ESQDIR%*LocalNode*

This directory contains the server directories for all local server systems generated on the current node.

%ESQDIR%*LocalNode\LCLSRV_x*

This directory contains files created during the generation of a local server LCLSRV_x.

The registry structure is shown below.

With the exception of the Sub Keys and Values shown under the Sub Key **Servers,** all other Sub Keys found below **SOFTWARE AG/ADABAS SQL Server** are established during the ADABAS SQL Server software installation as described in the previous chapter.

Note:
Do not modify the registry manually. This diagram is for reference purposes only.

```
KEY HKEY_LOCAL_MACHINE
    KEY SOFTWARE
        KEY ...
        KEY Software AG
            KEY ...
            KEY ADABAS SQL Server
                VALUE    ESQNODDIR
                VALUE    ESQSRVRT
                KEY Servers
                    KEY Server_x
                VALUE   ESQSRVDIR
                VALUE   ESQDBID
                VALUE   ESQCATF
                VALUE   ESQERRF
                VALUE   ESQDESC
                VALUE   ESQSEC
                VALUE   ESQSRVVERS
                VALUE   ESQSRVLOG
                VALUE   ESQPARMS
                VALUE   ESQSCHEMAID
                VALUE   START
KEY Vnnn
        VALUE    ESQBIN
        VALUE   ESQINIT
        VALUE   ESQLNKLIB
        VALUE   ESQSERVR
        VALUE   ESQTHS
KEY UserInfo
```

Step 3: Install the ADABAS SQL Server System

To run the ADABAS SQL Server, it is necessary to create a catalog and the SQL error message text file in an ADABAS database. This is done as follows:

1. If not already existent, create an ADABAS database using the ADABAS DBA Workbench clicking Database –> New... and then follow the dialog directions. Make sure that the sizes of the containers DATA, WORK and ASSO are altered to fit your needs. Please make sure that you have read **Step1: Check Prerequisites** before creating the new database.
2. Customize the files %ESQDIR%\%ESQVERS%\files\esqcat1.fdu, esqcat2.fdu, esqcat3.fdu and esqerr.fdu to meet your current ADABAS environment. The default values are as follows:

```
esqcat1.fdu       esqcat2.fdu       esqcat3.fdu       esqerr.fdu

dssize = 100b     dssize = 100b     dssize = 100b     dssize   =   40b
maxisn = 5000     maxisn = 1500     maxisn = 150      maxisn   =   1500
nisize = 1000b    nisize = 50b      nisize = 100b     nisize   =   20b
uisize =  100b    uisize = 40b      uisize = 100b     uisize   =   10b
```

Before loading the catalog and SQL error message text files, you should verify that there is sufficient free storage in the database. This can be done using the ADABAS DBA Workbench by selecting the host Database, then clicking Report–>General Information, then check Layout and click OK. Next, scroll to where Free space of Database is displayed.

Analyze the output and ensure that there is sufficient space in the ADABAS database. If there is not enough space, make a backup via ADABAS DBA Workbench selecting the database in question, then click Database–>Backup and follow the dialog instructions. When you are satisfied, select the database in question using the ADABAS DBA Workbench, then click Administrate–>Containers–>Addand follow the dialog instructions to add a new ASSO or DATA container.

3. Before starting the ADABAS database, ensure that the ADABAS nucleus parameters are set to at least the minimum values required by ADABAS SQL Server. Do this using the ADABAS DBA Workbench, selecting the database in question, then clicking Profile–>Nucleus Parameters and following the dialog instructions. Note that the database may have to be restarted before these new parameter values will take effect.

   The recommended minimum values are:

   ```
   lbp  =   1000000          lp   =   800              ls   =   70000
   lwp  =    220000          nh   =   4000         nisnhq   =   1000
   ```

   These values may have to be adapted according to the size of the container files for your ADABAS database.

   Note:
   All of the above parameters should be set by the DBA depending on the local requirements. For the installation verification of the ADABAS SQL Server we recommend these minimum values.

4. Start the ADABAS database from the ADABAS DBA Workbench by selecting the database and then clicking Database–>Start.

5. Generate one or more server environments within ADABAS SQL Server using either the ADABAS SQL Server **ADMINISTRATOR** or its command line interface **esqgen**. The ADABAS SQL Server **ADMINISTRATOR** is the preferred way of generating and operating a server on this platform. It has more features and it is more powerful than the command line interface. The command line interface has only been offered due to compatibility with other platforms, it works in a command prompt DOS window.

   In order to use the ADABAS SQL SERVER ADMINISTRATOR GUI to generate a server, do the following:

- Start the ADABAS SQL SERVER **ADMINISTRATOR**.
- Click Server–>New and follow the Server Generation Wizard (steps 1 to 4) that are presented by this tool. Please note that you can go forward and backwards between Wizard steps until satisfied and then click OK. If an error occurs, the Wizard will keep your settings until you generate a server successfully or cancel the operation.
- On step 1 enter the Server Name and a brief description.
- On step 2 change the database ID, catalog file numbers and error file number. You can also specify a default schema id to be stored in the client parameter file esqrun.par. Please refer to APPENDIX A in this manual for more information about the parameter processing language (PPL).

- On Step 3 various server options may be set:
  - Checking the Security Server box will create a secured server. Please note that it is not possible to convert a server to/from being secured once it has been generated. For details, please refer to the following section **To be Considered when Deciding for/against a Server with Security Features** for a brief overview or to the *ADABAS SQL Server Programmer's Guide,* chapter **The ADABAS SQL Server Security Concept.**
  - Checking the Start as a Service box will mark this server to be started as a Windows NT Service. Please note that the ADABAS SQL Server Service must first be installed and started before a server can be started as a service. For more information, please refer to the section **Install the ADABAS SQL Server Service** in Chapter 2 of this manual. This option may also be actived later via the menu Server –>Start as a Service.
  - Checking the Make Node Default box will mark this server as the system-wide default server for this Windows NT environment. Users communicate with the default server when issuing an implicit CONNECT (i.e. a server is not explicitly entered in the CONNECT). This option can also be set via the menu Server–>Make Node Default. Please note that this option requires administrator privilege because the system environment table will be updated. For more information about **Make a Server as this Node Default**, please check the ADABAS SQL ADMINISTRATOR online help.
  - Checking the Load SAGTOURS Demo file box will cause the SAGTOURS demonstration system to be loaded in this catalog. If this box is checked, you will be given the opportunity to change the default ADABAS file numbers assigned for the demo files. Please see the information about Loading the Demo via command interface line commands later on this chapter at **Install the ADABAS SQL Server Demonstration System via Command Line Interface.**
  - Checking the Load SAGTOURS Demo file box also enables you to then check the Load Booking CGI Demo File check box. The table BOOKING is needed only if the demonstration package for the ADABAS SQL Server / WWW Common Gateway Interface will be installed. Please note that installation of the demonstration system requires the Microsoft C Compiler.
- Step 4 displays a summary of options chosen. If correct, click on the Finish button to generate the server. If incorrect, click on the Back button to make corrections.

Using the ADABAS SQL SERVER Command Line Interface to generate a new server:

A server can be generated by starting a command prompt DOS window and then entering:

```
esqgen   servername dbid catalog_base_file error_file [desc] [sec]
```

The position of the individual keywords in the syntax string must be as shown above. The expected values for the specification of the above parameters are:

servername is the name of the server to be generated with a maximum length of 8 uppercase characters. (Server names are case sensitive.)

dbid is the ADABAS database ID for the catalog and error files,

catalog_base_file is the ADABAS base-file number for the ADABAS SQL Server catalog for the specified server. There must be three free consecutive files for the catalog.

error_file is the ADABAS file number for the ADABAS SQL Server error texts which may also be shared by several servers.

[*desc*] is an optional short description of the server with a maximum length of 35 characters. A description must be entered, whereby an empty string qualifies, when the keyword “sec” is specified. If the description contains an empty string/blanks or lowercase characters, it must be enclosed in double quotes.

[sec] is an optional keyword for a server to be generated with security features. If omitted, a server specification without security features is assumed. **Please be advised that it is not possible to convert a server to/from being secured once it has been generated.** For details, please refer to the following section **To be Considered when Deciding for/against a Server with Security Features** for a brief overview or to the *ADABAS SQL Server Programmer’s Guide,* chapter **The ADABAS SQL Server Security Concept.**

The options “Make Node Default” and “Start as Service” are not available via the command line interface but may be set at a later time via the Administration GUI tool.

The following will have happened once a server has been successfully generated via the administration GUI tool or esqgen:

- A new catalog (three ADABAS files containing SQL information needed to retrieve data from the catalog) and an SQL error message text file (one ADABAS file containing SQL error message text records) will be loaded in the ADABAS database.
- A server-specific Windows NT directory below %ESQDIR%\LocalNode, (where LocalNode is the name of your current node) will be generated to contain server-specific parameter template files, log files, trace files, etc.
- The user DBA will be created without a password.
- The Windows NT Registry will be updated with a new server subkey below the key **HKEY_LOCAL_MACHINE\SOFTWARE\SOFTWARE AG\ADABAS SQL Server\Servers.**

To be Considered when Deciding for/against a Server with Security Features

The following three points must be regarded in ***either version***, security or non-security:

- the USER must still be defined via the CREATE USER statement.
- the user DBA is the only one authorized to create a schema. The owner of a schema is the only one authorized to create tables, views, etc.
- full password support is included, the CREATE/DROP/ALTER USER statements can be executed as well as the CONNECT statement with user specification.

The following points must be regarded if an ADABAS SQL Server is generated ***without*** the security features:

- at runtime, no security check is performed. Access rights to a specified table, etc. will not be checked. For this reason, the performance of the server will be more favorable than in a security version.
- the execution of GRANT/REVOKE statements is not possible.
- those tables, established to hold privilege-related data in the catalog will be created but are empty (for example, the table: table_privileges).

The following point must be regarded if an ADABAS SQL Server is generated ***with*** the security features:

- The user DBA is treated like any other user. A SELECT statement against any of the INFORMATION_SCHEMA tables will return results for those objects only which have been GRANTed access to. To see all information, the DBA should use the DBA_SCHEMA tables (which are only accessible to the DBA). For further details, see **Appendix C — The ADABAS SQL Server Catalog Structure** in the *Programmer's Guide*.

Verification

To verify that the catalog and the SQL error message text file for the generated server have been loaded correctly, use the ADABAS DBA Workbench. Select the subject Database, and then click Report−>File Status. Next, scroll the report views and make sure that the following files have been loaded.
ESQCAT1_*servername*
ESQCAT2_*servername*
ESQCAT3_*servername*
ESQERR_*servername*

If, for some reason (e.g., not enough space in the database or file number not available) a new server generation aborts or concludes unsuccessfully, an attempt will be made to free up all resources allocated to it.

However, if the automatic clean up should fail, use esqremove to free up the resources as follows:

- Start a command prompt DOS window and enter

```
esqremove servername -a
```

- Enter the following if you only want to remove the server directory:

```
esqremove servername
```

Once the problem has been resolved, retry the server generation using the ADABAS SQL SERVER ADMINISTRATOR GUI or the command line interface esqgen.exe as described above.

6. Before customizing some of the parameter files, set the server to be your current default server. This can be accomplished in one of two ways:

- **Using the ADABAS SQL Server ESQSET GUI tool:**
  - Start the ADABAS SQL SERVER ESQSetGUI tool “ESQSetGUI”.
  - Using the mouse, right click the subject server, and then click on the Context Menu Set Server as Default.
- **Using the command line interface:**
  - Start a command prompt DOS window, then type:

```
esqset.bat servername
```

In the %ESQDIR%/LocalNode/*servername* directory are three parameter files which are preset with default values, but may be customized to your needs, if applicable. The customization may be done via the ADABAS SQL SERVER ADMINISTRATOR GUI by right clicking the subject server, then clicking on Server Parameters, Client Parameters, or Precompiler Parameters. A notepad session will be started for editing the files.

- **esqsrv.par** **(server parameter file)**
  This parameter file is used when starting a server process (for client/server processing).
- **esqrun.par** **(client/runtime parameter file)**
  This parameter file is used when starting an application using an ADABAS SQL Server.
- **esqpc.par** **(precompiler parameter file)**
  This parameter file is used when starting the precompiler.

For details on how to modify the above parameters refer to chapter: **Operating the ADABAS SQL Server,** section: **Parameter Processing** and to **Appendix A — The Parameter Processing Language (PPL)** in this manual.

7. Assigning a password to the user DBA

During the installation of the server, a DBA user was created without a password. If the server was created with the security feature turned on, you may wish to assign a password to the DBA account at this time. This can be done in two ways.

- **Via the BASIC Interative Facilities GUI tool**
  - Start the BASIC **Interative Facilities “basic”**
  - Select a server and its communication type.
  - Run the following SQL statement:

```
alter user dba set password ”<new password>”;
```

- **Via the command line interface esqint**
  - Start a command prompt DOS window, then type:

```
esgint

ESQ User: dba
Password: (carriage return)

esqint: alter user dba set password “<new password>”;
```

The password will have to be dropped again if you later wish to install the demonstration system (SAGTOURS) as described below.

8. The ADABAS SQL Server System installation is now complete.

Step 4: Install the ADABAS SQL Server Demonstration System (SAGTOURS)

Note:
Installation of the demonstration system requires the Microsoft C compiler.

The ADABAS SQL Server System Demonstration allows you to establish an environment from which the ADABAS SQL Server installation can be tested. Before this demonstration system can be installed, a server environment, for example a server named SAGTOURS, must have been generated via the ADABAS SQL SERVER ADMINISTRATOR GUI tool or the Command Line Interface esqgen.exe.

The SAGTOURS system consists of five SQL tables CONTRACT, CRUISE, PERSON, SAILOR, YACHT and, optionally, BOOKING contained as files in your ADABAS database. The table BOOKING is needed only if the demonstration package for the ADABAS SQL SERVER / WWW Common Gateway Interface will be installed.

Example:
Create the SAGTOURS Demonstration System tables by typing the following after starting a command prompt DOS window: (please note: Creating the SAGTOURS Demonstration System is not required at this point if it was already loaded during server creation via the Administration GUI Tool in Step 3 above.)

```
cd  %ESQDIR%\%ESQVERS%\demo
esqset.bat SAGTOURS
crttabs 4 5 6 7 8 9
```

The CREATE TABLE command line interface (crttabs.exe) creates the ADABAS file CONTRACT with file number 4, CRUISE with file number 5, PERSON with file number 6, SAILOR with file number 7, YACHT with file number 8, and BOOKING with file number 9. In addition, the schema SAGTOURS has been created with the owner “esq”. This schema contains the tables established as a result of running the crttabs.exe.

Note:
The tables are established with the schema identifier SAGTOURS.

Any warnings that may be produced by the Precompiler should be ignored as they are of informational value only.

The ADABAS DBA Workbench may be used to verify that the ADABAS files containing the SAGTOURS tables have been successfully created.

If, for any reason, the creation of the SAGTOURS tables aborted, or was completed unsuccessfully, for example, because the database is too small, the SAGTOURS system can be cleared by starting a command prompt DOS window and then typing:

```
cd %ESQDIR%\%ESQVERS%\demo
drptabs
```

which drops all tables already created. Once the cause of the failure has been located and repaired the SAGTOURS system can be restarted using the crttabs.exe, as described above.

To perform a single row SELECT on the SAGTOURS tables, start a command prompt DOS window and then type the following:

```
seltabs
```

To perform a DECLARE CURSOR operation on the SAGTOURS tables, start a command prompt DOS window and then type the following:

```
crstabs
```

To drop the SAGTOURS tables, start a command prompt DOS window and then type the following:

```
drptabs           (for all tables in succession)
```

Verify that the tables have been dropped by using the ADABAS DBA Workbench to see that the ADABAS files no longer exist.

The SAGTOURS tables can also be used to verify the installation of the ADABAS SQL Utilities. The SAGTOURS tables should not be dropped beforehand.

To start the command line BASIC Interactive Facility in a command prompt DOS window, type the following:

```
esqint
```

The system prompt requests user ID and password. In the case where the security feature is not turned on, press ENTER to bypass this input. If the security feature is turned on enter “esq” for the user and a RETURN for the password.

```
ESQ User: esq     (string 'esq' should be entered by the user id)
Password:         (press carriage return)
```

To display all tables owned by the current user, type the following:

```
esqint: select *  from information_schema.tables;
```

To display all columns of table SAILOR, type the following:

```
esqint: select *  from information_schema.columns
                  where table_schema = ”SAGTOURS” and table_name = ”SAILOR”;
```

Installation File Lists

%ESQDIR%\%ESQVERS%\bin:

bitmaps	A directory with the BASIC Interactive Facility GUI tool bit maps
basic.exe	BASIC Interactive GUI Facility
cscmem.bat	Batch file to establish CSCI environment
ESQAdmin.exe	ADABAS SQL SERVER ADMINISTRATOR GUI
ESQBIF.dll	Various functions
esqc.exe	“C” Precompiler Component
esqerl.exe	Precompiler Error Logger Utility
esqgen.exe	Server Generation, Command Line Interface
esqgtd.exe	Generate Table Description, Command Line Interface
esqini.exe	Server Initialization Image, Command Line Interface
esqint.exe	Basic Interactive Facility, Command Line Interface
esqkill.exe	Server Kill, Command Line Interface
Esqlnk32.dll	Various functions
esqopr.exe	Operator Utility, Command Line Interface
esqpc.exe	Precompiler, Command Line Interface
esqremove.exe	Server Removal, Command Line Interface
esqset.bat	Set Server as User Default, Command Line Interface
esqset.exe	Set Server as User Default to be used ONLY by esqset.bat
ESQSetGUI..exe	Show Servers and Set Server as User Default GUI
esqshow.exe	Show Servers, Command Line Interface
esqsrv.exe	Server Thread Executable
esqsrvmg.dll	Various Functions
esqstart.exe	Server Starting, Command Line Interface
esqstop.exe	Server Stopping, Command Line Interface
esqsvc.exe	Server Service
esqths.dll	Various Functions
Esqud.dll	Various Functions
sagvo.dll	Various Functions

%ESQDIR%\%ESQVERS%\files:

esqcat1.fdu	Input Files for ADABAS Utilities to load
esqcat2.fdu	Input Files for ADABAS Utilities to load
esqcat3.fdu	Input Files for ADABAS Utilities to load

etshte.dat	ADABAS SQL Server error file data
esqerr.fdu	ADABAS SQL Server error file data
esqcat.sql	ADABAS SQL Server catalog data
esq_routing.dat	Server Routing File (Template)
esqpc.par	Precompiler Parameter File (Template)
esqrun.par	Run-time (Client) Parameter File (Template)
esqsrv.par	Server Parameter File (Template)
esquex.def	A dll definition file for User Exits
makefile	Template makefile for user exits
README.1ST	Information that is not present in the manuals

%ESQDIR%\%ESQVERS%\lib:

Esqlnk32.lib	To be linked with User programs in order to map esqlnk32.dll

%ESQLIB%\%ESQVERS%\inc:

esqca.h	Files used for building user exits
esqacc.h	Files used for building user exits
esqda.h	Files used for building user exits
esqerr.h	Files used for building user exits
esqinf.h	Files used for building user exits
esquex1.h	Files used for building user exits
esquex2.h	Files used for building user exits

%ESQDIR%\%ESQVERS%\demo:

Executables for Demo Application

crstabs.exe crttabs.exe drptabs.exe seltabs.exe

Source files for Demo Application

cont_ld.cpc crs_cont.cpc crs_crui.cpc crs_pers.cpc crs_slr.cpccrs_yht.cpc
crui_ld.cpc pers_ld.cpc sel_cont.cpc sel_crui.cpc sel_pers.cpc sel_slr.cpc
sel_yht.cpc slr_ld.cpc yacht_cr.cpcs yacht_dr.cpc yht_ld.cpc

Make files for Demo Application

cont_ld.mak crs_cont.mak crs_crui.mak crs_pers.mak crs_slr.mak crs_yht.mak
crui_ld.mak pers_ld.mak sel_cont.mak sel_crui.mak sel_pers.mak sel_slr.mak
sel_yht.mak slr_ld.mak yacht_cr.mak yacht_dr.mak yht_ld.mak

Source file for BASIC Interactive SQL MU/PE sample table

city.sql

Data files for demo tables

contract.data cruise.data person.data sailor.data yacht.data

OPERATING THE ADABAS SQL SERVER

This section contains information pertaining to the operation of the ADABAS SQL Server, the ADABAS SQL Utilities and user programs which communicate with the ADABAS DBMS using ANSI/ISO SQL (Structured Query Language) in a Windows NT environment.

Overview of ADABAS SQl Server Commands

This is a brief overview of the ADABAS SQL Server commands. A detailed description of each command can be found later in this chapter. These commands can be run from a command prompt or in most cases from GUI tools that provide equivalent functionality. This overview describes the command version syntax and where available the GUI version menu navigation path.

Generate one or more Server Environments

Generate server environments:

```
C:\> esqgen   servername dbid catalog_base_file error_file [desc] [sec]
```

```
ADABAS SQL Server Administrator->Server->New
```

Starting the Utilities:

Start BASIC Interactive SQL:

```
C:\> esqint [servername [communication [destination]]]
```

```
ADABAS SQL Server Interactive
```

Start the Generate Table Description Utility:

Command line operation mode:

```
C:\> esqgtd  [option]  database_name  database_number  file_number  [ddm_name]
```

Input file operation mode:

```
C:\> esqgtd < input_file
```

Creating an Embedded SQL Application

Precompile a module:

```
C:\> esqpc    module [+L] [-w] [+K]
```

Driving an ADABAS SQL Server:

Set a default server:

```
C:\> esqset   servername
```

```
ADABAS SQL SERVER Set Server->Server->Set Server As Default
```

Verify the default server / all known servers:

```
C:\> esqshow  [-a]
```

```
ADABAS SQL SERVER Set Server
```

Start a server:

```
C:\> esqstart [servername]
```

```
ADABAS SQL Server Administrator->Server->Start
```

Terminate a server

```
C:\> esqopr [servername]
esqopr: shutdown
```

```
ADABAS SQL Server Administrator->Server->Stop->Shutdown
```

```
C:\> esqstop  [servername] [-a]
```

```
C:\> esqopr [servername]
esqopr: abort
```

```
ADABAS SQL Server Administrator->Server->Stop->Abort
```

```
C:\> esqkill  servername
```

```
ADABAS SQL Server Administrator->Server->Stop->Kill
```

Remove a server environment:

```
C:\> esqremove servername [-a]
```

```
ADABAS SQL Server Administrator->Server->Delete
```

Creating an SQL Application Program

An application program is built for execution in a 3-step procedure (precompile, compile and link).

Precompile

The ADABAS SQL Server Precompiler supports the C host language. The precompiler is found in the directory defined by the registry value ESQBIN, which was created during the installation of the ADABAS SQL Server. This directory is also included in the user's PATH.

If the application program design and coding are complete, the source statements are ready to be prepared for execution. Before a program can be executed, it must be compiled by an appropriate host language compiler. Before compilation, however, the SQL statements embedded in the host language must first be prepared by the ADABAS SQL Server Precompiler for compilation as host language statements.

The ADABAS SQL Server Precompiler scans each statement of the program source and produces a modified program in which every SQL statement has been replaced by host language statements, such as variable definitions and calls to the ADABAS SQL Server. Ensure that the ADABAS nucleus containing the catalog and error file is active.

Before starting the precompiler the default server has to be set. This can be done via the GUI application ADABAS SQL SERVER Set Server or by entering the following:

```
C:\> esqset  servername
```

The precompiler is started by entering the following command from a command prompt:

```
C:\> esqpc  modulename  [+L][-w] [+K]
```

The optional parameter +L generates a precompiler listing file. The optional parameter −w suppresses precompiler warnings. The optional parameter +K results in the output file ESQPCOUT being kept inspite of errors.

The precompiler searches in the current working directory for a precompiler source file according to the file extension convention shown in the following table, and then starts a language-dependent precompilation.

The following table shows the file extension convention for precompiler source files:

Programming Language	**Precompiler Input File**	**Precompiler Output File**	**Precompiler Listing**
C	*modulename*.**cpc** or *modulename*.**pc**	*modulename*.**c**	*modulename*.**pclis**

If an explicit schema identifier was selected during server generation, that value will be hard coded in the parameter file. If the value for the schema identifier is not hard-coded in the parameter file, the environment variable ESQSCHEMAID, if set, determines the default SQL schema identifier for the application module. If there is no value in the parameter file and the environment variable ESQSCHEMAID is not set, the user's login ID will be used as the default table qualifier.

If the value for the library name is not hard-coded in the parameter file, the environment variable ESQLIBRARY, if set, determines the library name under which the application module's meta programs are stored in the catalog at runtime. The default value for ESQLIBRARY is the user ID. Setting this variable allows distinction between different sets of applications.

By default, the precompiler uses the server-specific parameter file esqpc.par located in the server directory ESQSRVDIR as defined in the registry during server generation. If entry changes in the parameter file are necessary, there are two recommended procedures:

For permanent user- and server-specific changes enter go to the server directory specified by ESQSRVDIR and copy the file `esqpc.par` to file `esqpc.par.<user ID>`. Then use Notepad or another editor of your choice to make the necessary changes.

For temporary working-directory-specific changes copy the file esqpc.par from the server directory to the desired working directory and use an editor to make the necessary changes.

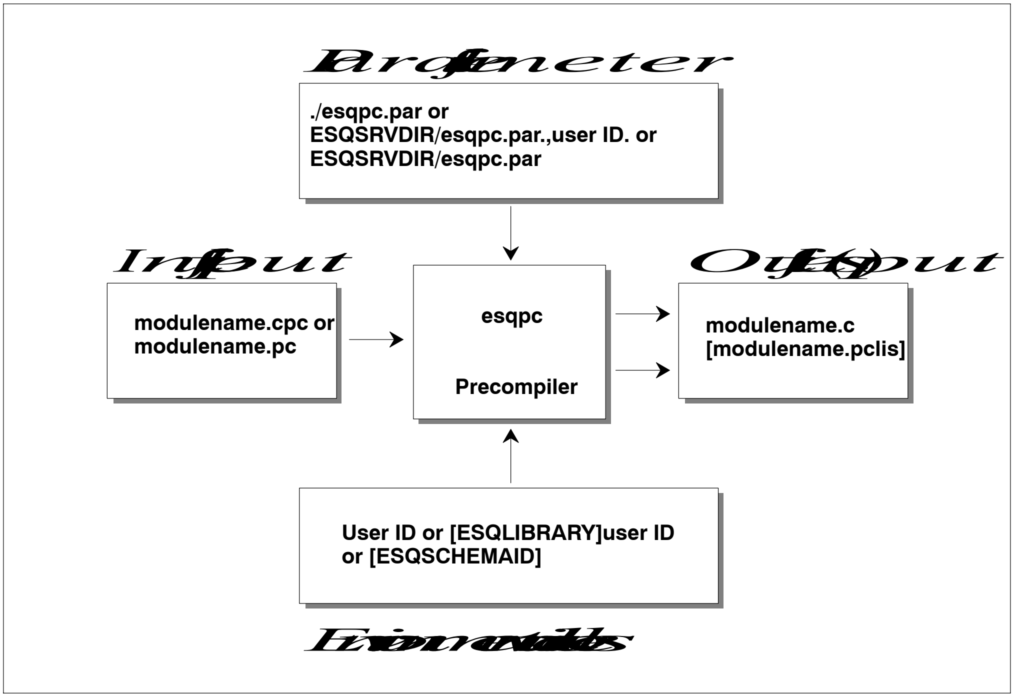


Figure 3-1: Precompilation data flow chart for a C-application

Precompiler Integration With Microsoft Developer Studio

Users of the Microsoft Developer Studio can add sources containing embedded SQL directly into their projects and workspaces. Invocation of the ADABAS SQL Server precompiler can be automated using the Custom Build feature in Developer Studio. The steps required to accomplish this are:

1. Before you start the Microsoft Developer Studio you must define the server that you wish to precompile your server against. This can be done system-wide via the ADABAS SQL Server Administration tool or for this user session via the ESQSetGUI tool.

2. Add the directories containing the precompiler and the ADABAS link routine to the Developer Studio search path. This is necessary because Developer Studio does not use the system path when searching for executables.

From the Tools menu select Options. Select tab Directories. Pull down the Show directories drop down list box and select Executable files. In the list box Directories double click on an open line and type the full path to your ESQBIN (e.g. g:\Program Files\Software AG\ADABAS SQL Server\v142\bin). Double click on another open line and type the full path to ADABIN (e.g. g:\Program Files\Software AG\ADABAS C\v310). Click on OK.

3. Create a new project and workspace as normal in Developer Studio.

4. Add the library for the ADABAS SQL Server link routine to your project.

   From the Project menu select Add To Project−>Files. Navigate to your ESQLIB directory (e.g. g:\Program Files\Software AG\ADABAS SQL Server\v142\lib) and select the file ESQLNK32.lib. This library file is the loader for ESQLNK32.dll.

5. Add or create the files containing your embedded SQL to your project.

   From the Project menu select Add To Project−>New (new sources) or Add To Project−>Files (existing sources) to create/add the *.cpc, *.pc or *pcc source files containing your embedded SQL to your project.

   Note:
   Before adding existing embedded SQL programs to a project, you must copy them to the project directory so that the ADABAS SQL precompiler (esqpc.exe) will find them.

6. Add a custom build rule to run the precompiler on your *.pc sources.

   From the Project menu select Settings. Pull down the Settings For drop down list box and select All Configurations. Expand the tree for your project to display all sources. Select all *cpc, *.pc or *.pcc sources. Select tab Custom Build. Double click on an open line in the Build commands(s) window and type in the command:

   ```
   esqpc.exe $(InputName)
   ```

   Double click on an open line in the Output file(s) window and type in the file name:

   ```
   $(InputName).c
   ```

   Click OK.

7. Build the project. The output window will show the output of all precompiles and will show a link error because there is no main fuction.

8. Add all the now generated *.c files to your project.

   From the Project menu select Add To Project−>Files and select all the *.c files that correspond to your original *.cpc, *.pc or *.pcc files.

9. Build the project again. The output window will show compiler output and a successful link.

   Once this procedure has been performed for a project you can freely modify any *.pc source at a later time and simply build the project. Developer Studio will detect that the *.pc file is modified and will run the custom build step for that file. This will execute the precompiler and cause the *.c file to be re-generated. Developer Studio will then detect that the *.c file has been modified and will compile it and link the application.

   You can also force the precompile to be performed on all *.pc sources by doing a Rebuild All from the Build menu or by doing a Clean followed by a Build.

Compile

After precompilation the application program is compiled using the standard compilation procedure suitable for the host language.

The procedure for invoking the C compiler may vary depending on what compiler you are using. In all cases all that is necessary is a standard compile using ANSI conformance.

For users of Microsoft's C compiler as included in Developer's Studio, the command is:

```
C:\> cl -c <program_name>.c
```

This will produce an intermediate output file `<program_name>.obj` that can be used in a later link.

It is also possible to compile and link in one step. All that is required in this case is that you add the ADABAS SQL Server link library pointed to by environment variable ESQLNKLIB to your module list. The command is:

```
C:\> cl <program_name>.c %ESQLNKLIB%
```

This will produce an executable file named `<program_name>.exe`.

Link

Linking is performed after compilation. At link-time the SQL library has to be linked to the application program previously produced by the host language compilation procedure. The library is stored in the directory ESQLIB which was also created during the initial installation.

The procedure for invoking the linker may vary depending on what compiler you are using. In all cases all that is necessary is a standard link that includes the ADABAS SQL Server link library pointed to by environment variable ESQLNKLIB in the module list.

For users of Microsoft's linker as included in Developer's Studio, the command is:

```
C:\> link <program_name>.obj %ESQLNKLIB%
```

This will produce an executable file named `<program_name>.exe.`

It is also possible to compile and link in one step. All that is required in this case is that you add the ADABAS SQL Server link library pointed to by environment variable ESQLNKLIB to your module list. The command is:

```
C:\> cl <program_name>.c %ESQLNKLIB%
```

This will produce an executable file named `<program_name>.exe`.

Execute

Before running an ADABAS SQL Server application the default server must be set by entering the following command:

```
C:\> esqset  servername
```

or by executing the GUI application ADABAS SQL SERVER Set Server. Both of these methods will set the environment variable ESQSRV in the user's environment variable table. This value is persistent across logins so once it is set the user does not need to set it again unless the value must be changed.

Once the default server is set simply execute the application normally.

A client parameter file is always used during application execution By default the runtime system uses the server specific parameter file esqrun.par from the server directory ESQSRVDIR as specified in the registry during server generation. If any changes in the parameter file are necessary, there are two recommended procedures:

1. For permanent user- and server-specific changes go to the directory pointed to by ESQSRVDIR and copy the file esqrun.par to file esqrun.par.<user ID>. Then use Notepad or another editor of your choice to make the desired changes.

2. For temporary working-directory-specific copy the file esqrun.par from the ESQSRVDIR directory to the current directory and use an editor to make the desired changes.

Driving an ADABAS SQL Server

The ADABAS SQL Server provides two methods to operate and maintain servers. The first is a GUI application named ADABAS SQL Server Administrator.This GUI administrator includes a full set of online help and classic user interface features that should be comfortable for Windows users. The second is a set of commands that provide the same administrative functionality for users accustomed to a character based interface. This chapter will describe in detail the command syntax and where appropriate will show the menu path to achieve equivalent functionality using ADABAS SQL Server Administrator.

A server environment is generated via the "esqgen" command as described in the chapter **Installing the ADABAS SQL Server System**. A server can also be generated using ADABAS SQL Server Administrator–>File–>New. The server environment is mainly characterized by the ADABAS files which hold the catalog for that server. The server itself, however, is referenced by its name only.

Set the Default Server

The command that sets the current default server within a user session is:

```
C:\> esqset  servername
```

Any following statement (including precompiler and SQL application execution) will take this server as default server, if no other server name is specified.

Equivalent functionality can be found in the GUI application ADABAS SQL SERVER Set Server–>Server–>Set Server as Default. This tool is intended for use by all users to set their personal default server.

Additionally an adminstrator can set the system default server for a given node (machine) via:

```
ADABAS SQL Server Administrator->Server->Make Node Default
```

Verify the Default Server

The command to verify the current default server settings is:

```
C:\> esqshow [-a]
```

The −a option shows also all known ADABAS SQL Servers on the current node.

Both GUI applications ADABAS SQL SERVER Set Server and ADABAS SQL Server Administrator have a main window that already displays a list of all servers known on this machine and an indication of which is the default or system default.

Start a Server

A server process has to be started, if you want to perform either local (client and server on same node) or remote (client and server on different nodes) client/server computing.

Before starting a server the client/server communication interface (CSCI) has to be activated. Whether you start your server from the command line or from the GUI administrator, this activation will be performed for you automatically so you need not worry about as is necessary on other platforms like UNIX.

For more information on CSCI refer to the appropriate SOFTWARE AG manuals.

For remote client/server computing NET-WORK FOR Windows NT must also be active and parameterized properly on both sides. Refer to the chapter **Client/Server Topics** in the *ADABAS SQL Server Programmer's Guide*.

To start the ADABAS SQL Server use the following command:

```
% esqstart [servername]
```

Equivalent functionality can be found in:

ADABAS SQL Server Administrator−>Server−>Start

The server processes are started in background using the parameter file ESQSRVDIR/esqsrv.par, which among others defines the number of thread processes to be started for this server.

Note:
One server thread can handle only one client session at a time, so the number of threads determines the maximum number of concurrent clients sessions at a given time.

Terminate an ADABAS SQL Server

There are three ways to terminate an ADABAS SQL Server from the command line ranging from a normal shutdown to a fast kill. Equivalent GUI functionality can be found in:

ADABAS SQL Server Administrator–>Server–>Stop–>Shutdown
ADABAS SQL Server Administrator–>Server–>Stop–>Abort
ADABAS SQL Server Administrator–>Server–>Stop–>Kill

- **SHUTDOWN**

  The normal server shutdown is done via the operator command (esqopr) together with the SHUTDOWN command:

  ```
  C:\> esqopr [servername]
  esqopr: shutdown
  ```

  An equivalent command, whereby the user is not prompted for confirmation and shutdown reasons is the ESQSTOP command:

  ```
  C:\> esqstop [servername] [-a]
  ```

  After execution of these commands any new client sessions will be rejected by the server. All server threads that currently have no active client session will shut down within one minute. If the –a option is specified, an abort is performed with the consequences explained below. The other server threads remain active until the clients disconnect from the server or the session being timed-out by the server. Only if there are no more active sessions will the server shut down completely.

- **ABORT**

  The next level of stopping a server is done via the operator command with the ABORT command.

  ```
  C:\> esqopr [servername]
  esqopr: abort
  ```

  After execution of this command any new client sessions and requests will be rejected by the server. All server threads continue with their current client requests, if any are active, and will shutdown within one minute regardless of any active client sessions, i.e. the sessions are terminated.

- **KILL**

  This command terminates the server at once, regardless of any active client sessions or requests. Nevertheless, a proper clean-up will be performed.

  ```
  C:\> esqkill servername
  ```

Display the Server Log File

There is no command to display a server log file on Windows NT because this information is easily available using GUI tools. Any text editor can be used to display this file because it is a regular text file. The location of the server log for a given server is defined in the registry but always defaults to ESQSRVDIR\esqsrv.log.

An even easier method to view the log file without knowing its location is via:

ADABAS SQL Server Administrator−>Report−>Log File

This GUI tool will look up the correct location of the log file in the registry and will open a scrollable child window to display the file content. This view can be refreshed at any time by pressing F5.

Display Error Text

There is no command to display a error texts on Windows NT because this information is easily available using GUI tools. Any text editor can be used to display this raw data because it is a regular text file. The location of the raw error text data for any version can be found in the location:

files\etshte.dat

An even easier method to view the log file without knowing its location is via:

ADABAS SQL Server Administrator−>Help−>Error Texts

This GUI tool will access a standard Windows NT help file that contains all error texts.

Remove an ADABAS SQL Server Environment

The esqremove command removes an existing server environment which was generated using the esqgen command. Default functionality removes the server-specific files in the OS file system. If the option −a is specified, additionally, the catalog and error files in ADABAS will be removed.

```
C:\> esqremove servername [−a]
```

Equivalent GUI functionality can be found via:

ADABAS SQL Server Administrator−>Server−>Delete

A dialog will be presented to provide the option of removing ADABAS catalog and error files.

The ADABAS SQL Server Operator Utility

The ADABAS SQL Server Operator Utility provides the functionality to obtain actual and statistical information about a specific ADABAS SQL Server running on the local node, and commands to terminate a server. All functionality available in this command line tool is also available in the GUI ADABAS SQL Server Administrator.

Start the Utility

The utility is invoked by typing:

```
C:\> esqopr [servername]
```

The following sample sessions shows the usage and functionality of the ADABAS SQL Server Operator Utility. The commands entered by the user are printed in bold.

Example:

```
C:\> esqopr
%ESQOPR-I-STARTED, 15-APR-1998 16:48:26, Version 1.4.2.13 (INTEL-80386/WINDOWS
NT)
%ESQOPR-I-ONLINE,  Server ESQNIST attached online on node GENESIS
esqopr: help

ESQ operator utility online help display

 help      - Online help display (this display)
 dis=act   - Display active servers on local node
 dis=scb   - Display Server Control Block (SCB)
 rep=n     - Repeat display every n seconds (VTxxx only)
 server=SS - Attach to server named SS
 .         - Redisplay most recent display
 shutdown  - Shutdown server (wait for disconnects)
 abort     - Abort server    (disconnect sessions)
 quit      - Leave esqopr

esqopr: dis=scb

ESQ Server Control Block (SCB) Information

 Server name        = ESQNIST on node GENESIS
 Startup time       = 15-APR-1998 16:47:50
 Version            = 1.4.2.13
 Server type        = CSCI
 Catalog Files      = DB 214, Files 4-6
 Thread processes   = 5
 Sessions / Thread  = 1
 Last active thread = none
```

```
Server Users:          Client/Server    Linked-in  Precompiler        Total

 Active sessions    =              0            0            0            0
 Active requests    =              0            0            0            0
 Active threads     =              5            0            0            5
 Total sessions     =              0            0            0            0
 Total requests     =              0            0            0            0

15-APR 16:48:18.66 Server ESQNIST up and running

esqopr: dis=act

Active ADABAS SQL servers on node GENESIS:

   Name     Catalog     Last Activity    act.ses  act.req  tot.ses  tot.req

 ESQNIST    214/004   15-APR-1998 16:47        0        0        0        0

esqopr: shutdown
Really SHUTDOWN server ESQNIST ? y
Enter reason for shutdown: System Maintenance
%ESQOPR-I-SHUTDWN, Server shutdown initiated at 15-APR-1998 17:12:37.32
%ESQOPR-I-DETACH,  Server ESQNIST now detached
esqopr: quit
%ESQOPR-I-TERMINATED, 15-APR-1998 17:12:44
```

User Exits

Overview

A user exit is a user-written routine that enables the user to participate in the processing performed by the ADABAS SQL Server. The user-written routine is dynamically loaded at the startup of the server or application, and is called at predefined stages in the processing of either.

The routines should be written in the C programming language.

On Windows NT a user exit must be present as an exported function in a DLL.

In the product version subdirectory `files` there is a file (`makefile`) containing an example makefile for use with the `nmake` command. This example assumes that all four possible user exit functions are present in a single source that will be built in a single DLL. It is, of course, possible to put different exit functions in different sources and/or different DLL's.

The following user exits are available for the ADABAS SQL Server:

User Exit	**Use**
Client user exit 1	user processing on an ESQLNK call before it is processed by the server.
Client user exit 2	user processing on an ESQLNK call after it has been processed by the server.
Server user exit 5	user processing on an ADABAS call. The function is called before the ADABAS call is executed.
Server user exit 6	user processing on an ADABAS call. The function is called after the ADABAS call is executed.

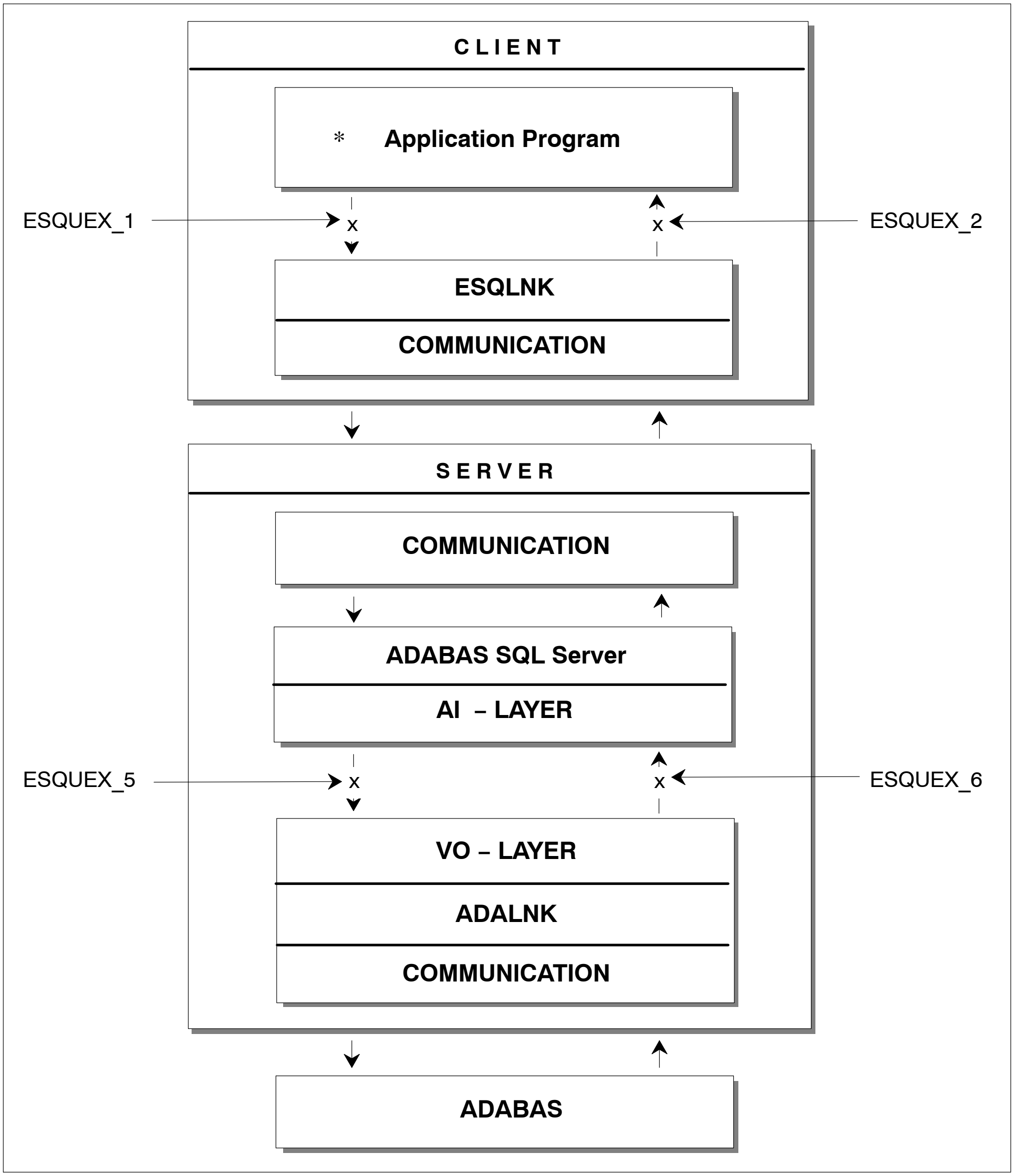


Figure 3-2: Locations of user exits within the ADABAS SQL Server

With the user exit logging you are able to log the user exit call and some parameters.

Descriptions

Client User Exit 1

The ADABAS SQL Server user exit 1 is a function that performs user processing on an ESQLNK call. The user exit is called when the processing of a statement begins.

```
Function’s prototype: void uex_1(   struct sqlca * p_sqlca,
                                        struct esq_uex1_inp * p_inp,
                                        struct esq_uex1_out * p_out ):
```

Client User Exit 2

The ADABAS SQL Server user exit 2 is a function that performs user processing on an ESQLNK call. The user exit is called when the processing of a statement ends. The input parameter that is specified enables the user exit to change the response code of the ESQLNK call.

```
Function’s prototype:     void uex_2(   struct sqlca *p_sqlca);
```

Server User Exit 5

The ADABAS SQL Server user exit 5 is a function that performs user processing on an ADABAS call. The user exit is called when the processing of an SQL request causes an ADABAS call and before this ADABAS call is executed.

```
Function’s prototype:     int uex_5(    unsigned char  * CB,
                                        ESQACC         * p_acc,
                                        unsigned char  * FB,
                                        unsigned char  * RB,
                                        unsigned char  * SB,
                                        unsigned char  * VB,
                                        unsigned char  * IB);
```

Server User Exit 6

The ADABAS SQL Server user exit 6 is a function that performs user processing on an ADABAS call. The user exit is called after an ADABAS call has been executed.

```
Function’s prototype:    int uex_6(    unsigned char   * CB,
                                       ESQACC          * p_acc,
                                       unsigned char   * FB,
                                       unsigned char   * RB,
                                       unsigned char   * SB,
                                       unsigned char   * VB,
                                       unsigned char   * IB);
```

Common Rules for Server User Exits 1 and 2

Parameters

p_sqlca	provides access to the SQLCA
p_inp	provides reads access to the input data (user ID, password, etc.)
p_out	provides write acesss to the output data (user ID, password, etc.)

Examples

The user exit calls the function my_security() for a CONNECT statement. If this function fails, ESQERR_USER_AUTH_FAIL is returned via sqlcode field of SQLCA. The parameters of the function my_security() are user-ID and password. This information was passed on to the user exit via the parameter struct esq_uex1_inp.

```
#include <stdio.h>
#include <esqacc.h>
#include <esqerr.h>
#include <esquex1.h>
void uex_1(
     struct sqlca * p_sqlca,
     struct esq_uex1_inp * p_inp,
     struct esq_uex1_out * p_out )
{
     if (p_inp->sql_command == ESQ_CONNECT)
     {
         if (my_security(
             p_inp->l_name,    p_inp->c_name,
             p_inp->l_pwd,     p_inp->c_pwd     ))
           p_sqlca->sqlcode = ESQERR_USER_AUTH_FAIL;
     }
     return;
}
```

The following routine changes the user ID to JOHN for a CONNECT statement.

```
#include <stdio.h>
#include <esqacc.h>
#include <esqerr.h>
#include <esquex1.h>

void uex_1(
    struct sqlca * p_sqlca,
    struct esq_uex1_inp * p_inp,
    struct esq_uex1_out * p_out )



{
    if (p_inp->sql_command == ESQ_CONNECT)
    {
        p_out->l_name = strlen( ”JOHN” );
        strcpy( p_out->c_name, ”JOHN” );
    }
    return;
}
```

Common Rules for Server User Exits 5 and 6

Parameters

The following explains the parameters of the function's prototype for user exit 5 and 6:

- CB provides access to the ADABAS control block as specified in the *ADABAS Command Reference Manual*.
- ESQACC provides access to the ADABAS SQL Server session context.
- FB provides access to the ADABAS format buffer.
- RB provides access to the ADABAS record buffer.
- SB provides access to the ADABAS search buffer.
- VB provides access to the ADABAS value buffer.
- IB provides access to the ADABAS ISN buffer.

Note:
Ensure that the user exit is prepared to handle NULL pointer values for some of the parameters.

ADABAS SQL Server Session Contexts

The ADABAS SQL Server session context is provided in the include file <esqacc.h>, and is defined as follows:

```
#define  ESQRTS_CONTEXT     (-1)
#define  L_CLIENTS_NAME     (18)
#define  L_CLIENTS_PWD      (32)
#define  L_CLIENTS_NODE     (8)

typedef struct esqacc
 {
  long    c_sid;    /* session id           */
  long    c_con;    /* current context      */
  unsigned char c_node [ L_CLIENTS_NODE ];
  unsigned char c_name [ L_CLIENTS_NAME ];
  unsigned char c_pwd  [ L_CLIENTS_PWD ];
 } ESQACC;
```

As stated in the *ADABAS SQL Server Programmer's Guide*, **Appendix D**, one active client may occupy up to two ADABAS session contexts (user queue elements). In order to distinguish between these contexts, the following rules have been established:

RULE_0: c_pwd will be set to binary zeroes and is, therefore, not visible.

RULE_1: c_sid is the session identifier for the currently active client session. The value is assigned by the ADABAS SQL Server.

RULE_2: c_con is the currently active operation context for the current ADABAS call.

RULE_3: If c_con is equal to c_sid, then the current ADABAS call is executed on behalf of the client, for example, to retrieve data.

RULE_4: If c_con is equal to ESQRTS_CONTEXT, then the current ADABAS call is executed on behalf of the ADABAS SQL Server, for example, to store a meta program.

Embedding Server User Exits 5 and 6

The user exit processing has been implemented within the ADABAS SQL Server as follows (pseudo language example):

```
.
.
if (ESQUEX5 defined)
     BEGIN
     rc = ESQUEX5 (CB,ESQACC,FB,RB,SB,VB,IB);
     if (rc <> 0)
          BEGIN
          CB -> response_code = rc;
          goto do_not_call_ADABAS;
          END
     END

ADABAS (CB, ......);

if (ESQUEX6 defined)
     BEGIN
     rc = ESQUEX6 (CB,ESQACC,FB,RB,SB,VB,IB);
     if (rc <> 0)
          BEGIN
          CB -> response_code = rc;
          END
     END
do_not_call_ADABAS:
.
.
```

User Exit Response Codes

Both user exits 5 and 6 are specified to return an integer value via the “return” statement. The returned value is interpreted within the ADABAS SQL Server as an ADABAS response code.

User Exit 5

- If user exit 5 returns the value zero, the current ADABAS call is executed.
- If the returned value is non-zero, the value is placed in the provided ADABAS control block and the ADABAS call is not executed.

User Exit 6

- If user exit 6 returns the value zero, the processing continues as if no user exit was called.
- If the returned value is non-zero, the value is placed in the provided ADABAS control block as if ADABAS had returned this value.

Based on the above, any returned non-zero value may be visible to the ADABAS SQL Server client (application program). It is, therefore, advisable to use appropriate values, such as reponse code 200, to signal security violations. Refer to the *ADABAS Messages and Codes Manual* for further response codes.

The value passed on must be within the following range: 0 <= value <= 255

Note:
some original ADABAS response codes trigger special exception handling routines. In case of the original ADABAS response code 9, an ADABAS SQL Server internal “reopen” logic for the ESQRTS context is started.

Creation and Definition

The installation contains an example for client user exit 1 written in C.

A user exit is defined by performing the following steps:

1. Write the user exit in the C programming language.

   The exits should be written with default function names. The convention is ”uex_ ” followed by the number of the user exit.

2. Compile the source file(s) of the user exit(s). See example file `files\makefile` under the product version directory for a specific compilation example.

3. Link the user exit as a DLL. See example file `files\makefile` under the product version directory for a specific link example.

4. Make the user exit available to the ADABAS SQL Server by setting an environment variable to point to the shared library. The environment variable is ”ESQUEX_” followed by the number of the user exit. These environment variables can be set system wide or for specific users via Control Panel–>System–>Environment. If you are using more than one user exit in a single DLL, set each environment variable to point to the same DLL.

```
setenv ESQUEX_1 esquex.dll          # set user exit 1
setenv ESQUEX_2 esquex.dll          # set user exit 2
```

Parameter Processing

The execution of the 3 major components of the ADABAS SQL Server (Server, Precompiler, and Runtime System) can be influenced by parameter settings via PPL (Parameter Processing Language).

When generating a server environment, the following three parameter files are created in the server-specific directory ESQSRVDIR:

- **ESQSRVDIR\esqsrv.par (server parameter file)**
  This parameter file is read in at server startup time. One use could be to adjust the number of threads started by the server depending on the number of concurrent sessions. It defines parameters, such as server name, server catalog DBID/FNR, communication type (CSCI), number of server threads, default client parameters, request/reply buffer lengths, optional: server trace/log control statements etc.
- **ESQSRVDIR\esqpc.par (precompiler parameter file)**
  This parameter file is read in at precompiler startup time. It defines parameters, such as default schema identifier, maximal number of compile errors, server catalog DBID/FNR (for single-user linked-in mode), optional: precompiler trace/log control statements etc.
- **ESQSRVDIR\esqrun.par (client parameter file)**
  This optional parameter file is read in at client application startup time. It defines parameters, such as default schema identifier, server session time-out value, maximum number of cursors, server catalog DBID/FNR (for single-user linked-in mode), optional: client trace/log control statements etc.

These files can be customized to meet server-specific needs.

Sort Buffer Size Setting

Using the ORDER BY or the GROUP BY clause may sometimes force the ADABAS SQL Server to perform internal sorting of data.

If the default size (16 KB) of the internal buffer which holds the data to be sorted is exceeded, the overflow will be written to a temporary file. To balance this behavior, the buffer size can be set externally via the environment variable ESQSRTBUF.

A detailed analysis of the resource usage during the sort can be obtained by setting the environment variable ESQLOG to “sort”. An example with the use of the default buffer size (16KB) can be found in the chapter **Logging Facilities**, section **Sort Logging** in the *ADABAS SQL Server Programmer’s Guide.*

OPERATING THE ADABAS SQL SERVER UTILITIES

This chapter contains a detailed description of how to work with the four ADABAS SQL Server Utilities.:

1. BASIC Interactive Facilities (BASIC)

   The BASIC Interactive Facilities consist of two elements which can be used on any platform.

   - BASIC Interactive SQL allows the user to enter SQL statements interactively and see results displayed on the screen immediately.
   - BASIC Catalog Retrieval allows the user to retrieve catalog information via predefined views.

2. Generate Table Description Utility

   If existing ADABAS files must be introduced to the ADABAS SQL Server via the CREATE TABLE/CLUSTER DESCRIPTION statements, the drafting of such statements can be a complex operation, as well as a laborious one. By using this utility, you can easily draft up CREATE TABLE/CLUSTER DESCRIPTION statements. They may then be edited by hand and be submitted to the server via the BASIC Interactive SQL Utility.

3. ESQSHOW Utility

   The ESQSHOW utility provides the user with information about a server that has been established as the default server, as well as a list of all local and remote servers defined in the current environment.

4. ESQSET Utility

   The ESQSET utility allows the user to change the default server setting for his/or environment.

BASIC Interactive Facilities

Target Users

BASIC has been designed for users who require quick and easy access to:

- interactively entered SQL statements and immediate results.
- catalog information, for example, databases, tables, views.

Overview

The BASIC Interactive Facilities consists of both a command line interface and GUI tool. The BASIC Interactive Facilities are an interactive SQL query tool which supports basic functionality and is fast and effective. The BASIC Interactive Facilities consist of two elements:

- BASIC Interactive SQL and
- BASIC Directory Retrieval

Statements containing cursor names, host variables or parameter markers cannot be executed interactively. Statements marked for embedded or dynamic mode in the *ADABAS SQL Server Reference Manual*, for example, the CLOSE, DECLARE or CONNECT statements cannot be used in this context.

Invoking the BASIC Interactive Facilities

BASIC Command Line Interface

To invoke the BASIC Interactive Facilities via the command line, type the following:

```
esqint [servername [communication [destination]]]
```

The optional parameters must be specified according to the same rules as in the Server Routing File. For details see the *ADABAS SQL Server Programmer's Guide*, chapter: **Client/Server Topics.**

Setting Server Name, Communication and Destination

Examples:

To connect to a local server named LOCESQ, enter:

```
esqint LOCESQ
```

To connect to the remote server named VAXESQ via CSCI on node SAGVAX11, enter:

```
esqint VAXESQ CSCI SAGVAX1
```

To connect to the remote server named IBMESQ via the ENTIRE BROKER with the broker ID IBMBROKER11, enter:

```
esqint IBMESQ BROKER IBMBROKER11
```

Setting User ID and Password

BASIC will then issue prompts asking for the user ID and password which will be used to perform a CONNECT statement to the server. When the input is redirected to a file, the user ID and password must be in the first two lines of that file.

Alternatively, the symbol ESQUSER can be defined with user ID and password. The prompting will be suppressed in this case. In a system with the security feature, the symbol ESQUSER must contain both the user and the password, separated by a comma: “userid,password”. In a non-security system, the environment variable needs to contain the user ID only.

If, in a non-security system, an empty string is used for the user ID, a default CONNECT will be executed. The user ID is then identical to the operating system user ID.

```
esqint

ESQ User:
Password:

esqint:
```

A system prompt will appear and ad-hoc SQL statements may be entered at this point.

BASIC GUI Tool

The BASIC Interactive Facilities GUI Tool may be invoked via Windows NT Explorer or command line:

Windows NT Explorer:	Double click on "basic.exe" found in the ..\bin directory of the current version of ESQ.
Command line:	Since the ..\bin directory of the current version of ESQ is in the PATH, simply type **basic**

The ADABAS SQL Server Login Screen will appear. From here you are required to enter Username, Password, Server name, and Communication type. The server list contains all local servers and those remote servers which are defined in the ADABAS SQL Server routing file "esq_routing.dat". You must choose from one of the servers in the list. If a Default Server name has been set either via the ESQSET Utility or during server generation, it will be the default here.

BASIC Interactive SQL

BASIC Command Line Interface

BASIC Interactive SQL allows the user to enter SQL statements interactively and see results displayed on the screen. Conceptually it consists of 3 elements:

- Input Mode with a set of Input Mode Commands,
- Output Mode with a set of Input Mode Commands,
- Help Mode with Online Help Texts.

In addition to these three elements, there is a set of Global Commands for general navigation purposes.

Global Commands

Command	Description
exit	returns to the higher level.
help	activates the help function.
quit	returns to the higher level.
spawn *command*	spawns the specified DCL command.
$ *command*	spawns the specified DCL command (abbreviated form).

Input Mode (esqint:)

When you receive the prompt “esqint:” BASIC is automatically in the input mode and a line editor is provided. SQL statements can now be entered or edited. An SQL statement can consist of several lines. Each statement must end with a semicolon. If ENTER or RETURN is pressed and no semicolon is found, the line editor provides a new line with a descending line number. If a semicolon is found the statement is executed immediately after pressing ENTER or RETURN. Executing a statement automatically activates the output mode where the results are displayed. Only one statement can be active at a time and this so-called currently active statement can be run or edited again. In the case of syntax errors or other reasons for re-editing the statement: leave the output mode, list and edit the statement and run it again. Already saved statements have to be read into the input session to take the place of the currently active statement. If the previously active statement has not been saved, it is no longer available. The compressed maximum length of a statement is 63000 characters, whereby the term compressed means that multiple white spaces are replaced one blank.

Executing Interactive SQL Statements

After entering a new valid SQL statement on the blank input screen, properly end it with a semicolon and press ENTER or RETURN to immediately execute the statement.

A currently active and valid statement can be executed via the "run" command.

A saved and valid statement must be read into the input area and executed via the "run" command.

Input Mode Commands

Command	Description
continuation "*x*"	defines the character "x" as a line continuation marker.
edit *nn*	edit line *nn* of the current SQL statement.
list	lists the current SQL statement.
read *xx*	reads the file *xx* into the input session.
run	executes/runs the current SQL statement.
save *xx*	saves the SQL statement with the name *xx*.
server	displays server information.

Output Mode (output (col *l-r* of *t*):)

The output mode is invoked by executing a valid SQL statement and is indicated by the prompt "output (col *l-r* of *t*):". The numbers in brackets indicate the offset in spaces when executing the output mode command "right".

Where *(l)* and *(r)* mark the left and right margin of the output window in reference to the size of the total output window (*t*) which depends on the number of columns requested.

The other output mode commands available are "left" which moves the display window to the left and "home" which returns to column 1. Scrolling down page by page is achieved by pressing ENTER or RETURN. To return to the input mode, the global commands "quit" or "exit" are used.

Output Mode Commands

Command	Description
home	scrolls back to the first column.
left	scrolls output window to the left.
right	scrolls output window to the right.

Online Help

Typing "help" after the prompt will display the online help text informing the user about the commands available in BASIC.

Summary: Executing Statements

Working with the Currently Active Statement:

1. At the prompt “esqint:” type in the statement considering the syntax regulations. The statement may exceed one line,
2. delimit each statement with a semicolon (;),
3. to immediately execute the statement, press ENTER or RETURN. You are now in the output mode. If you want to edit the statement (it is still active) or simply want to return to the input mode,
4. enter “quit” at the prompt output(col *l-r* of *t*):,
5. enter “list” which will list the entire statement and note the line number(s) to be edited (only one line can be edited per edit command),
6. enter “edit *n”* where *n* is the number of the line to be edited. After editing, you can either execute the statement again by entering “run” or save it with a name by entering “save *name”*.

Working with a Previously Saved Statement:

1. A saved statement has to be read into the input session to gain the status of an currently active statement. Enter “read *name”,*
2. now you may list, edit, run or save the statement as described above.

BASIC Catalog Retrieval

BASIC Catalog Retrieval allows the user to retrieve information stored in the catalog, like tables, columns, etc.

How to Retrieve Information from the Catalog

The following views are available: databases, views, indexes, metaprograms, tables and columns. After starting the BASIC Interactive Facilities, the following statements may be entered:

To display all databases, type the following:

esqint: select * from information_schema.databases;

To display all tables, type the following:

esqint: select * from information_schema.tables;

To display all tables, type the following:

esqint: select * from information_schema.tablespaces;

To display all columns of table SAILOR, type the following:

esqint: select * from information_schema.columns where table_name = ”SAG-TOURS.SAILOR”;

To display all *views*, type the following:

esqint: select * from information_schema.views;

To display all indexes, type the following:

esqint: select * from information_schema.indexes;

Note:
The results of the SELECT statements are displayed in output mode in the same manner as other SELECT statements entered in the BASIC Interactive SQL. The same set of commands applies.

BASIC GUI Tool

BASIC Interactive SQL GUI Tool allows the user to enter SQL statements interactively via the Query Form and immediately view the results in an output grid. It also offers the user with the ability to browse the server’s catalog file via the Catalog Browser.

Query Form

After successfully connecting to a server, a Query form is displayed. When the Query form is displayed, SQL statements can now be entered or edited in the Query window. An SQL statement can consist of several lines. If more than one valid SQL statement is entered, a syntax error is reported. A semi-colon is not required to terminate an SQL statement. However, if a semi-colon is used, all information following the semi-colon will be ignored.

Executing Interactive SQL Statements

You may execute an SQL statement by clicking the “Run Query” button, pressing F8, or selecting “Run Query” from the Run Menu Option. Only one statement can be active at a time and this so-called currently active statement can be run or edited again. The result of the run will be displayed in the Status Window. In the case of an error, simply modify the statement if appropriate and execute it again.

If the SQL statement is a SELECT statement, the resultant rows will be displayed in the Result Grid. If the resultant rows exceed the size of the current window, horizontal and vertical scroll bars will be provided to view the information. To limit the number of resultant rows returned via a SELECT statement, go to the “Limit # of Rows” option found in the Options Menu.

The Results Grid will not be cleared until another SELECT statement is issued.

Saving and Restoring SQL Statements

You may save the currently active SQL statement at any time by simply selecting the “Save Query” option found in the File Menu. You will then be prompted with a “Saved As” menu. SQL Statements will be stored in simple text format.

To restore a previously saved SQL statement, simply select the “Open Query” option found in the File Menu. You will then be prompted with a “Open” menu. The contents of the opened file will be placed in the Query Window.

When exiting the Query form, you will be prompted with a message asking if you wish to Save the currently active SQL statement if it has not already been saved.

Saving Results Grid

You may save the contents of the Results Grid at any time by simply selecting the “Save Results” option found in the File Menu. You will then be prompted with a “Saved As” menu. Results will be stored in a simple text format. Columns will be delimited by a TAB unless otherwise specified via the “Results File Layout” option found in the Options Menu.

Result files may then imported into spreadsheet applications for further processing. They cannot be re-read into the BASIC Interactive GUI tool.

When exiting the Query form, you will be prompted with a message asking if you wish to Save the Results Grid if it has not already been saved.

MENU OPTIONS AVAILABLE

FILE MENU

New Session	Allows the user to login to another server by starting another BASIC INTERACTIVE GUI session.
Close Session	Close the currently active session and send a DISCONNECT to the server. This will also close the Catalog Browser (discussed later in this chapter), if currently open.
Catalog Browser	Start the Catalog Browser (discussed later in this chapter).
Open Query	Restore a previously saved SQL statement from a text file.
Save Query	Save the currently active SQL statement to a text file.
Save Results	Save the Results Grid to a text file.
Exit	Disconnect from the currently active session and exit the BASIC Interactive GUI tool. This does not affect other BASIC Interactive GUI sessions started via the “New Session” option.

EDIT MENU

This menu provides all editing functions available to the Query Window.

RUN

Run Query	Run the currently active SQL statement.

OPTIONS

Limit # of Rows	Set a limit on the number of resultant rows to be returned as a result of a SELECT statement.
Results File Layout	Select the column delimiter to be used when saving the Results Grid to a text file. This is useful when importing the results file into a spreadsheet application.

HELP

Help Topics	On-line help for the BASIC Interactive GUI Tool
Help About	Version information for the BASIC Interactive GUI Tool

Catalog Browser

The Catalog Browser allows the user to retrieve information stored in the catalog, like tables, columns etc. The information is displayed in a hierarchical format and is read-only.

The Catalog Browser and Query Form access the server via the same session. Therefore, it is not possible to send SQL commands simultaneously from both windows.

To invoke the Catalog Browser, select the “Catalog Browser” from the File Menu while in the Query Form.

When the Catalog Browser is displayed, you may click on the “DB” folder to open the current server catalog.

Clicking on the server catalog will display schemata, databases and users defined in the current server catalog.

You are then able to traverse through the catalog by clicking on any one or more of these items.

Buttons Available

Display Buttons are available to change the way the information is displayed. Currently four display buttons are available:

- Large Icon
- Small Icon
- List
- Details

Generate Table Description Utility

Target User

Anyone who wants to introduce an existing ADABAS file to the ADABAS SQL Server.

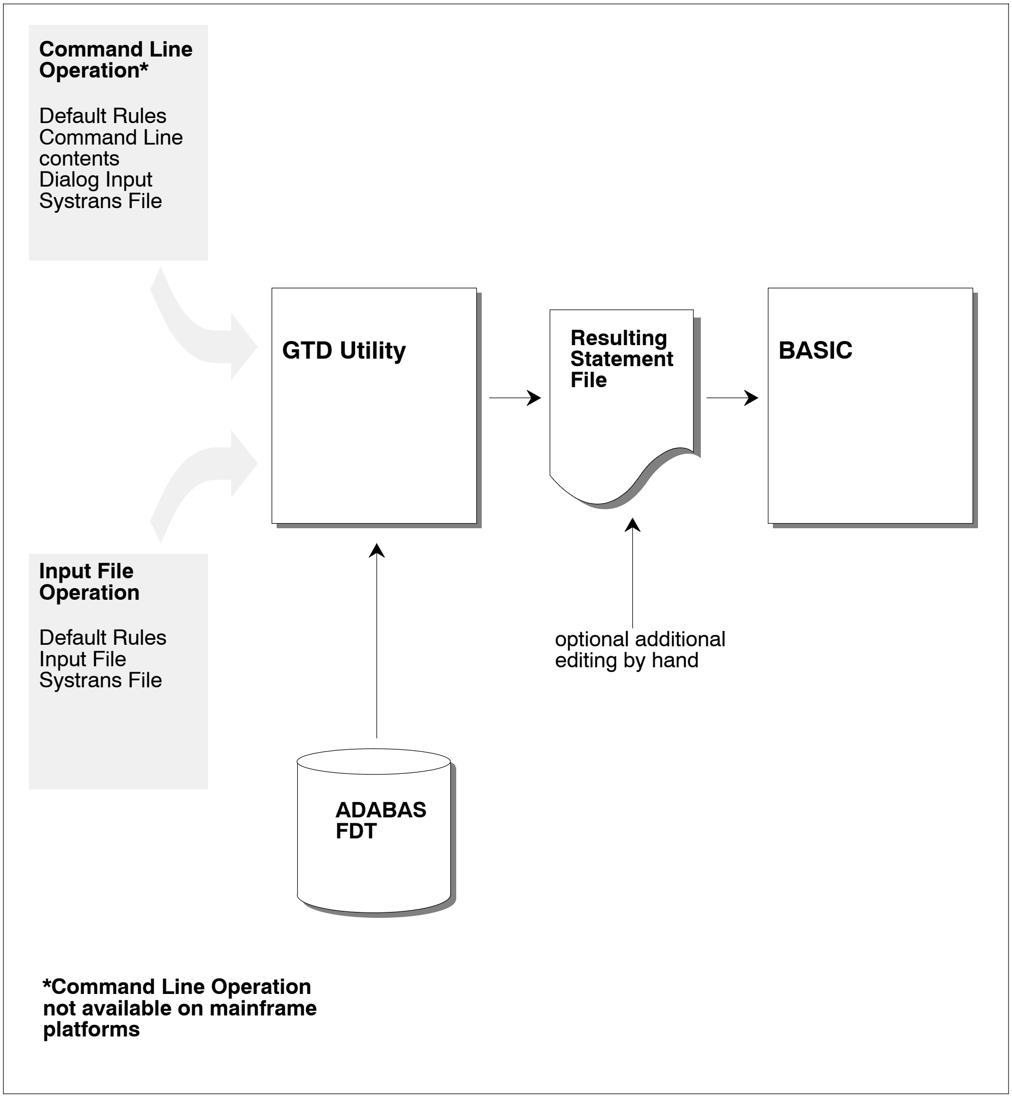
Overview

The Generate Table Description Utility (in the following called GTD Utility) is used to generate either CREATE TABLE DESCRIPTION statements or CREATE CLUSTER DESCRIPTION statements for existing ADABAS files. To make existing ADABAS files accessible, they must be introduced to the ADABAS SQL Server by means of these statements. The drafting by hand of such statements can be a complex operation, as well as a laborious one. Therefore, the GTD Utility is used to automatically generate a text file containing such statements. This text file must then be submitted to the Basic Interactive Utility command line interface “esqint” for execution, thus introducing the existing ADABAS file to the ADABAS SQL Server.

The GTD Utility can be operated in two distinct operative modes:

- Command Line Operation
  Command line operation is best suited for quick and easy use. From command line operation the dialog mode can be activated as an option. The dialog mode provides a large subset of the full functionality of the utility.
- Input File Operation
  The full functionality of the utility is only available in input file operation. Fine tuning of the resulting generated statements is possible by modifying and resubmitting the input file.

It can be seen from the diagram, that the GTD Utility obtains information from various sources, depending upon the operative mode. The resultant generated statement is dependent upon the information which the GTD Utility will obtain from one or more of these sources:



Note:
Some of the above information sources are mutually exclusive. Please refer to the section: Informations Sources Overview later in this chapter.

The resultant statement output file is not automatically presented to the ESQINT Basic Interactive Utility command line interface. There is therefore always the opportunity to review the generated output and indeed edit it further by hand.

Because the GTD Utility must obtain information from the relevant ADABAS FDT, during its execution, it is therefore a pre-requisite that the appropriate ADABAS database is online and accessible.

Invoking the GTD Utility

As mentioned in the Overview, there are two distinct operative modes. These modes are mutually exclusive.

Command Line Operation

In command line operation mode, the GTD Utility is invoked as follows:

```
esqgtd  [option] database_name  database_number  file_number  [ddm_name]
```

where:

option: none, one or more of the following options may be specified in any order:

–f dialog mode is suppressed: default rules therefore apply.

–d The generation of the column's data type is forced in the statement. Normally, the data type is only generated for long-alpha character fields, or numeric or decimal fields where a scale is specified in a NATURAL DDM SYSTRANS file.

–t *systrans_file* This option has an additional parameter. The option indicates that a NATURAL DDM SYSTRANS file is to be used as an additional information source. The parameter *systrans_file* must indicate a valid file path name, where the file contains the SYSTRANS DDM information to be considered.

–s *schema_name* This option has an additional parameter. The option indicates that an explicit schema identifier is to be generated in the resultant statement. The parameter *schema_name* specifies this identifier.

−a In dialog mode, a prompt is issued for all fields. Normally, only MU/PE fields are prompted.

−o in dialog mode, a prompt is issued for the column and subtable identifiers.

−h invokes on-line help only.

database_name specifies the name of the ADABAS SQL Server target database. This name will appear in the final statement.

database_number specifies the number of the ADABAS source database. The ADABAS file to be described must reside in this database.

file_number specifies the number of the ADABAS source file to be described.

ddm_name this optional parameter specifies the particular SYSTRANS DDM in the *systrans_file* which is to be used as additional information. If the *ddm_name* contains a period ".", the part before the period is the NATURAL library name and the part after the period is the real DDM name.

For example:

```
esqgtd -f -t systrans.file DBNAME 202 181 > gen.statement
```

Note:
The generated statement is to be found in the output file named gen.statement.

Input File Operation

In input file operation mode, the GTD Utility is invoked as follows:

```
esqgtd < input_file
```

where:
input_file must indicate a valid file path name. The indicated file must contain a valid input file specification (see below).

For example:

```
esqgtd < input.file > gen.statement
```

Information Sources Overview

The interpretation that can be placed upon an ADABAS file in terms of SQL table mapping is numerous. The GTD Utility can generate differing statements based upon the same ADABAS FDT depending on the information that is presented to it. The information sources and the information presented within these sources, control the nature of the resultant statement.

The ADABAS FDT

The GTD Utility has the task of producing a description of an ADABAS file in terms of SQL. It therefore takes, as its starting point, the ADABAS FDT (Field Description Table) of the file to be described. The appropriate ADABAS database must be online and accessible to the GTD Utility. The information derived from this source is:

- The ADABAS file structure.
- The ADABAS field short names.
- The ADABAS field data types.

Command Line Information

When in Command Line Operation mode, the GTD Utility obtains vital information via the command line input. The presence of command line input, automatically suppresses Input File Operative mode. Some of the information is mandatory. Some of the information also indicates if further information sources are to available. The information derived from this source is:

- The DBID and file number of the ADABAS file to be described (mandatory).
- An indication if dialog mode is to be used as an information source (optional).
- An indication if a NATURAL DDM SYSTRANS file is to be used as an information source (optional).

Should additional information sources not be available, i.e. the dialog mode has been suppressed (the −f option) and no SYSTRANS file has been specified, then the resultant statement is generated according to the default rules.

Dialog Input

If in the command line information, dialog input has not been explicitly suppressed, then a dialog will be initiated by the GTD Utility with the user. The user dialog is intended as a comfortable interface to the GTD Utility, which is quick and easy to use and will cover most cases. The user always has the option to manually edit the resultant file. Via the dialog, the interpretation of the ADABAS fields themselves can be controlled and the following can be influenced:

- PE groups can be interpreted as a subtable.
- PE groups can be suppressed, in which case:
  - PE fields can be interpreted as a rotated field.
  - PE fields can be interpreted as long-alpha (if of data type character).
  - MU fields within the PE group are automatically suppressed.
- MU fields can be interpreted as a subtable.
- MU fields can be suppressed.
- MU fields can be interpreted as a rotated field.
- MU fields can be interpreted as long-alpha (if of data type character).

If the GTD Utility is additionally invoked with the "−o" option, then the user is prompted for the column and subtable identifiers. Otherwise the default rules for identifier generation are employed.

If the GTD Utility is invoked with the −a option, then a prompt is issued for all fields, thus enabling the suppression of a non MU or PE field. Otherwise only fields where a decision is required would be presented within the dialog i.e. MU or PE fields.

Default Rules

Default rules apply when no other information is available to the GTD Utility. This may be that no information is available at all for the specified ADABAS file, in which case the default rules apply to all aspects of the statement generation. Alternatively, no additional information is available for a particular field, in which case the default rules apply only to that particular field.

By default:

- Any MU field will be interpreted as a subtable.
- Any PE group will be interpreted as a subtable.
- If the file contains no MU or PE fields, then a CREATE TABLE DESCRIPTION statement is generated.
- If the file contains MU or PE fields, then a CREATE CLUSTER DESCRIPTION statement is generated.
- All available SEQNO columns will be generated into the table descriptions.
- Any foreign key relationship will always be based upon appropriate SEQNO column(s).
- The cluster identifier will be “CLUSTER_XXX”, where “XXX” is the file number.
- The master table identifier will be “TABLE_XXX”, where “XXX” is the file number.
- The table identifier of a subtable derived from a PE group will be “TABLE_XXX_YY”, where “XXX” is the file number and “YY” is the PE group short name.
- The table identifier of a subtable derived from an MU field will be “TABLE_XXX_YY”, where “XXX” is the file number and “YY” is the field short name.
- The table identifier of a subtable derived from numerous MU fields in parallel will be “TABLE_XXX_YY”, where “XXX” is the file number and “YY” is the field short name of the first MU field. Note this interpretation of numerous MU fields is itself not by default.
- The column identifier will be “COL_YY”, where “YY” is the field short name.

The pure default operation is intended for quick spontaneous use. It does, however, lead to very cryptic column and table names and may also lead to cluster structures which are not what the user requires. This approach is intended as a starting point. The user always has the option of either editing the generated statement by hand or introducing additional information sources.

The NATURAL DDM SYSTRANS File

A NATURAL SYSTRANS file can be introduced to the GTD Utility, primarily for the purpose of supplying long names for columns and indeed, in some cases, for the tables themselves. The information that it contains is therefore merged with the basic information of the ADABAS FDT.

The following information is extracted from a NATURAL DDM SYSTRANS file:

- Long names for master tables.
- Long names for periodic group-based (PE) subtables.
- Long names for multiple-value fields (MU) based subtables.
- Long names for column identifiers (except SEQNO columns).
- The precision and scale of a column of data type numeric or decimal.
- The explicit data type of natural date or natural time columns.

Names of master tables, or tables which do not appear in a cluster description, are derived from the actual DDM name itself. If this name is qualified with a library name, then this is ignored for the purposes of table identifier generation.

Names of subtables based upon a PE group are derived from the concatenation of the generated table identifier (itself derived from the DDM name) with the PE group long name.

Names of subtables based upon an MU are derived from the concatenation of the generated table identifier (itself derived from the DDM name) with the MU field long name.

Names of subtables based upon numerous MU fields in parallel are derived from the concatenation of the generated table identifier (itself derived from the DDM name) with the first MU field long name.

Column names are simply derived from the long name associated with the appropriate ADABAS field. If the column is derived from a rotated field, then a numeric suffix is concatenated.

Should any resultant identifiers not correspond to a legal SQL identifier, then it will be appropriately amended. Such an action will result in a warning being generated. Examples of common conversions are as follows:

- A delimiter character has been used in the long name e.g. “yearly-bonus” would become “yearly_bonus”
- An SQL keyword has been used for a long name e.g. “group” would become “group_”
- The long name does not start with an alphabetic character e.g. “1stbonus” would become “col_1stbonus”.

Should any resultant long name contain more than the permitted 32 characters, then a warning is generated. It is then necessary to amend such identifiers by hand in the output file. Failure to do so will result in an error when attempting to execute the generated statement.

Any information extracted from a DDM file is subordinate to any contradictory information presented to the GTD Utility from other sources.

It is sometimes the case that the DDM in a SYSTRANS file refers to a DBID and a file number that does not correspond with the physical DBID and file number of the file concerned. In such a case, an explicit DDM name must be supplied, in order to identify the desired DDM. In such a case the DBID and the file number given in the DDM are ignored.

It is sometimes the case that the SYSTRANS DDM file actually contains more than one DDM representation. Only one DDM may be considered by the GTD Utility during execution. If the SYSTRANS DDM file does contain more than one DDM representation and the desired DDM is not explicitly specified, then the DBID and file number are used for the purposes of identification. A DBID of “0” in the SYSTRANS DDM file, will be recognized as a match. The first DDM to be found will be the one used during execution, should there be more than one DDM in the file which match the search criterion. Alternatively if an explicit name was given, then the first instance of the specified DDM will be used.

For information on how to create a NATURAL SYSTRANS DDM file, refer to the appropriate NATURAL documentation.

The Input File

The input file can only be presented to the GTD Utility in Input File Operation Mode.

All aspects of the statement generation can be controlled via the input file. In addition to this information source, the information sources NATURAL SYSTRANS DDM file and default information can also be employed. However, information contained in the input file has precedence over any other information source.

In addition to the control which is available in dialog mode, the following functionality is available when using an input file:

- Suppression of SEQNO columns.
- Long names for SEQNO columns.
- Non-SEQNO primary keys in master tables.
- Long names for foreign key columns.
- Specific names for columns derived from rotated field members.
- Subtables consisting of more than one MU field.

Should a particular optional piece of information be omitted from the input file, then the default rules will apply in this particular regard.

How to ....

As already mentioned above, the whole purpose of the GTD Utility is to produce a CREATE TABLE/CLUSTER DESCRIPTION statement, which maps an existing ADABAS file to an SQL data structure. In doing so, choices between various alternatives must be made. A clear understanding of what is to be achieved is required. This sections sets out what particular actions are possible and how to invoke them.

Generate a Cluster Description

The GTD Utility can generate a CREATE CLUSTER DESCRIPTION statement or alternatively a CREATE TABLE DESCRIPTION based upon a single ADABAS file.

Default Rules

If the ADABAS file contains MU or PE fields, then a CREATE CLUSTER DESCRIPTION statement is generated by default. Otherwise a CREATE TABLE DESCRIPTION statement is generated.

Dialog Mode

If the ADABAS file does not contain MU or PE fields, then a CREATE TABLE DESCRIPTION statement is always generated. If however, the ADABAS file does contain MU or PE fields and if these fields are either suppressed or rotated or (if applicable) determined to be long alpha columns, then a CREATE TABLE DESCRIPTION statement is also generated. Otherwise a CREATE CLUSTER DESCRIPTION statement is generated

Input File

If the ADABAS file does not contain MU or PE fields, then a CREATE TABLE DESCRIPTION statement is always generated. If however, the ADABAS file does contain MU or PE fields and if these fields are either suppressed or rotated or (if applicable) determined to be long alpha columns, then a CREATE TABLE DESCRIPTION statement is also generated. Otherwise a CREATE CLUSTER DESCRIPTION statement is generated

Generate a Table Description

The GTD Utility can generate a CREATE TABLE DESCRIPTION statement or alternatively a CREATE CLUSTER DESCRIPTION based upon a single ADABAS file.

Default Rules

If the ADABAS file does not contain MU or PE fields, then a CREATE TABLE DESCRIPTION statement is always generated.

Dialog Mode

If the ADABAS file does not contain MU or PE fields, then a CREATE TABLE DESCRIPTION statement is always generated. If however, the ADABAS file does contain MU or PE fields and if these fields are either suppressed or rotated or (if applicable) determined to be long alpha columns, then a CREATE TABLE DESCRIPTION statement is also generated. Otherwise a CREATE CLUSTER DESCRIPTION statement is generated.

Input File

If the ADABAS file does not contain MU or PE fields, then a CREATE TABLE DESCRIPTION statement is always generated. If however, the ADABAS file does contain MU or PE fields and if these fields are either suppressed or rotated or (if applicable) determined to be long alpha columns, then a CREATE TABLE DESCRIPTION statement is also generated. Otherwise a CREATE CLUSTER DESCRIPTION statement is generated.

Generate a Database Identifier

A database identifier is required as part of the syntax of a CREATE TABLE/CLUSTER DESCRIPTION statement.

Default Rules

No default available.

Command Line

The database identifier is specified as one of the command line parameters.

Dialog Mode

Not available.

Input File

The database identifier is specified via a database directive.

For example:

```
DATABASE 214 DB_NAME
```

Generate a File Number

A file number is required as part of the syntax of a CREATE TABLE/CLUSTER DESCRIPTION statement.

Default Rules

No default available.

Command Line

The file number is specified as one of the command line parameters.

Dialog Mode

Not available.

Input File

The file number is specified via the file specification directive.

For example:

```
FILE 181
```

Generate a Cluster Identifier

A cluster identifier is required as part of the syntax of a CREATE CLUSTER DESCRIPTION statement.

Default Rules

The cluster identifier is generated as “CLUSTER_XXX” where “XXX” is the source ADABAS file number.

Dialog Mode

Not available

Input File

The cluster identifier can be specified via the file specification directive.

For example:

```
FILE 181 CLUSTER_ID
```

Generate a PE Group or an MU Field as a Subtable

Default Rules

By default, a PE group or an individual MU field are always interpreted as a subtable.

Dialog Mode

When a PE group or an MU field is encountered, a prompt will be issued. At this point, the option "T" should be entered.

Input File

By specifying a subtable directive, a PE group or an MU field can be interpreted as a subtable. The following directive would map the field AB to a subtable.

For example:

```
AB SUBTABLE
```

Generate Multiple MU Fields Subtable

Multiple MU fields can be brought together in one subtable. The fields are then in "parallel".

Default Rules

Not possible.

Dialog Mode

Not possible

Input File

This can be specified by use of the SUBTABLE directive. The following directive would map the MU fields to a single subtable containing the fields AB and AC plus the various sequence number columns (total 3 in this example).

For example:

```
AB, AC SUBTABLE
```

Generate a Rotated MU Field

An MU field can also be interpreted as a rotated field.

Default Rules

Not possible

Dialog Mode

When an MU field is encountered, a prompt will be issued. At this point, the option “R” should be entered. A further prompt will be issued which requires the depth of the rotated field.

Input File

An MU field can be interpreted as a rotated field. The following directive would map the 12 occurrences of the field AB to 12 individual SQL columns.

For example:

```
AB ROTATE 12
```

Generate a Rotated PE Field

An individual PE group member field can also be interpreted as a rotated field.

Default Rules

Not possible

Dialog Mode

Initially, when a PE group is encountered the prompt is issued in order to generate a subtable. However, if at this point the table is suppressed (enter “S”), then further prompts can be issued which refer to the individual PE group member fields. These can be rotated by entering “R” in response to the prompt. A further prompt will be issued which requires the depth of the rotated field.

Input File

In order to rotate fields within a PE group, the group itself must be suppressed so that it does not form a subtable. Thereafter, PE group member fields can be rotated. The following directives would generate 12 individual columns based upon the field BA and 20 individual columns based upon the field BB rather than one subtable based upon the PE group BO.

For example:

```
SUPPRESS BO

BA ROTATE 12
BB ROTATE 20
```

Generate a Long-Alpha MU Field

An MU field can also be interpreted as a long-alpha field, provided that the field is of type character.

Default Rules

Not possible

Dialog Mode

When an MU field is encountered, a prompt will be issued. At this point, the option "L" should be entered. A further prompt will be issued which requires the size of the character column.

Input File

An MU field can be interpreted as a long-alpha field. The following directive would map the field AB to an SQL column of 1024 characters.

For example:

```
AB LONGALPHA 1024
```

Generate Table Identifiers

The generation of table identifiers can be specified.

Default Rules

The master table identifier will be generated as "TABLE_XXX", where "XXX" is the file number.

- The table identifier of a subtable derived from a PE group will be "TABLE_XXX_YY", where "XXX" is the file number and "YY" is the PE group short name.
- The table identifier of a subtable derived from an MU field will be "TABLE_XXX_YY", where "XXX" is the file number and "YY" is the field short name.
- The table identifier of a subtable derived from numerous MU fields in parallel will be "TABLE_XXX_YY", where "XXX" is the file number and "YY" is the field short name of the first MU field. Note this interpretation of numerous MU fields is itself not by default

Dialog Mode

In general, the provision of explicit table identifiers in dialog mode is not possible. However, with the specification of the command line option "–o", where appropriate, a prompt is issued requesting a suitable table identifier.

Note:
the specification of a table identifier in dialog mode has precedence over any specification in a NATURAL DDM SYSTRANS file.

Input File

The master table identifier can be specified within the file directive.

For example:

```
FILE 181 Cluster_name City
```

A subtable identifier is specified via the following subtable directive.

For example:

```
B0 SUBTABLE Cities
```

Note:
The specification of a table identifier in the input file has precedence over any specification in a NATURAL DDM SYSTRANS file.

NATURAL DDM SYSTRANS File

- The master table identifier will be generated from the DDM name.
- The table identifier of a subtable derived from a PE group will be concatenation of the DDM name and the PE group name (if available) or the PE shortname.
- The table identifier of a subtable derived from an MU field of level 1 will be the concatenation of the DDM name and the MU field name (if available) or the MU shortname.
- The table identifier of a subtable derived from an MU field of level 2 will be the concatenation of the DDM name, the PE name and the MU field name (if available) or the MU shortname.

Generate Column Identifiers

The generation of column identifiers can be specified.

Default Rules

A column identifier will be generated as “COL_YY”, where “YY” is the field short name.

Dialog Mode

The specification of a column identifier in dialog mode has precedence over any specification in a NATURAL DDM SYSTRANS file.

Input File

A column identifier is explicitly specified via the name directive. This specification would cause the field AB to be generated with the name state_name.

For example:

```
NAME AB state_name
```

Note:
The specification of a column identifier in dialog mode has precedence over any specification in a NATURAL DDM SYSTRANS file.

NATURAL DDM SYSTRANS File

Any name associated with a particular field will be used as the column’s identifier.

Generate Columns Identifiers for Rotated Field Members

Each occurrence of a rotated field must be uniquely identified as a column within the table.

Default Rules

Although rotated fields themselves are not by default generated, identifiers for them are. The default column identifier for a rotated field is “COL_YY_Z” where “YY” is the field shortname of the MU field or the PE group member field and “Z” is the occurrence number of the particular column.

Dialog Mode

In general, the provision of explicit column identifiers in dialog mode is not possible. However, with the specification of the command line option “–o”, where appropriate, a prompt is issued requesting a suitable column identifier prefix. The generated column identifier is the concatenation of this prefix plus the occurrence number of the particular column.

Input File

A rotated field column identifier is explicitly specified via the name directive:

For example:

```
NAME MA 3 BONUS_MARCH
```

NATURAL DDM SYSTRANS File

Any name associated with a particular field will be used as the column’s identifier prefix.

Generate SEQNO Column Identifiers

Any generated SEQNO column will have a column identifier. The contents of a NATURAL DDM SYSTRANS file has no influence on the generation of column identifiers for SEQNO columns

Default Rules

A SEQNO column identifier is by default “COL_SEQNO_0” for level 0 sequence number column, “COL_SEQNO_1” for level 1 and “COL_SEQNO_2” for level 2.

Dialog Mode

No action possible.

Input File

A SEQNO column identifier is explicitly specified via the name directive. The shortname is given as “*0” for a level 0 sequence number, “*1” for a level 1 and “*2” for a level 2:

For example:

```
NAME *0 STATE_ID
```

Generate a Primary/Foreign Key Specification

It is possible to generate primary/foreign key relationships for the various master and subtables, other than what is available by default.

Default Rules

- The primary key for a master table is the column COL_SEQ_0.
- The primary key for a level 1 subtable is the columns COL_SEQ_0 and COL_SEQ_1.
- The primary key for a level 2 subtable is the columns COL_SEQ_0, COL_SEQ_1 and COL_SEQ_2.

Dialog Mode

No action possible.

Input File

If a primary key is required, based on columns other than just the SEQNO column then this is specified via the primary directive.

For example:

```
PRIMARY AA abbreviation
```

Similarly, if a foreign key is required, based on columns other than just the SEQNO column then this is specified via the FOREIGN directive.

For example:

```
FOREIGN AA stat_abrv
```

Suppression of Columns

It may be that certain fields are not required in the generated statement.

Default Rules

All fields are included in the generated statement.

Dialog Mode

Fields that would “naturally” induce a prompt can be suppressed via the input “S”. These are fields which are either members of a PE group or are MU fields.

In general in dialog mode, “normal” fields do not induce a prompt. However, with the specification of the command line option “−a”, a prompt is issued for all fields. All fields can therefore be suppressed via the input “S”.

Input File

Fields can be explicitly suppressed via the SUPPRESS directive.

For example:

```
SUPPRESS BA
```

Input File Language Syntax

The input file is used to provide information to the GTD Utility. All aspects of the generation can be controlled via the provision of an input file.

The file consists of directives. Each directive is started by a new line and is terminated by an end of line. The order of the directives is of no significance.

The GTD Utility is only able to process one ADABAS file at a time.

Comments

Comments are delimited by the appearance of a # character in the first column and by a new line.

SYSTRANS DDM File Specification

Should a SYSTRANS DDM file be required it must be specified by using the following directive:

```
SYSTRANS systrans_file
```

where:

systrans_file	is a valid file name concerned and indicates the SYSTRANS DDM file.

For example:

```
SYSTRANS systrans.ddm
```

Database Specification

The appropriate ADABAS database must be specified by using the following directive:

```
DATABASE db_nr database_name
```

where:

db_nr	is the target ADABAS database number
database_name	is the associated database name which will be used in the generated description statements.

For successful eventual execution of the generated statement, a CREATE DATABASE statement must already have been executed associating this database name with the given database number.

For example:

```
DATABASE 202 TEST_DB
```

Schema Identifier

In order to specify the schema identifier, the following directive is used:

```
SCHEMA schema_name
```

For example:

```
SCHEMA production_schema
```

File Specification

The source ADABAS file is specified by the following syntax:

FILE *file_nr* [DDM=*ddm_name*] [[*cluster_name*] *table_name*]

where:

file_nr	specifies the ADABAS file from which the description statement will be generated.
DDM=*ddm_name*	optionally specifies the exact DDM representation in the SYSTRANS DDM file which is to be used. If the *ddm_name* contains a period ".", the part before the period is the NATURAL library name and the part after the period is the real DDM name.
table_name	optionally specifies the table name which will be used either for the master table or for the single table if there is no cluster.
cluster_name	optionally specifies the cluster name which will be used in the generated statement. If both a cluster name and a table name are present, then this forces the generation of a CREATE CLUSTER DESCRIPTION statement, even if such a statement is not strictly necessary.

For example:

```
FILE 32  DDM=employees  cluster_employees employees
```

Subtable Specification

A subtable can be built around a PE group, a single MU field or a number of MU fields which act in parallel. The subtable directive is able to express either of these possibilities and to enable the optional specification of an identifier for the resultant subtable. In the first two cases, only a single field is required. In the third case, numerous MU fields can be specified in the form of a list separated by commas. The syntax is as follows:

```
short_name [, short_name]... SUBTABLE [table_name]
```

where:

short_name is as it suggests.

table_name represents the intended table identifier for the subtable.

For example:

```
AB SUBTABLE
```

would map the field AB to a subtable and would generate a table name accordingly.

```
AC SUBTABLE Employee_number
```

would map the field AC to a subtable but would use the name as indicated.

```
AE, AF SUBTABLE Employee_status
```

would map the two MU fields to the single subtable Employee_status.

Rotated Field Interpretation

If a field is to be interpreted as a rotated field, then the following syntax applies:

```
short_name ROTATE value
```

where:

short_name is as it suggests.

value is the fixed number of occurrences to be rotated e.g. 12 in our example above.

For example:

```
MA ROTATE 12
```

would map the twelve occurrences of the field MA to twelve individual SQL columns.

Longalpha Field Interpretation

If an MU field of type character is to be interpreted as a longalpha field, then the following syntax applies:

```
short_name LONGALPHA <value>
```

where:

short_name	is as it suggests.
value	is the number of bytes which are to be considered in the character field.

For example:

```
LT LONGALPHA 512
```

This would map the field LT to a character column of 512 characters.

Long name Specification

A column identifier for a particular field can be specified as follows:

```
NAME short_name [index] column_name
```

where:

short_name	is as it suggests.
index	optionally refers to the occurrence number of an MU if the field is rotated.
column_name	is a valid SQL identifier representing the intended column identifier.

For example:

```
NAME MA 3 BONUS_MARCH
```

Any name directive overrides any long name for the column derived from a SYSTRANS DDM file.

Field Suppression

If a field is not to be included in a description, then its suppression is specified as follows:

```
SUPPRESS short_name
```

where:

short_name	is as it suggests.

For example:

```
SUPPRESS BA
```

Foreign Key Specification

The GTD Utility, by default forms the primary key/foreign key relationship, based upon the SEQNO columns. If a different basis for this relationship is required, then the FOREIGN directive is necessary. This directive (or directives) must immediately follow the SUBTABLE directive. A foreign key may consist of one or more columns and this is represented simply by a repetition of the directive. The syntax is as follows:

```
FOREIGN short_name [column_name]
```

where:

short_name	is as it suggests.
column_name	is a valid SQL identifier representing the intended column identifier. This is therefore equivalent to the FOREIGN directive without a column name and an appropriate NAME directive.

For example:

```
FOREIGN AA
```

Primary Key Specification

By default, the utility forms the primary key of a table based upon the SEQNO column. By using the PRIMARY KEY directive, a different basis can be taken. A primary key may consist of one or more columns and this is represented simply by a repetition of the directive. The syntax is as follows:

```
PRIMARY short_name [column_name]
```

where:

short_name is as it suggests.

column_name is a valid SQL identifier representing the intended column identifier. This is therefore equivalent to the PRIMARY directive without a column name and an appropriate NAME directive

For example:

```
PRIMARY AA
```

Data Type Generation

In general, the GTD Utility does not generate the explicit data type declaration for a column in the resultant generated statement. The explicit data type declaration is however not strictly necessary as the ADABAS SQL Server can operate on a minimum create table/cluster statement, i.e. the server will fill in any missing information automatically. In such a case, the resultant generated statement can be significantly longer. The forced generation of the column’s data type is helpful for documentation purposes. The syntax is as follows:

```
DATATYPE
```

Error Handling

Should the GTD Utility encounter an error condition during execution, then it will generally terminate with an exit code and in addition an error message.

The possible exit codes are:

0	Successful completion / or warnings.
−1	Error detected.

The error text can be found in the standard error file.

Examples

The examples presented here are all based upon the following FDT:

Field Definition Table:

Level	Name	Length	Format	Options	Flags
1	AA	2	A	DE,UQ	SP
1	AB	20	A	DE,UQ	
1	AC	20	A	DE,NC	
1	AD	4	F	NC	
1	B0			PE	
2	BA	20	A	DE,NU	SP
2	BB	4	F	NU	
2	CA	20	A	NU,MU	
2	CB	2	F	NU,MU	
2	DA	20	A	NU,MU	

It is assumed that the file is to be found in database 202, file number 181.

Example 1: Default Command Line Invocation

This example shows the simplest way of envoking the GTD Utility, namely via the command line input without any extra information sources other than the default rules.

Invoking the GTD Utility as shown below:

esqgtd –f WS2V14 202 181 > output.file

will generate the following output, based upon no dialog and the default rules.

```
########################################################################
#  ESQGTD Version      : 1.4.2
#  Date/Time                : 12-SEP-1996 17:17:49.49
#  File          : 202:181
########################################################################
continuation '\'
create cluster description CLUSTER_181 database WS2V14 file number 181
(
    create table description TABLE_181
   (
       COL_SEQNO_0                       seqno(0)  not null
      ,COL_AA                            shortname 'AA'
      ,COL_AB                            shortname 'AB'
      ,COL_AC                            shortname 'AC'
      ,COL_AD                            shortname 'AD'
      ,primary key ( COL_SEQNO_0)
# Number of columns for this table:    5.
   )
   ,create table description TABLE_181_B0
   (
       COL_SEQNO_0                       seqno(0)  not null
      ,COL_SEQNO_1                       seqno(1)  not null
      ,COL_BA                            shortname 'BA'
      ,COL_BB                            shortname 'BB'
      ,foreign key ( COL_SEQNO_0) references TABLE_181
      ,primary key ( COL_SEQNO_0, COL_SEQNO_1)
# Number of columns for this table:    4.
   )
   ,create table description TABLE_181_CA
   (
       COL_SEQNO_0                       seqno(0)  not null
      ,COL_SEQNO_1                       seqno(1)  not null
      ,COL_SEQNO_2                       seqno(2)  not null
      ,COL_CA                            shortname 'CA'
      ,foreign key ( COL_SEQNO_0, COL_SEQNO_1) references TABLE_181_B0
      ,primary key ( COL_SEQNO_0, COL_SEQNO_1, COL_SEQNO_2)
# Number of columns for this table:    4.
   )
   ,create table description TABLE_181_CB
   (
       COL_SEQNO_0                          seqno(0)  not null
      ,COL_SEQNO_1                          seqno(1)  not null
      ,COL_SEQNO_2                          seqno(2)  not null
      ,COL_CB                               shortname 'CB'
      ,foreign key ( COL_SEQNO_0, COL_SEQNO_1) references TABLE_181_B0
      ,primary key ( COL_SEQNO_0, COL_SEQNO_1, COL_SEQNO_2)
# Number of columns for this table:    4.
   )
   ,create table description TABLE_181_DA
   (
       COL_SEQNO_0                          seqno(0)  not null
      ,COL_SEQNO_1                          seqno(1)  not null
      ,COL_SEQNO_2                          seqno(2)  not null
      ,COL_DA                               shortname 'DA'
      ,foreign key ( COL_SEQNO_0, COL_SEQNO_1) references TABLE_181_B0
      ,primary key ( COL_SEQNO_0, COL_SEQNO_1, COL_SEQNO_2)
# Number of columns for this table:    4.
   )
);
```

Example 2: Command Line Invocation With a Systrans File

The following NATURAL DDM SYSTRANS file was used in this example:

```
*H**ANAT2201199510311127127HP_HPUX         1
*C**                                SYSTEM  CITY_GUIDE                     V S
*D01NAT2201V SYSTEM  CITY_GUIDE                      KJO                     S
*D02            19951025154500019951025154500000000001061
*D03HP_HPUX
*D040020200181
*S** 1AASTATES_ABBREVIATION             A0020 D
*S** 1ABSTATES_STATE_NAME               A0200 D
*S** 1ACSTATES_CAPITAL                  A0200 D
*S** 1ADSTATES_POPULATION               I0040
*S**P1B0CITIES                           0000
*S** 2BACITIES_CITY_NAME                A0200ND
*S** 2BBCITIES_POPULATION               I0040N
*S**M2CABUILDINGS_BUILDING_NAME         A0200N
*S**M2CBBUILDINGS_HEIGHT                I0020N
*S**M2DAPLACES_PLACE_NAME               A0200N
*S**P1X1X1-1                            A0220 S
*S***      -------- SOURCE FIELD(S) -------
*S***      CITIES_CITY_NAME(1-20)
*S***      STATES_ABBREVIATION(1-2)
*S********DDM OUTPUT TERMINATED******
*E
```

Invoke the GTD Utility as follows:

```
esqgtd -f -t systrans.file WS2V14 202 181 > output.file
```

will produce the following output:

```
#######################################################################
#  ESQGTD Version      : 1.4D.2
#  Date/Time               : 12-SEP-1996 17:18:13.58
#  File           : 202:181
#######################################################################
continuation '\'
create cluster description CLUSTER_181 database WS2V14 file number 181
(
    create table description CITY_GUIDE
   (
       COL_SEQNO_0                      seqno(0)  not null
      ,STATES_ABBREVIATION              shortname 'AA'
      ,STATES_STATE_NAME                shortname 'AB'
      ,STATES_CAPITAL                   shortname 'AC'
      ,STATES_POPULATION                shortname 'AD'
      ,primary key ( COL_SEQNO_0)
# Number of columns for this table:    5.
   )
   ,create table description CITY_GUIDE_CITIES
   (
       COL_SEQNO_0                      seqno(0)  not null
      ,COL_SEQNO_1                      seqno(1)  not null
      ,CITIES_CITY_NAME                 shortname 'BA'
      ,CITIES_POPULATION                shortname 'BB'
      ,foreign key ( COL_SEQNO_0) references CITY_GUIDE
      ,primary key ( COL_SEQNO_0, COL_SEQNO_1)
# Number of columns for this table:    4.
   )
   ,create table description CITY_GUIDE_BUILDINGS_BUILDING_NAME
   (
       COL_SEQNO_0                      seqno(0)  not null
      ,COL_SEQNO_1                      seqno(1)  not null
      ,COL_SEQNO_2                      seqno(2)  not null
      ,BUILDINGS_BUILDING_NAME          shortname 'CA'
      ,foreign key ( COL_SEQNO_0, COL_SEQNO_1) references CITY_GUIDE_CITIES
      ,primary key ( COL_SEQNO_0, COL_SEQNO_1, COL_SEQNO_2)
# Number of columns for this table:    4.
   )
   ,create table description CITY_GUIDE_BUILDINGS_HEIGHT
   (
       COL_SEQNO_0                      seqno(0)  not null
      ,COL_SEQNO_1                      seqno(1)  not null
      ,COL_SEQNO_2                      seqno(2)  not null
      ,BUILDINGS_HEIGHT                 shortname 'CB'
      ,foreign key ( COL_SEQNO_0, COL_SEQNO_1) references CITY_GUIDE_CITIES
      ,primary key ( COL_SEQNO_0, COL_SEQNO_1, COL_SEQNO_2)
# Number of columns for this table:    4.
   )
   ,create table description CITY_GUIDE_PLACES_PLACE_NAME
   (
       COL_SEQNO_0                      seqno(0)  not null
      ,COL_SEQNO_1                      seqno(1)  not null
      ,COL_SEQNO_2                      seqno(2)  not null
      ,PLACES_PLACE_NAME                shortname 'DA'
      ,foreign key ( COL_SEQNO_0, COL_SEQNO_1) references CITY_GUIDE_CITIES
      ,primary key ( COL_SEQNO_0, COL_SEQNO_1, COL_SEQNO_2)
# Number of columns for this table:    4.
   )
);
```

Example 3: Input File Usage (Default Rules)

The following input file was used in this example:

```
DATABASE 202 WS2V14
FILE 181
```

Invoking the GTD Utility as follows:

```
esqgtd < input.file > output.file
```

will produce in this case, the same output as example #1.

Example 4: Input file with Natural Systrans DDM File

In this example, the following input file was used. It references the same NATURAL SYSTRANS DDM file as above:

```
DATABASE 202 WS2V14
FILE 181
SYSTRANS systrans.file
```

Invoking the GTD Utility as follows:

```
esqgtd < input.file > output.file
```

will produce in this case, the same output as example #2.

Example 5: Input File With Directives

In this example the following input file was used:

```
DATABASE 202 WS2V14
FILE 181
B0 SUBTABLE LEVEL_1_TABLE
CA,CB SUBTABLE LEVEL_2_TABLE
SUPPRESS DA
```

Invoking the GTD Utility as follows:

```
esqgtd < input.file > output.file
```

will produce in this case, the following output:

```
#######################################################################

#  ESQGTD Version      : 1.4D.2

#  Date/Time           : 12-SEP-1996 17:18:24.85

#  File            : 202:181
#######################################################################

continuation '\'
create cluster description CLUSTER_181 database WS2V14 file number 181

(

    create table description TABLE_181

   (

       COL_SEQNO_0                         seqno(0)  not null

      ,COL_AA                              shortname 'AA'

      ,COL_AB                              shortname 'AB'

      ,COL_AC                              shortname 'AC'

      ,COL_AD                              shortname 'AD'

      ,primary key ( COL_SEQNO_0)

# Number of columns for this table:    5.

   )

   ,create table description LEVEL_1_TABLE

   (

       COL_SEQNO_0                         seqno(0)  not null

      ,COL_SEQNO_1                         seqno(1)  not null

      ,COL_BA                              shortname 'BA'
      ,COL_BB                           shortname 'BB'
      ,foreign key ( COL_SEQNO_0) references TABLE_181
      ,primary key ( COL_SEQNO_0, COL_SEQNO_1)
# Number of columns for this table:    4.
   )
   ,create table description LEVEL_2_TABLE
   (
       COL_SEQNO_0                      seqno(0)  not null
      ,COL_SEQNO_1                      seqno(1)  not null
      ,COL_SEQNO_2                      seqno(2)  not null
      ,COL_CA                           shortname 'CA'
      ,COL_CB                           shortname 'CB'
      ,foreign key ( COL_SEQNO_0, COL_SEQNO_1) references LEVEL_1_TABLE
      ,primary key ( COL_SEQNO_0, COL_SEQNO_1, COL_SEQNO_2)
# Number of columns for this table:    5.
   )
);
```

The ESQSHOW Utility

Target Users

ESQSHOW has been provided to allow users the ability to easily determine what server they are currently defaulted to, as well as what other servers are available to them either locally or remotely.

Overview

The ESQSHOW Utility is used to display the current default server as well as all locally defined servers and remote servers defined in ADABAS SQL Server's routing file "esq_routing.dat". The ESQSHOW Utility is provided via a command line interface as well as part of the ESQSET Utility discussed later in this chapter.

Invoking the ESQSHOW utility

The command to verify the current default server is:

esqshow

The following output will be displayed:

Server name
The name of the current default server. A default server is established either during server generation or via the ESQSET Utility.

Description
A description assigned to the server during server generation.

Security
Whether or not the server has the security option turned on. Security is set during server generation.

Environment
Indicates whether the server is local or remote.

Server routing
Indicates the default communication method the client will use to communicate with the server. The default is always LINKED-IN, unless specified otherwise in the ADABAS SQL Server routing file "esq_routing.dat". If the server is remote, LINKED-IN may not be used.

Catalog files
The ADABAS database number and file numbers assigned to the server's catalog files.

Error files
The ADABAS database number and file number assigned to the server's error file.

Options Available

The ESQSHOW Utility supports the −a option. This option provides all of the information listed above as well as information pertaining to all other local and remote servers defined in the ADABAS SQL Server’s routing file “esq_routing.dat”.

In the case of local servers, the ADABAS database number, catalog and error file numbers and server description are displayed. Also if defined in the ADABAS SQL Server’s routing file, the default communication method is displayed.

In the case of remote servers, the default communication method and node on which the server exists are displayed, as documented in the ADABAS SQL Server’s routing file.

The ESQSET Utility

Target Users

The ESQSET utility is provided to users who wish to establish a default server for communication.

Overview

The ESQSET Utility consists of both a command line interface and GUI tool. The ESQSET Utility is used to establish a default ADABAS SQL Server to communicate with in your environment. When an SQL CONNECT statement is issued without specifying the server parameter, you will be connecting to the default server. Any default server setting can be overridden by explicitly setting the server parameter in the SQL CONNECT statement.

This default server setting will remain in affect until it is changed again by running the ESQSET Utility. The default server setting will also remain even if the user logs out and logs back in again.

Invoking the ESQSET Utility

The ESQSET Utility is available via a command line interface and GUI tool.

ESQSET Command Line Interface

To invoke the command line interface, enter the following command at a DOS prompt:

```
esqset.bat  servername
```

This will set the default server for this session as well as any Windows NT sessions you open after running this command. PLEASE NOTE: Any Windows NT sessions already open will not be affected by this action.

Once you have executed the above command, you can issue the following command to see the server you will now be communicating with by default:

```
esqshow
```

ESQSET GUI TOOL

The ESQSET GUI Tool may be invoked via Windows NT Explorer or command line:

Windows NT Explorer: Double click on “ESQSetGUI.exe” found in the ..\bin directory of the current version of ESQ.

Command line: Since the ..\bin directory of the current version of ESQ is in the PATH, simply enter **esqsetgui**

Upon starting this tool, you will be presented with a server list displaying the current default server as well as all locally defined servers and remote servers defined in the ADABAS SQL Server’s routing file “esq_routing.dat”.

The following information is displayed in the server list:

SERVER NAME A list of all local servers and remote servers defined in the ADABAS SQL Server routing file “esq_routing.dat”.

DEFAULT Indicates which server (if any) has been set to be the default server in the case of an SQL CONNECT statement in which no server name is explicitly given. A default server may be established either via the user running the ESQSET Utility or the administrator establishing a system wide default during server generation. Setting a default server via ESQSET overrides any system wide default server setting.

STATUS This column displays the current status of each server in the list. The following statuses apply:

Active The server is defined on the local node and has been made active via esqstart.

Inactive The server is defined on the local node but is not active.

Unknown The server is a remote server defined in the ADABAS SQL Server routing file “esq_routing.dat”. Whether it is active or not is unknown.

Removed The server is defined on the local node but is currently in the process of being removed.

NODE Displays the node name on which the server is defined.

COMM TYPE — Displays the default method of communication to be used when starting a session with the associated server. The communication type will default to “LINKED-IN” unless specified otherwise in the ADABAS SQL Server routing file “esq_routing.dat”. At time of publication, the communication types “CSCI” and “LINKED-IN” are supported. The use of CSCI requires the ENTIRE NET-WORK product.

DBID — The database number of the ADABAS database on which this server’s catalog and error files are defined. A DBID=0 signifies a remote server.

CATALOGS — The ADABAS file numbers associated with the Server’s catalog files. A file number = 0 signifies a remote server.

ERROR FILE — The ADABAS file number associated with the Server’s error file.

SECURITY — This column indicates whether or not the server was generated with the SECURITY option. The following values apply:

ON – The local server was generated with SECURITY.

OFF – The local server was not generated with SECURITY.

UNKNOWN — The server is remote and therefore the presence of SECURITY is unknown to this application.

DESCRIPTION — This column displays the description of the server as defined during server generation.

Buttons Available

D – Set the selected server name as the default server.

? – Display program information, version number and copyright

➧ ? – Display help for clicked on buttons, menus and windows.

Menu Options Available

Server Menu

Set Server As Default	Set the selected server name as the default server.
Exit	Exit the ESQSET GUI Tool.

View Menu

Toolbar	Show or hide the toolbar.
Status Bar	Show or hide the status bar.
Refresh	Refresh the screen.

Help Menu

Help Topics	Display help topics for the ESQSetGUI tool.
About ESQSetGUI	Display program information, version number and copyright.

LOGGING FACILITIES

Introduction

Numerous actions and operations performed by the ADABAS SQL Server can be logged. The act of logging records that a particular event has occurred and also records additional information associated with the event. Events are recorded chronologically. This record of events can then be examined at a later date. Different types of events can be recorded simultaneously and the types of events to be logged can be specified as required.

Logging can be activated before executing the client application or before starting the server. Logging for the server can be activated without having to interrupt the server by activating logging on the client side. The client stub (ESQLNK) informs the server about the additional logging parameters. These additional logging parameters are valid for the current client session only.

Using the logging facilities will affect the overall system performance. In addition, certain logging facilities are extremely detailed and liable to produce very large amounts of information.

Logging can provide information necessary for performance analysis, problem diagnosis and system access monitoring.

For example, the Explain Logging Facility provides detailed information about the access paths used when executing DML statements. Using this information, it may be possible to construct statements which are more efficient, or to introduce appropriate indices to improve performance. The Client/Server Communication Logging can be used to investigate any problems experienced in a particular client/server link. Finally, the Security Logging can be used to monitor changing privileges.

Logging is activated by setting the environment variable ESQLOG appropriately. By setting this variable, a particular type of logging with options, if applicable, can be activated. Furthermore, combinations of types of logging can be specified. The character “*” activates the complete logging.

Logging Facilities Overview

The following tables outline the types of logging available and the string setting for ESQLOG required to activate it:

Logging Facilities Activated by an Environment Variable (ESQLOG)

Client/Server Characteristics Logging	MODE
Brief SQL Command Logging	
Client and Server side	CMD
Client side only	CCMD
Server side only	SCMD
Static SQL Command Logging	
80-Character Block Format	STATEMENT
Continuous Stream Format	STATEMENT=STREAM
Dynamic SQL Command Logging	
80-Character Block Format	DYNHVS
Continuous Stream Format	DYNHVS=STREAM
Client/Server Communication Logging	CSC
Server Session Logging	SES
Schema Identifier Logging	SCHEMAIDENT
User Exit Logging	USEREXIT
Elapsed Time Logging	
without component time	TIME
with component time	TIME = ALL
Sort Logging	SORT
ADABAS Utility Logging	ADU
Security Logging	
User authentication check logging	AUTH
Simple statement logging	SEC
ALL resultant actions logged	SEC=ALL

ESQLNK Call Logging	ENTRY
Explain Logging	EXPL

Logging Facilities Activated by an Environment Variable (VOLOG)

System Call Logging	ON

Logging Facilities Activated by Parameter File Specification

ADABAS Command Logging

How to Activate Logging

With the exception of the ADABAS Command Logging (which is controlled via the ADABAS SQL Parameter file and is discussed later in this chapter), all other logging is controlled via the ESQLOG/VOLOG environment variables.

Logging is activated by assigning a string to ESQLOG/VOLOG containing one or more keywords, specifying the entity for logging.

Example:

`set ESQLOG=”CMD,CSC”`	turns on the brief SQL command logging and the client/server communication logging.
`set ESQLOG=”*,-EXPL”`	turns on the logging without Explain Logging.
`set VOLOG=ON`	turns on the system call logging.

Logging Output

The Logging Facility logs information from either the client or the server standpoint, or both, depending on the value assigned to ESQLOG. Therefore, the destination of the logging information will depend on who is performing the work:

- the client or
- the server

and the client/server configuration being used:

- LINKED-IN Mode
  When logging is used in LINKED-IN mode, the work performed by both the client and the server is logged to the standard output of the client.
- Client/Server Mode
  When logging is used in Client/Server mode, the work performed by the client is logged to the client's standard output and the work performed by the server is logged to the server log file esqsrv.log.

Logging information that is being directed to standard output of the client is in the form of a text file. This might conflict with the overlying application program if it also directs its output there. Therefore logging output may be redirected to a system file. This is achieved by setting the ESQLOG/VOLOG environment variable to REDIRECT.

Example:

```
set ESQLOG=”CMD,REDIRECT”
set VOLOG=”REDIRECT”
```

This will result in only one line being written to the standard output. This line contains the name and location of the file to which the rest of the logging output is written.

GUI Utility Logging

To activate logging for GUI Utilities such as the BASIC Interactive Facilities GUI tool, the ESQLOG/VOLOG environment variables must be set prior to starting the GUI Utility. The logging output will then be written to a text file located in the directory pointed to by the environment variable “SAGTMP”. The text file name generated will be “ESQSTDOUT_<user id>_<process_id>”.

Example:

```
set ESQLOG=”CMD”
basic                       name of the BASIC Interactive Facilities GUI Tool
```

Logging Facilities Reference

The following sections describe all types of logging. You will find information about:

- actions and information to be logged
- examples of how to activate and run logging

Most of the following examples were executed in LINKED-IN mode. This has the advantage that the server logging also appears on the client side. For examples executed in Client/Server mode, the environment variable ESQLOG has to be used on the server side as well in order to get the server’s logging output.

Client/Server Characteristics Logging

This logging gives information about:

- Server, client stub, precompiler version
- Trace availability
- Thread mode (single/multi user linked-in)
- Application was linked/loaded with esq.o, ESQLNK or ESQTHS

Example:

```
set ESQLOG=mode
esqint
%ESQINT-I-STARTED, 16-APR-1998 10:25:14, Version 1.4/2.13 (INTEL-80386/WINDOWS
NT)
ESQ User: dba
Password: %ESQRTS-I-ESQINFO, ESQLOG is enabled for ’MODE’
%ESQRTS-I-ESQINFO, MT/TR trace is NOT available
%ESQRTS-I-ESQINFO, ESQ application was linked/loaded with ESQLNK V1.4.2.13
%ESQRTS-I-ESQINFO, session/process memory is reliable between requests
%ESQRTS-I-ESQINFO, ESQLNK/ESQSRV and ESQTHS don’t share static data
%ESQRTS-I-ESQINFO, ESQ application was loaded with ESQTHS V1.4.2.13
%ESQRTS-I-ESQINFO, Th0: Single-user linked-in mode
%ESQINT-I-CONNECT, Server DEMOSERV connected successfully
esqint: select * from information_schema.tables;



 TABLE_SCHEMA                     TABLE_NAME                       TABLE_TYPE


 DBA_SCHEMA                       SERVER_INFO                      VIEW
 DBA_SCHEMA                       INFORMATION_SCHEMA_CATALOG_NAME  VIEW
 DBA_SCHEMA                       SCHEMATA                         VIEW
 DBA_SCHEMA                       CLUSTERS                         VIEW
 DBA_SCHEMA                       TABLES                           VIEW

...
esqint: q

%ESQINT-I-TERMINATED, 16-APR-1998 10:28:11
```

Brief SQL Command Logging

The brief SQL Command Logging is available on both, client and server side. The log lines are produced at the end of each SQL request so that the response code is visible.

- Following logging lines are produced on client side:

```
%ESQRTS-I-ESQINFO, Clt: PREPARE   cnt=    7/   3    rsp=0
                        |                 |    |        |
                        |                 |    |        |
                        |                 |    |        response code
                        |                 |    |
                              |                 |    C/S communication count
                        |                 |
                        |                 SQL request count
                        |
                        SQL statement
```

- Following logging lines are produced on server side:

```
30-NOV 14:00:07.58 Th0: PREPARE  -cnt=  7/    2 usr=wsesq7:JOHN   rsp=0
                        |        |       |    |     |      |          |
                        |        |       |    |     |      |          |
                        |        |       |    |     |      |          response
                        |        |       |    |     |      |          code
                        |        |       |    |     |      |
                        |        |       |    |     |      CONNECT client
                        |        |       |    |     |      user ID
                        |        |       |    |     |
                        |        |       |    |     Client node name
                        |        |       |    |
                        |        |       |     C/S communication count
                        |        |       |
                        |        |       SQL request count
                        |        |
                        |        + : Meta program found in the catalog buffer
                        |        - : Meta program loaded into the catalog
                                     buff.
                        |        = : Meta program generated and loaded
                        |
                        SQL statement
```

- Especially for the SQL statement FETCH (with MULTIFETCH) following logging lines are produced:

  ```
  30-NOV 14:00:08.37 Th0: mFETCH (rf=16) +cnt=  12/   7 usr=wsesq7:JOHN rsp=0
  ```

  MULTIFETCH was executed on server side. 16 records were fetched and transported to the client. “rf” stands for “records fetched”.

  ```
  %ESQRTS-I-ESQINFO, Clt: mFETCH (rf=16)  cnt=  12/   8 rsp=0
  ```

  MULTIFETCH was executed on client side. 16 records were received by the client. First record is offered to the application.

  ```
  %ESQRTS-I-ESQINFO, Clt: sFETCH (rp=2)   cnt=  13/   8 rsp=0
  ```

  Single FETCH was executed on client side only. The second record on client side is offered to the application. “rp” stands for “record position”.

Example:

```
% set ESQLOG=cmd
esqint
%ESQINT-I-STARTED, 16-APR-1998 10:29:35, Version 1.4/2.13 (INTEL-80386/WINDOWS
NT)
ESQ User: john
Password: %ESQRTS-I-ESQINFO, ESQLOG is enabled for ’CMD’
16-APR 10:29:37.65 Th0: CONNECT         cnt=   1/   1 usr=SASME:JOHN rsp=0
%ESQRTS-I-ESQINFO, Clt: CONNECT         cnt=   1/   2 rsp=0
%ESQINT-I-CONNECT, Server DEMOSERV connected successfully
esqint: select * from information_schema.tables;
16-APR 10:30:02.54 Th0: PREPARE        -cnt=   7/   2 usr=SASME:JOHN rsp=0
%ESQRTS-I-ESQINFO, Clt: PREPARE         cnt=   7/   3 rsp=0
16-APR 10:30:02.55 Th0: DESCRIBE       -cnt=   8/   3 usr=SASME:JOHN rsp=0
%ESQRTS-I-ESQINFO, Clt: DESCRIBE        cnt=   8/   4 rsp=0
16-APR 10:30:02.56 Th0: DESCRIBE       -cnt=   9/   4 usr=SASME:JOHN rsp=0
%ESQRTS-I-ESQINFO, Clt: DESCRIBE        cnt=   9/   5 rsp=0
16-APR 10:30:02.57 Th0: DYN DECL CURSO -cnt=  10/   5 usr=SASME:JOHN rsp=0
%ESQRTS-I-ESQINFO, Clt: DYN DECL CURSO  cnt=  10/   6 rsp=0
16-APR 10:30:02.59 Th0: OPEN CURSOR    -cnt=  11/   6 usr=SASME:JOHN rsp=0
%ESQRTS-I-ESQINFO, Clt: OPEN CURSOR     cnt=  11/   7 rsp=0
16-APR 10:30:03.26 Th0: mFETCH (rf=16)  cnt=  12/   7 usr=SASME:JOHN rsp=0
%ESQRTS-I-ESQINFO, Clt: mFETCH (rf=16)  cnt=  12/   8 rsp=0
%ESQRTS-I-ESQINFO, Clt: sFETCH (rp=2)   cnt=  13/   8 rsp=0
%ESQRTS-I-ESQINFO, Clt: sFETCH (rp=3)   cnt=  14/   8 rsp=0
%ESQRTS-I-ESQINFO, Clt: sFETCH (rp=4)   cnt=  15/   8 rsp=0
%ESQRTS-I-ESQINFO, Clt: sFETCH (rp=5)   cnt=  16/   8 rsp=0
```

```
%ESQRTS-I-ESQINFO, Clt: sFETCH (rp=6)   cnt=  17/   8 rsp=0
%ESQRTS-I-ESQINFO, Clt: sFETCH (rp=7)   cnt=  18/   8 rsp=0
%ESQRTS-I-ESQINFO, Clt: sFETCH (rp=8)   cnt=  19/   8 rsp=0
%ESQRTS-I-ESQINFO, Clt: sFETCH (rp=9)   cnt=  20/   8 rsp=0
%ESQRTS-I-ESQINFO, Clt: sFETCH (rp=10)  cnt=  21/   8 rsp=0
%ESQRTS-I-ESQINFO, Clt: sFETCH (rp=11)  cnt=  22/   8 rsp=0
%ESQRTS-I-ESQINFO, Clt: sFETCH (rp=12)  cnt=  23/   8 rsp=0
%ESQRTS-I-ESQINFO, Clt: sFETCH (rp=13)  cnt=  24/   8 rsp=0
%ESQRTS-I-ESQINFO, Clt: sFETCH (rp=14)  cnt=  25/   8 rsp=0
%ESQRTS-I-ESQINFO, Clt: sFETCH (rp=15)  cnt=  26/   8 rsp=0
%ESQRTS-I-ESQINFO, Clt: sFETCH (rp=16)  cnt=  27/   8 rsp=0
16-APR 10:30:03.43 Th0: mFETCH (rf=6)   cnt=  28/   8 usr=SASME:JOHN rsp=100
%ESQRTS-I-ESQINFO, Clt: mFETCH (rf=6)   cnt=  28/   9 rsp=0
%ESQRTS-I-ESQINFO, Clt: sFETCH (rp=2)   cnt=  29/   9 rsp=0
%ESQRTS-I-ESQINFO, Clt: sFETCH (rp=3)   cnt=  30/   9 rsp=0
%ESQRTS-I-ESQINFO, Clt: sFETCH (rp=4)   cnt=  31/   9 rsp=0

 TABLE_SCHEMA                     TABLE_NAME                       TABLE_TYPE

 INFORMATION_SCHEMA               SERVER_INFO                      VIEW
 INFORMATION_SCHEMA               INFORMATION_SCHEMA_CATALOG_NAME  VIEW
 INFORMATION_SCHEMA               SCHEMATA                         VIEW
 INFORMATION_SCHEMA               CLUSTERS                         VIEW
 INFORMATION_SCHEMA               TABLES                           VIEW
 INFORMATION_SCHEMA               BASE_TABLES                      VIEW
 INFORMATION_SCHEMA               VIEWS                            VIEW
 INFORMATION_SCHEMA               COLUMNS                          VIEW
 INFORMATION_SCHEMA               TABLE_PRIVILEGES                 VIEW
 INFORMATION_SCHEMA               COLUMN_PRIVILEGES                VIEW
 INFORMATION_SCHEMA               TABLE_CONSTRAINTS                VIEW
 INFORMATION_SCHEMA               TABLE_INDEXES                    VIEW
 INFORMATION_SCHEMA               ALL_TABLE_INDEX_ELEMENTS         VIEW
 INFORMATION_SCHEMA               KEY_COLUMN_USAGE                 VIEW
 INFORMATION_SCHEMA               VIEW_TABLE_USAGE                 VIEW
 INFORMATION_SCHEMA               VIEW_COLUMN_USAGE                VIEW
 INFORMATION_SCHEMA               CONSTRAINT_TABLE_USAGE           VIEW
 INFORMATION_SCHEMA               CONSTRAINT_COLUMN_USAGE          VIEW
 INFORMATION_SCHEMA               REFERENTIAL_CONSTRAINTS          VIEW
 INFORMATION_SCHEMA               DATABASES                        VIEW

output (col 1-79 of 107): q
esqint: q
16-APR 10:30:08.96 Th0: CLOSE CURSOR    cnt=  32/   9 usr=SASME:JOHN rsp=0
%ESQRTS-I-ESQINFO, Clt: CLOSE CURSOR    cnt=  32/  10 rsp=0
16-APR 10:30:09.00 Th0: DISCONNECT      cnt=  33/  10 usr=SASME:JOHN rsp=0
%ESQRTS-I-ESQINFO, Clt: DISCONNECT      cnt=  33/  11 rsp=0
%ESQINT-I-TERMINATED, 16-APR-1998 10:30:09
%ESQRTS-I-ESQINFO, Clt: DISCONNECT ALL  cnt=  34/  11 rsp=0
```

Static SQL Command-String Logging

The Static SQL Command String Logging produces log lines containing the complete SQL statements. This logging type is available for runtime and for precompilation time. Two logging formats are supported:

- 80-character Block Format (setenv ESQLOG statement)
- Continuous Stream Format (setenv ESQLOG statement=stream)

Example:

```
% set ESQLOG=statement
% esqint
%ESQINT-I-STARTED, 16-APR-1998 11:36:37, Version 1.4/2.13 (INTEL-80386/WINDOWS
NT)
ESQ User: john
Password: %ESQRTS-I-ESQINFO, ESQLOG is enabled for ’STATEMENT’
%ESQRTS-I-ESQINFO, CONNECT TO DEFAULT USER : ”&sql_user[0]” CHAR-
ACTER(,32,0,0,8
_01 ,33)
%ESQINT-I-CONNECT, Server DEMOSERV connected successfully
esqint: select * from information_schema.tables;
%ESQRTS-I-ESQINFO, PREPARE : ”&sql_id[0]” CHARACTER(,32,0,0,46,33) FROM :
”&sma
_01 ll_sql_statement[0]” CHARACTER(,128,0,0,47,129)
%ESQRTS-I-ESQINFO, DESCRIBE : ”&sql_id[0]” CHARACTER(,32,0,0,55,33) INTO INPUT
_01 : ”if_sql_input_sqlda” ( ”if_sql_input_sqlda” POINTER ( ,0,0,0,0))
%ESQRTS-I-ESQINFO, DESCRIBE : ”&sql_id[0]” CHARACTER(,32,0,0,56,33) INTO OUT-
PUT
_01  : ”if_sql_output_sqlda” ( ”if_sql_output_sqlda” POINTER ( ,0,0,0,1))
%ESQRTS-I-ESQINFO, DECLARE : ”&sql_cursor[0]” CHARACTER(,18,0,0,87,19) CURSOR
F
_01 OR : ”&sql_id[0]” CHARACTER(,32,0,0,88,33)
%ESQRTS-I-ESQINFO, OPEN : ”&sql_cursor[0]” CHARACTER(,18,0,0,89,19)
%ESQRTS-I-ESQINFO, FETCH : ”&sql_cursor[0]” CHARACTER(,18,0,0,91,19) USING
DESC
_01 RIPTOR : ”if_sql_output_sqlda” ( ”if_sql_output_sqlda” POINTER (
,0,0,0,1))
%ESQRTS-I-ESQINFO, FETCH : ”&sql_cursor[0]” CHARACTER(,18,0,0,91,19) USING
DESC
_01 RIPTOR : ”if_sql_output_sqlda” ( ”if_sql_output_sqlda” POINTER (
,0,0,0,1))
%ESQRTS-I-ESQINFO, FETCH : ”&sql_cursor[0]” CHARACTER(,18,0,0,91,19) USING
DESC
```

```
_01 RIPTOR : ”if_sql_output_sqlda” ( ”if_sql_output_sqlda” POINTER (
,0,0,0,1))
%ESQRTS-I-ESQINFO, FETCH : ”&sql_cursor[0]” CHARACTER(,18,0,0,91,19) USING
DESC
_01 RIPTOR : ”if_sql_output_sqlda” ( ”if_sql_output_sqlda” POINTER (
,0,0,0,1))
%ESQRTS-I-ESQINFO, FETCH : ”&sql_cursor[0]” CHARACTER(,18,0,0,91,19) USING
DESC
_01 RIPTOR : ”if_sql_output_sqlda” ( ”if_sql_output_sqlda” POINTER (
,0,0,0,1))
%ESQRTS-I-ESQINFO, FETCH : ”&sql_cursor[0]” CHARACTER(,18,0,0,91,19) USING
DESC
_01 RIPTOR : ”if_sql_output_sqlda” ( ”if_sql_output_sqlda” POINTER (
,0,0,0,1))
%ESQRTS-I-ESQINFO, FETCH : ”&sql_cursor[0]” CHARACTER(,18,0,0,91,19) USING
DESC
_01 RIPTOR : ”if_sql_output_sqlda” ( ”if_sql_output_sqlda” POINTER (
,0,0,0,1))
%ESQRTS-I-ESQINFO, FETCH : ”&sql_cursor[0]” CHARACTER(,18,0,0,91,19) USING
DESC
_01 RIPTOR : ”if_sql_output_sqlda” ( ”if_sql_output_sqlda” POINTER (
,0,0,0,1))
%ESQRTS-I-ESQINFO, FETCH : ”&sql_cursor[0]” CHARACTER(,18,0,0,91,19) USING
DESC
_01 RIPTOR : ”if_sql_output_sqlda” ( ”if_sql_output_sqlda” POINTER (
,0,0,0,1))
%ESQRTS-I-ESQINFO, FETCH : ”&sql_cursor[0]” CHARACTER(,18,0,0,91,19) USING
DESC
_01 RIPTOR : ”if_sql_output_sqlda” ( ”if_sql_output_sqlda” POINTER (
,0,0,0,1))
%ESQRTS-I-ESQINFO, FETCH : ”&sql_cursor[0]” CHARACTER(,18,0,0,91,19) USING
DESC
_01 RIPTOR : ”if_sql_output_sqlda” ( ”if_sql_output_sqlda” POINTER (
,0,0,0,1))
%ESQRTS-I-ESQINFO, FETCH : ”&sql_cursor[0]” CHARACTER(,18,0,0,91,19) USING
DESC
_01 RIPTOR : ”if_sql_output_sqlda” ( ”if_sql_output_sqlda” POINTER (
,0,0,0,1))
%ESQRTS-I-ESQINFO, FETCH : ”&sql_cursor[0]” CHARACTER(,18,0,0,91,19) USING
DESC
_01 RIPTOR : ”if_sql_output_sqlda” ( ”if_sql_output_sqlda” POINTER (
,0,0,0,1))
%ESQRTS-I-ESQINFO, FETCH : ”&sql_cursor[0]” CHARACTER(,18,0,0,91,19) USING
DESC
_01 RIPTOR : ”if_sql_output_sqlda” ( ”if_sql_output_sqlda” POINTER (
,0,0,0,1))
```

```
%ESQRTS-I-ESQINFO, FETCH : ”&sql_cursor[0]” CHARACTER(,18,0,0,91,19) USING
DESC
_01 RIPTOR : ”if_sql_output_sqlda” ( ”if_sql_output_sqlda” POINTER (
,0,0,0,1))
%ESQRTS-I-ESQINFO, FETCH : ”&sql_cursor[0]” CHARACTER(,18,0,0,91,19) USING
DESC
_01 RIPTOR : ”if_sql_output_sqlda” ( ”if_sql_output_sqlda” POINTER (
,0,0,0,1))
%ESQRTS-I-ESQINFO, FETCH : ”&sql_cursor[0]” CHARACTER(,18,0,0,91,19) USING
DESC
_01 RIPTOR : ”if_sql_output_sqlda” ( ”if_sql_output_sqlda” POINTER (
,0,0,0,1))
%ESQRTS-I-ESQINFO, FETCH : ”&sql_cursor[0]” CHARACTER(,18,0,0,91,19) USING
DESC
_01 RIPTOR : ”if_sql_output_sqlda” ( ”if_sql_output_sqlda” POINTER (
,0,0,0,1))
%ESQRTS-I-ESQINFO, FETCH : ”&sql_cursor[0]” CHARACTER(,18,0,0,91,19) USING
DESC
_01 RIPTOR : ”if_sql_output_sqlda” ( ”if_sql_output_sqlda” POINTER (
,0,0,0,1))
%ESQRTS-I-ESQINFO, FETCH : ”&sql_cursor[0]” CHARACTER(,18,0,0,91,19) USING
DESC
_01 RIPTOR : ”if_sql_output_sqlda” ( ”if_sql_output_sqlda” POINTER (
,0,0,0,1))

 TABLE_SCHEMA                     TABLE_NAME                       TABLE_TYPE

 INFORMATION_SCHEMA               SERVER_INFO                      VIEW
 INFORMATION_SCHEMA               INFORMATION_SCHEMA_CATALOG_NAME  VIEW
 INFORMATION_SCHEMA               SCHEMATA                         VIEW
 INFORMATION_SCHEMA               CLUSTERS                         VIEW
 INFORMATION_SCHEMA               TABLES                           VIEW
 INFORMATION_SCHEMA               BASE_TABLES                      VIEW
 INFORMATION_SCHEMA               VIEWS                            VIEW
 INFORMATION_SCHEMA               COLUMNS                          VIEW
 INFORMATION_SCHEMA               TABLE_PRIVILEGES                 VIEW
 INFORMATION_SCHEMA               COLUMN_PRIVILEGES                VIEW
 INFORMATION_SCHEMA               TABLE_CONSTRAINTS                VIEW
 INFORMATION_SCHEMA               TABLE_INDEXES                    VIEW
 INFORMATION_SCHEMA               ALL_TABLE_INDEX_ELEMENTS         VIEW
 INFORMATION_SCHEMA               KEY_COLUMN_USAGE                 VIEW
 INFORMATION_SCHEMA               VIEW_TABLE_USAGE                 VIEW
 INFORMATION_SCHEMA               VIEW_COLUMN_USAGE                VIEW
 INFORMATION_SCHEMA               CONSTRAINT_TABLE_USAGE           VIEW
 INFORMATION_SCHEMA               CONSTRAINT_COLUMN_USAGE          VIEW
 INFORMATION_SCHEMA               REFERENTIAL_CONSTRAINTS          VIEW
 INFORMATION_SCHEMA               DATABASES                        VIEW

output (col 1-79 of 107): q
esqint: q
%ESQRTS-I-ESQINFO, CLOSE : ”&sql_cursor[0]” CHARACTER(,18,0,0,90,19)
%ESQRTS-I-ESQINFO, DISCONNECT CURRENT
%ESQINT-I-TERMINATED, 16-APR-1998 11:36:54
%ESQRTS-I-ESQINFO, DISCONNECT ALL
```

Dynamic SQL Command-String Logging

The Static SQL Command String Logging produces log lines containing the dynamic SQL statements passed on to a PREPARE or EXECUTE IMMEDIATE statement. Two logging formats are supported:

- 80-character Block Format (setenv ESQLOG dynhvs)
- Continuous Stream Format (setenv ESQLOG dynhvs=stream)

Example:

```
% set ESQLOG=dynhvs
% esqint
%ESQINT-I-STARTED, 16-APR-1998 11:38:04, Version 1.4/2.13 (INTEL-80386/WINDOWS
NT)
ESQ User: john
Password: %ESQRTS-I-ESQINFO, ESQLOG is enabled for ’DYNHVS   ’
%ESQINT-I-CONNECT, Server DEMOSERV connected successfully
esqint: select * from information_schema.tables;
%ESQRTS-I-ESQINFO, HV01: sql_i
%ESQRTS-I-ESQINFO, HV02:  select * from information_schema.table

 TABLE_SCHEMA                     TABLE_NAME                       TABLE_TYPE

 INFORMATION_SCHEMA               SERVER_INFO                      VIEW
 INFORMATION_SCHEMA               INFORMATION_SCHEMA_CATALOG_NAME  VIEW
 INFORMATION_SCHEMA               SCHEMATA                         VIEW
 INFORMATION_SCHEMA               CLUSTERS                         VIEW
 INFORMATION_SCHEMA               TABLES                           VIEW
 INFORMATION_SCHEMA               BASE_TABLES                      VIEW
 INFORMATION_SCHEMA               VIEWS                            VIEW
 INFORMATION_SCHEMA               COLUMNS                          VIEW
 INFORMATION_SCHEMA               TABLE_PRIVILEGES                 VIEW
 INFORMATION_SCHEMA               COLUMN_PRIVILEGES                VIEW
 INFORMATION_SCHEMA               TABLE_CONSTRAINTS                VIEW
 INFORMATION_SCHEMA               TABLE_INDEXES                    VIEW
 INFORMATION_SCHEMA               ALL_TABLE_INDEX_ELEMENTS         VIEW
 INFORMATION_SCHEMA               KEY_COLUMN_USAGE                 VIEW
 INFORMATION_SCHEMA               VIEW_TABLE_USAGE                 VIEW
 INFORMATION_SCHEMA               VIEW_COLUMN_USAGE                VIEW
 INFORMATION_SCHEMA               CONSTRAINT_TABLE_USAGE           VIEW
 INFORMATION_SCHEMA               CONSTRAINT_COLUMN_USAGE          VIEW
 INFORMATION_SCHEMA               REFERENTIAL_CONSTRAINTS          VIEW
 INFORMATION_SCHEMA               DATABASES                        VIEW

output (col 1-79 of 107): q
esqint: q
%ESQINT-I-TERMINATED, 16-APR-1998 11:38:40
```

Client/Server Communication Logging

The Client/Server Communication Logging gives information about messages sent between ADABAS SQL Server client and ADABAS SQL Server server. It can be enabled on client and/or on server side and applies only to real client/server mode (not in linked-in mode):

- Following logging lines are produced on client side for one SQL request within one session:

```
%ESQRTS-I-ESQINFO, Clt: sending an OPEN to WSESQ7:ESQV14 ...
%ESQRTS-I-ESQINFO, Clt: ONS cid=0 sndl=0 rcvl=0 retl=0
%ESQRTS-I-ESQINFO, Clt: ONS cid=400842E8 sndl=0 rcvl=0 retl=0
%ESQRTS-I-ESQINFO, Clt: ... OPEN was successful
%ESQRTS-I-ESQINFO, Clt: sending a CSCI request ...
%ESQRTS-I-ESQINFO, Clt: QNS cid=400842E8 sndl=1024 rcvl=338 retl=0
%ESQRTS-I-ESQINFO, Clt: QNS cid=400842E8 sndl=1024 rcvl=338 retl=338
%ESQRTS-I-ESQINFO, Clt: ... received CSCI reply
%ESQRTS-I-ESQINFO, Clt: sending a CLOSE ...
%ESQRTS-I-ESQINFO, Clt: CNS cid=400842E8 sndl=0 rcvl=0 retl=310
%ESQRTS-I-ESQINFO, Clt: CNS cid=400842E8 sndl=0 rcvl=0 retl=310
%ESQRTS-I-ESQINFO, Clt: ... CLOSE was successful

%ESQRTS-I-ESQINFO, Clt: QNS cid=400842E8 sndl=1024 rcvl=338 retl=0
                        |||     |             |         |        |
                        |||     |             |         |        returned
                        |||     |             |         |        length
                        |||     |             |         |
                        |||     |             |         receive-buffer length
                        |||     |             |
                        |||     |             send-buffer length
                        |||     |
                        |||     connection ID
                        |||
                        ||Mode      S : synchronous
                        ||
                        |Type       N : native
                        |
                        Function    O : open
                                    Q : request
                                    C : close
```

- Following logging lines are produced on server side for one SQL request within on session:

```
01-DEC 10:47:47.51 Th2: f=ION c=1,        ,         l=0,0,0
01-DEC 10:47:52.30 Th2: waiting for a request ...
01-DEC 10:50:41.60 Th2: f=LOS c=0,grg     ,WSESQ7   l=0,8240,0
01-DEC 10:50:41.60 Th2: accept OPEN
01-DEC 10:50:41.61 Th2: f=AOS c=40030A08,grg     ,WSESQ7   l=0,8240,0
01-DEC 10:50:41.61 Th2: waiting for a request ...
01-DEC 10:50:41.63 Th2: f=LQS c=40030A08,grg     ,WSESQ7   l=0,8240,1024
01-DEC 10:50:45.89 Th2: sending the reply
01-DEC 10:50:45.90 Th2: f=PQS c=40030A08,grg     ,WSESQ7   l=338,0,0
01-DEC 10:50:45.90 Th2: waiting for a request ...
01-DEC 10:51:01.33 Th2: f=LCS c=40030A08,grg     ,WSESQ7   l=0,8240,0
01-DEC 10:51:01.33 Th2: accept CLOSE
01-DEC 10:51:01.34 Th2: f=PCS c=40030A08,grg     ,WSESQ7   l=0,8240,0
```

```
01-DEC 10:50:41.63 Th2: f=LQS c=40030A08,grg     ,WSESQ7   l=0,8240,1024
                          |||   |        |        |          | |    |
                          |||   |        |        |          | |    returned
                              |||   |        |        |          | |    length
                          |||   |        |        |          | |
                          |||   |        |        |          | receive-buffer
                          |||   |        |        |          | length
                          |||   |        |        |          |
                          |||   |        |        |          send-buffer length
                          |||   |        |        |
                          |||   |        |        client node name
                          |||   |        |
                          |||   |        OS client user-ID
                          |||   |
                          |||   connection ID
                          |||
                          ||Mode    S : synchronous
                          ||
                          |Type     O : open
                          |         Q : requet
                          |         C : close
                          |
                          Function  I : initialise
                                    L : listen
                                    A : accept open
                                    P : reply / close
```

Example:

```
% set ESQLOG=csci
% esqset DEMOSERV
% esqstart
%SAGESQ-I-ESQINFO, ESQLOG is enabled for 'CSCI'
%ESQSTART-I-STARTED, 16-APR-1998 11:41:17, Version 1.4/2.13 (INTEL-80386/WIN-
DOWS NT)
%ESQSTART-I-WAITING, Waiting for Server DEMOSERV to go up


 Server DEMOSERV successfully started

%ESQSTART-I-TERMINATED, 16-APR-1998 11:41:51

Log file of ESQ server DEMOSERV:
%ESQINI-I-ESQINFO, ESQLOG is enabled for 'CSCI'
%ESQINI-I-ESQINFO, Software AG ADABAS SQL Server starting up on IN-
TEL-80386/WINDOWS NT
%ESQINI-I-ESQINFO, Version:         1.4.2.13
%ESQINI-I-ESQINFO, Startup time:    16-APR-1998 11:41:37
%ESQINI-I-ESQINFO, Server name:     DEMOSERV on node SASME
%ESQINI-I-ESQINFO, Server type:     CSCI
%ESQINI-I-ESQINFO, Catalog files:   DB 5, Files 20-22
%ESQINI-I-ESQINFO, Server pid:      159
%ESQINI-I-ESQINFO, Server image:    G:\SAG\esq\v142d\bin\esqini.exe
%ESQINI-I-ESQINFO, Thread image:    G:\SAG\esq\v142d\bin\esqsrv.exe
%ESQINI-I-ESQINFO, Threads:         2
%ESQINI-I-ESQINFO, Sessions/Thread: 1
%ESQSRV-I-ESQINFO, ESQLOG is enabled for 'CSCI'
16-APR 11:41:38.42 Th1: initializing C/S Interface
%ESQRTS-I-ESQINFO, Th1: compile-time CSC_VERSION=1, run-time csc.ident=1
16-APR 11:41:38.43 Th1: f=ION c=1,        ,         l=0,0,0
16-APR 11:41:38.43 Server DEMOSERV Thread 1 pid 189 up and running
16-APR 11:41:38.43 Th1: waiting for a request ...
%ESQSRV-I-ESQINFO, ESQLOG is enabled for 'CSCI'
16-APR 11:41:41.43 Th2: initializing C/S Interface
%ESQRTS-I-ESQINFO, Th2: compile-time CSC_VERSION=1, run-time csc.ident=1
16-APR 11:41:41.44 Th2: f=ION c=1,        ,         l=0,0,0
16-APR 11:41:41.44 Server DEMOSERV Thread 2 pid 191 up and running
16-APR 11:41:41.44 Th2: waiting for a request ...
16-APR 11:41:51.00 Server DEMOSERV up and running
```

```
% esqint
%ESQINT-I-STARTED, 16-APR-1998 11:46:41, Version 1.4/2.13 (INTEL-80386/WINDOWS
NT)
%SAGESQ-I-ESQINFO, ESQLOG is enabled for ’CSCI’
ESQ User: dba
Password: %ESQRTS-I-ESQINFO, ESQLOG is enabled for ’CSCI’
%ESQRTS-I-ESQINFO, Clt: CSCI ident 1/1
%ESQRTS-I-ESQINFO, Clt: sending an OPEN to SASME:DEMOSERV ...
%ESQRTS-I-ESQINFO, Clt: ONN cid=0 sndl=0 rcvl=0 retl=0
%ESQRTS-I-ESQINFO, Clt: ONN cid=CE16B8 sndl=0 rcvl=0 retl=0
%ESQRTS-I-ESQINFO, Clt: ... OPEN was successful
%ESQRTS-I-ESQINFO, Clt: sending a CSCI request ...
%ESQRTS-I-ESQINFO, Clt: QNS cid=CE16B8 sndl=1274 rcvl=338 retl=0
%ESQRTS-I-ESQINFO, Clt: QNS cid=CE16B8 sndl=1274 rcvl=338 retl=338
%ESQRTS-I-ESQINFO, Clt: ... received CSCI reply
%ESQINT-I-CONNECT, Server DEMOSERV connected successfully
esqint: select * from information_schema.tables;
%ESQRTS-I-ESQINFO, Clt: sending a CSCI request ...
%ESQRTS-I-ESQINFO, Clt: QNS cid=CE16B8 sndl=432 rcvl=342 retl=338
%ESQRTS-I-ESQINFO, Clt: QNS cid=CE16B8 sndl=432 rcvl=342 retl=342
%ESQRTS-I-ESQINFO, Clt: ... received CSCI reply
%ESQRTS-I-ESQINFO, Clt: sending a CSCI request ...
%ESQRTS-I-ESQINFO, Clt: QNS cid=CE16B8 sndl=306 rcvl=1532 retl=342
%ESQRTS-I-ESQINFO, Clt: QNS cid=CE16B8 sndl=306 rcvl=1532 retl=1532
%ESQRTS-I-ESQINFO, Clt: ... received CSCI reply
%ESQRTS-I-ESQINFO, Clt: sending a CSCI request ...
%ESQRTS-I-ESQINFO, Clt: QNS cid=CE16B8 sndl=306 rcvl=1532 retl=1532
%ESQRTS-I-ESQINFO, Clt: QNS cid=CE16B8 sndl=306 rcvl=1532 retl=1532
%ESQRTS-I-ESQINFO, Clt: ... received CSCI reply
%ESQRTS-I-ESQINFO, Clt: sending a CSCI request ...
%ESQRTS-I-ESQINFO, Clt: QNS cid=CE16B8 sndl=304 rcvl=310 retl=1532
%ESQRTS-I-ESQINFO, Clt: QNS cid=CE16B8 sndl=304 rcvl=310 retl=310
%ESQRTS-I-ESQINFO, Clt: ... received CSCI reply
%ESQRTS-I-ESQINFO, Clt: sending a CSCI request ...
%ESQRTS-I-ESQINFO, Clt: QNS cid=CE16B8 sndl=264 rcvl=310 retl=310
%ESQRTS-I-ESQINFO, Clt: QNS cid=CE16B8 sndl=264 rcvl=310 retl=310
%ESQRTS-I-ESQINFO, Clt: ... received CSCI reply
%ESQRTS-I-ESQINFO, Clt: sending a CSCI request ...
%ESQRTS-I-ESQINFO, Clt: QNS cid=CE16B8 sndl=324 rcvl=2118 retl=310
%ESQRTS-I-ESQINFO, Clt: QNS cid=CE16B8 sndl=324 rcvl=2118 retl=2118
%ESQRTS-I-ESQINFO, Clt: ... received CSCI reply
%ESQRTS-I-ESQINFO, Clt: sending a CSCI request ...
%ESQRTS-I-ESQINFO, Clt: QNS cid=CE16B8 sndl=324 rcvl=2118 retl=2118
%ESQRTS-I-ESQINFO, Clt: QNS cid=CE16B8 sndl=324 rcvl=2118 retl=2118
%ESQRTS-I-ESQINFO, Clt: ... received CSCI reply
```

```
 TABLE_SCHEMA                     TABLE_NAME                       TABLE_TYPE

 DEFINITION_SCHEMA                ESQCAT1                          BASE TABLE
 DEFINITION_SCHEMA                ESQCAT2                          BASE TABLE
 DEFINITION_SCHEMA                HEADER                           VIEW
 DEFINITION_SCHEMA                SCHEMATA                         VIEW
 DEFINITION_SCHEMA                TABLES                           VIEW
 DEFINITION_SCHEMA                COLUMNS                          VIEW
 DEFINITION_SCHEMA                CONSTRNTS                        VIEW
 DEFINITION_SCHEMA                CONSTRNT_ELEMENTS                VIEW
 DEFINITION_SCHEMA                VIEW_TABLE_U                     VIEW
 DEFINITION_SCHEMA                VIEW_COLUMN_U                    VIEW
 DEFINITION_SCHEMA                COLUMN_COLUMN_U                  VIEW
 DEFINITION_SCHEMA                TABLE_PRIVILEGES                 VIEW
 DEFINITION_SCHEMA                COLUMN_PRIVILEGES                VIEW
 DEFINITION_SCHEMA                STATEMENT_TABLE_U                VIEW
 DEFINITION_SCHEMA                DATABASES                        VIEW
 DEFINITION_SCHEMA                USERS                            VIEW
 DEFINITION_SCHEMA                TABLESPACES                      VIEW
 DBA_SCHEMA                       SERVER_INFO                      VIEW
 DBA_SCHEMA                       INFORMATION_SCHEMA_CATALOG_NAME  VIEW
 DBA_SCHEMA                       SCHEMATA                         VIEW

output (col 1-79 of 107): q
esqint: q
%ESQRTS-I-ESQINFO, Clt: sending a CSCI request ...
%ESQRTS-I-ESQINFO, Clt: QNS cid=CE16B8 sndl=264 rcvl=310 retl=2118
%ESQRTS-I-ESQINFO, Clt: QNS cid=CE16B8 sndl=264 rcvl=310 retl=310
%ESQRTS-I-ESQINFO, Clt: ... received CSCI reply
%ESQRTS-I-ESQINFO, Clt: sending a CSCI request ...
%ESQRTS-I-ESQINFO, Clt: QNS cid=CE16B8 sndl=264 rcvl=310 retl=310
%ESQRTS-I-ESQINFO, Clt: QNS cid=CE16B8 sndl=264 rcvl=310 retl=310
%ESQRTS-I-ESQINFO, Clt: ... received CSCI reply
%ESQRTS-I-ESQINFO, Clt: sending a CLOSE ...
%ESQRTS-I-ESQINFO, Clt: CNS cid=CE16B8 sndl=0 rcvl=0 retl=310
%ESQRTS-I-ESQINFO, Clt: CNS cid=CE16B8 sndl=0 rcvl=0 retl=310
%ESQRTS-I-ESQINFO, Clt: ... CLOSE was successful
%ESQINT-I-TERMINATED, 16-APR-1998 11:46:58



Log file of ESQ server DEMOSERV:
%ESQINI-I-ESQINFO, ESQLOG is enabled for 'CSCI'


%ESQINI-I-ESQINFO, Software AG ADABAS SQL Server starting up on IN-
TEL-80386/WINDOWS NT
%ESQINI-I-ESQINFO, Version:         1.4.2.13
```

```
%ESQINI-I-ESQINFO, Startup time:    16-APR-1998 11:41:37
%ESQINI-I-ESQINFO, Server name:     DEMOSERV on node SASME
%ESQINI-I-ESQINFO, Server type:     CSCI
%ESQINI-I-ESQINFO, Catalog files:   DB 5, Files 20-22
%ESQINI-I-ESQINFO, Server pid:      159
%ESQINI-I-ESQINFO, Server image:    G:\SAG\esq\v142d\bin\esqini.exe
%ESQINI-I-ESQINFO, Thread image:    G:\SAG\esq\v142d\bin\esqsrv.exe
%ESQINI-I-ESQINFO, Threads:         2
%ESQINI-I-ESQINFO, Sessions/Thread: 1
%ESQSRV-I-ESQINFO, ESQLOG is enabled for 'CSCI'
16-APR 11:41:38.42 Th1: initializing C/S Interface
%ESQRTS-I-ESQINFO, Th1: compile-time CSC_VERSION=1, run-time csc.ident=1
16-APR 11:41:38.43 Th1: f=ION c=1,        ,         l=0,0,0
16-APR 11:41:38.43 Server DEMOSERV Thread 1 pid 189 up and running
16-APR 11:41:38.43 Th1: waiting for a request ...
%ESQSRV-I-ESQINFO, ESQLOG is enabled for 'CSCI'
16-APR 11:41:41.43 Th2: initializing C/S Interface
%ESQRTS-I-ESQINFO, Th2: compile-time CSC_VERSION=1, run-time csc.ident=1
16-APR 11:41:41.44 Th2: f=ION c=1,        ,         l=0,0,0
16-APR 11:41:41.44 Server DEMOSERV Thread 2 pid 191 up and running
16-APR 11:41:41.44 Th2: waiting for a request ...
16-APR 11:41:51.00 Server DEMOSERV up and running
16-APR 11:46:44.01 Th2: f=LOS c=0,XXX     ,SASME    l=0,8240,0
16-APR 11:46:44.01 Th2: accept OPEN
16-APR 11:46:44.02 Th2: f=AOS c=581920,XXX     ,SASME    l=0,8240,0
16-APR 11:46:44.02 Th2: waiting for a request ...
16-APR 11:46:44.04 Th2: f=LQS c=581920,XXX     ,SASME    l=0,8240,1274
%ESQRTS-I-ESQINFO, ESQLOG is enabled for 'CSCI'
16-APR 11:46:44.12 Th2: sending the reply
16-APR 11:46:44.14 Th2: f=PQS c=581920,XXX     ,SASME    l=338,0,0
16-APR 11:46:44.14 Th2: waiting for a request ...
16-APR 11:46:55.96 Th2: f=LQS c=581920,XXX     ,SASME    l=0,8240,432
16-APR 11:46:56.32 Th2: sending the reply
16-APR 11:46:56.33 Th2: f=PQS c=581920,XXX     ,SASME    l=342,0,0
16-APR 11:46:56.33 Th2: waiting for a request ...
16-APR 11:46:56.34 Th2: f=LQS c=581920,XXX     ,SASME    l=0,8240,306
16-APR 11:46:56.35 Th2: sending the reply
16-APR 11:46:56.36 Th2: f=PQS c=581920,XXX     ,SASME    l=1532,0,0
16-APR 11:46:56.36 Th2: waiting for a request ...
16-APR 11:46:56.37 Th2: f=LQS c=581920,XXX     ,SASME    l=0,8240,306
16-APR 11:46:56.37 Th2: sending the reply
16-APR 11:46:56.37 Th2: f=PQS c=581920,XXX     ,SASME    l=1532,0,0
16-APR 11:46:56.37 Th2: waiting for a request ...
16-APR 11:46:56.39 Th2: f=LQS c=581920,XXX     ,SASME    l=0,8240,304
16-APR 11:46:56.40 Th2: sending the reply
16-APR 11:46:56.41 Th2: f=PQS c=581920,XXX     ,SASME    l=310,0,0
```

```
16-APR 11:46:56.41 Th2: waiting for a request ...
16-APR 11:46:56.42 Th2: f=LQS c=581920,XXX     ,SASME    l=0,8240,264
16-APR 11:46:56.43 Th2: sending the reply
16-APR 11:46:56.44 Th2: f=PQS c=581920,XXX     ,SASME    l=310,0,0
16-APR 11:46:56.44 Th2: waiting for a request ...
16-APR 11:46:56.44 Th2: f=LQS c=581920,XXX     ,SASME    l=0,8240,324
16-APR 11:46:56.60 Th2: sending the reply
16-APR 11:46:56.61 Th2: f=PQS c=581920,XXX     ,SASME    l=2118,0,0
16-APR 11:46:56.61 Th2: waiting for a request ...
16-APR 11:46:56.64 Th2: f=LQS c=581920,XXX     ,SASME    l=0,8240,324
16-APR 11:46:56.75 Th2: sending the reply
16-APR 11:46:56.78 Th2: f=PQS c=581920,XXX     ,SASME    l=2118,0,0
16-APR 11:46:56.78 Th2: waiting for a request ...
16-APR 11:46:58.74 Th2: f=LQS c=581920,XXX     ,SASME    l=0,8240,264
16-APR 11:46:58.74 Th2: sending the reply
16-APR 11:46:58.75 Th2: f=PQS c=581920,XXX     ,SASME    l=310,0,0
16-APR 11:46:58.75 Th2: waiting for a request ...
16-APR 11:46:58.76 Th2: f=LQS c=581920,XXX     ,SASME    l=0,8240,264
16-APR 11:46:58.80 Th2: sending the reply
16-APR 11:46:58.81 Th2: f=PQS c=581920,XXX     ,SASME    l=310,0,0
16-APR 11:46:58.81 Th2: waiting for a request ...
16-APR 11:46:58.82 Th2: f=LCS c=581920,XXX     ,SASME    l=0,8240,0
16-APR 11:46:58.82 Th2: accept CLOSE
16-APR 11:46:58.82 Th2: f=PCS c=581920,XXX     ,SASME    l=0,8240,0
16-APR 11:46:58.82 Th2: waiting for a request ...
```

Session Logging on Server Side

Session Logging only works on server side and, only if CSCI is used, for C/S communication. It gives information about:

- Opening of a client session
- Closing of a client session
- Time-out of a client session

This logging is also switched on with the SQL command logging (CMD).

Example:

```
% set ESQLOG=ses
% esqset ESQDEMO
% esqstart
%SAGESQ-I-ESQINFO, ESQLOG is enabled for ’SES’
%ESQSTART-I-STARTED, 16-APR-1998 12:51:33, Version 1.4/2.13 (INTEL-80386/WIN-
DOWS  NT)
%ESQSTART-I-WAITING, Waiting for Server DEMOSERV to go up

 Server DEMOSERV successfully started

%ESQSTART-I-TERMINATED, 16-APR-1998 12:52:08

Log file of ESQ server DEMOSERV:
%ESQINI-I-ESQINFO, ESQLOG is enabled for ’SES’
%ESQINI-I-ESQINFO, Software AG ADABAS SQL Server starting up on IN-
TEL-80386/WINDOWS NT
%ESQINI-I-ESQINFO, Version:         1.4.2.13
%ESQINI-I-ESQINFO, Startup time:    16-APR-1998 14:07:56
%ESQINI-I-ESQINFO, Server name:     DEMOSERV on node SASME
%ESQINI-I-ESQINFO, Server type:     CSCI
%ESQINI-I-ESQINFO, Catalog files:   DB 5, Files 20-22
%ESQINI-I-ESQINFO, Server pid:      149
%ESQINI-I-ESQINFO, Server image:    G:\SAG\esq\v142d\bin\esqini.exe
%ESQINI-I-ESQINFO, Thread image:    G:\SAG\esq\v142d\bin\esqsrv.exe
%ESQINI-I-ESQINFO, Threads:         2
%ESQINI-I-ESQINFO, Sessions/Thread: 1
%ESQSRV-I-ESQINFO, ESQLOG is enabled for ’SES’
16-APR 14:07:58.54 Server DEMOSERV Thread 1 pid 151 up and running
%ESQSRV-I-ESQINFO, ESQLOG is enabled for ’SES’
16-APR 14:08:00.29 Server DEMOSERV Thread 2 pid 154 up and running
16-APR 14:08:09.83 Server DEMOSERV up and running
```

```
% esqint
%ESQINT-I-STARTED, 16-APR-1998 14:08:18, Version 1.4/2.13 (INTEL-80386/WINDOWS
NT)
%SAGESQ-I-ESQINFO, ESQLOG is enabled for 'SES'
ESQ User: john
Password: %ESQRTS-I-ESQINFO, ESQLOG is enabled for 'SES'
%ESQINT-I-CONNECT, Server DEMOSERV connected successfully
esqint: select * from information_schema.tables;

 TABLE_SCHEMA                     TABLE_NAME                       TABLE_TYPE

 INFORMATION_SCHEMA               SERVER_INFO                      VIEW
 INFORMATION_SCHEMA               INFORMATION_SCHEMA_CATALOG_NAME  VIEW
 INFORMATION_SCHEMA               SCHEMATA                         VIEW
 INFORMATION_SCHEMA               CLUSTERS                         VIEW
 INFORMATION_SCHEMA               TABLES                           VIEW
 INFORMATION_SCHEMA               BASE_TABLES                      VIEW
 INFORMATION_SCHEMA               VIEWS                            VIEW
 INFORMATION_SCHEMA               COLUMNS                          VIEW
 INFORMATION_SCHEMA               TABLE_PRIVILEGES                 VIEW
 INFORMATION_SCHEMA               COLUMN_PRIVILEGES                VIEW
 INFORMATION_SCHEMA               TABLE_CONSTRAINTS                VIEW
 INFORMATION_SCHEMA               TABLE_INDEXES                    VIEW
 INFORMATION_SCHEMA               ALL_TABLE_INDEX_ELEMENTS         VIEW
 INFORMATION_SCHEMA               KEY_COLUMN_USAGE                 VIEW
 INFORMATION_SCHEMA               VIEW_TABLE_USAGE                 VIEW
 INFORMATION_SCHEMA               VIEW_COLUMN_USAGE                VIEW
 INFORMATION_SCHEMA               CONSTRAINT_TABLE_USAGE           VIEW
 INFORMATION_SCHEMA               CONSTRAINT_COLUMN_USAGE          VIEW
 INFORMATION_SCHEMA               REFERENTIAL_CONSTRAINTS          VIEW
 INFORMATION_SCHEMA               DATABASES                        VIEW

output (col 1-79 of 107): q
esqint: q
%ESQINT-I-TERMINATED, 16-APR-1998 14:08:33
```

```
Log file of ESQ server DEMOSERV:

%ESQINI-I-ESQINFO, ESQLOG is enabled for ’SES’
%ESQINI-I-ESQINFO, Software AG ADABAS SQL Server starting up on IN-
TEL-80386/WINDOWS NT
%ESQINI-I-ESQINFO, Version:         1.4.2.13
%ESQINI-I-ESQINFO, Startup time:    16-APR-1998 14:07:56
%ESQINI-I-ESQINFO, Server name:     DEMOSERV on node SASME
%ESQINI-I-ESQINFO, Server type:     CSCI
%ESQINI-I-ESQINFO, Catalog files:   DB 5, Files 20-22
%ESQINI-I-ESQINFO, Server pid:      149
%ESQINI-I-ESQINFO, Server image:    G:\SAG\esq\v142d\bin\esqini.exe
%ESQINI-I-ESQINFO, Thread image:    G:\SAG\esq\v142d\bin\esqsrv.exe
%ESQINI-I-ESQINFO, Threads:         2
%ESQINI-I-ESQINFO, Sessions/Thread: 1
%ESQSRV-I-ESQINFO, ESQLOG is enabled for ’SES’
16-APR 14:07:58.54 Server DEMOSERV Thread 1 pid 151 up and running
%ESQSRV-I-ESQINFO, ESQLOG is enabled for ’SES’
16-APR 14:08:00.29 Server DEMOSERV Thread 2 pid 154 up and running
16-APR 14:08:09.83 Server DEMOSERV up and running
16-APR 14:08:21.46 Th2: OPEN SESSION                csusr=SASME:STDUSER
%ESQRTS-I-ESQINFO, ESQLOG is enabled for ’SES’
16-APR 14:08:33.54 Th2: CLOSE SESSION               csusr=SASME:STDUSER
```

Schema Identifier Logging

ADABAS SQL Server schema identifier logging gives information about the schema identifier setting on client and on server side.

Example:

```
% set ESQLOG=schemaident
% esqint
%ESQINT-I-STARTED, 16-APR-1998 14:14:11, Version 1.4/2.13 (INTEL-80386/WINDOWS
NT)
ESQ User: john
Password: %ESQRTS-I-ESQINFO, ESQLOG is enabled for 'SCHEMAIDENT'
%ESQINT-I-CONNECT, Server DEMOSERV connected successfully
esqint: select * from information_schema.tables;
16-APR 14:14:24.18 Th0: schema-ident srv=DBA clt=DBA
16-APR 14:14:24.33 Th0: schema-ident srv=DBA clt=DBA
16-APR 14:14:24.34 Th0: schema-ident srv=DBA clt=DBA
16-APR 14:14:24.34 Th0: schema-ident srv=DBA clt=DBA
16-APR 14:14:24.35 Th0: schema-ident srv=DBA clt=DBA
16-APR 14:14:24.36 Th0: schema-ident srv=DBA clt=DBA
16-APR 14:14:24.93 Th0: schema-ident srv=DBA clt=DBA
 TABLE_SCHEMA                     TABLE_NAME                       TABLE_TYPE

 INFORMATION_SCHEMA               SERVER_INFO                      VIEW
 INFORMATION_SCHEMA               INFORMATION_SCHEMA_CATALOG_NAME  VIEW
 INFORMATION_SCHEMA               SCHEMATA                         VIEW
 INFORMATION_SCHEMA               CLUSTERS                         VIEW
 INFORMATION_SCHEMA               TABLES                           VIEW
 INFORMATION_SCHEMA               BASE_TABLES                      VIEW
 INFORMATION_SCHEMA               VIEWS                            VIEW
 INFORMATION_SCHEMA               COLUMNS                          VIEW
 INFORMATION_SCHEMA               TABLE_PRIVILEGES                 VIEW
 INFORMATION_SCHEMA               COLUMN_PRIVILEGES                VIEW
 INFORMATION_SCHEMA               TABLE_CONSTRAINTS                VIEW
 INFORMATION_SCHEMA               TABLE_INDEXES                    VIEW
 INFORMATION_SCHEMA               ALL_TABLE_INDEX_ELEMENTS         VIEW
 INFORMATION_SCHEMA               KEY_COLUMN_USAGE                 VIEW
 INFORMATION_SCHEMA               VIEW_TABLE_USAGE                 VIEW
 INFORMATION_SCHEMA               VIEW_COLUMN_USAGE                VIEW
 INFORMATION_SCHEMA               CONSTRAINT_TABLE_USAGE           VIEW
 INFORMATION_SCHEMA               CONSTRAINT_COLUMN_USAGE          VIEW
 INFORMATION_SCHEMA               REFERENTIAL_CONSTRAINTS          VIEW
 INFORMATION_SCHEMA               DATABASES                        VIEW

output (col 1-79 of 107): q
esqint: q
16-APR 14:14:28.06 Th0: schema-ident srv=DBA clt=DBA
%ESQINT-I-TERMINATED, 16-APR-1998 14:14:28
```

User Exit Logging

The User Exit Logging gives information about the execution of user exits. Following information is logged:

- call of any user exit
- return of any user exit
- important parameters and return values

Example:

```
% set ESQLOG=userexit
% esqint
%ESQINT-I-STARTED, 16-APR-1998 15:15:34, Version 1.4/2.13 (INTEL-80386/WINDOWS
NT)
ESQ User: john
Password: %ESQRTS-I-ESQINFO, ESQLOG is enabled for 'USEREXIT'
%ESQINT-I-CONNECT, Server DEMOSERV connected successfully
esqint: select yacht_id from yacht;

%ESQRTS-I-ESQINFO, Th0: > uex5(sid:2158, con:-1, cc:OP)
%ESQRTS-I-ESQINFO, Th0: < uex5(rc=0)
%ESQRTS-I-ESQINFO, Th0: > uex5(sid:2158, con:-1, cc:LF)
%ESQRTS-I-ESQINFO, Th0: < uex5(rc=0)
%ESQRTS-I-ESQINFO, Th0: > uex5(sid:2158, con:-1, cc:LF)
%ESQRTS-I-ESQINFO, Th0: < uex5(rc=0)
%ESQRTS-I-ESQINFO, Th0: > uex5(sid:2158, con:-1, cc:L2)
%ESQRTS-I-ESQINFO, Th0: < uex5(rc=0)
%ESQRTS-I-ESQINFO, Th0: > uex5(sid:2158, con:-1, cc:S1)
%ESQRTS-I-ESQINFO, Th0: < uex5(rc=0)
%ESQRTS-I-ESQINFO, Th0: > uex5(sid:2158, con:2158, cc:OP)
%ESQRTS-I-ESQINFO, Th0: < uex5(rc=0)
%ESQRTS-I-ESQINFO, Th0: > uex5(sid:2158, con:-1, cc:L3)
%ESQRTS-I-ESQINFO, Th0: < uex5(rc=0)
%ESQRTS-I-ESQINFO, Th0: > uex5(sid:2158, con:-1, cc:S1)
%ESQRTS-I-ESQINFO, Th0: < uex5(rc=0)
%ESQRTS-I-ESQINFO, Th0: > uex5(sid:2158, con:-1, cc:S1)
%ESQRTS-I-ESQINFO, Th0: < uex5(rc=0)
%ESQRTS-I-ESQINFO, Th0: > uex5(sid:2158, con:-1, cc:S2)
%ESQRTS-I-ESQINFO, Th0: < uex5(rc=0)
%ESQRTS-I-ESQINFO, Th0: > uex5(sid:2158, con:-1, cc:L1)
%ESQRTS-I-ESQINFO, Th0: < uex5(rc=0)
%ESQRTS-I-ESQINFO, Th0: > uex5(sid:2158, con:-1, cc:RC)
```

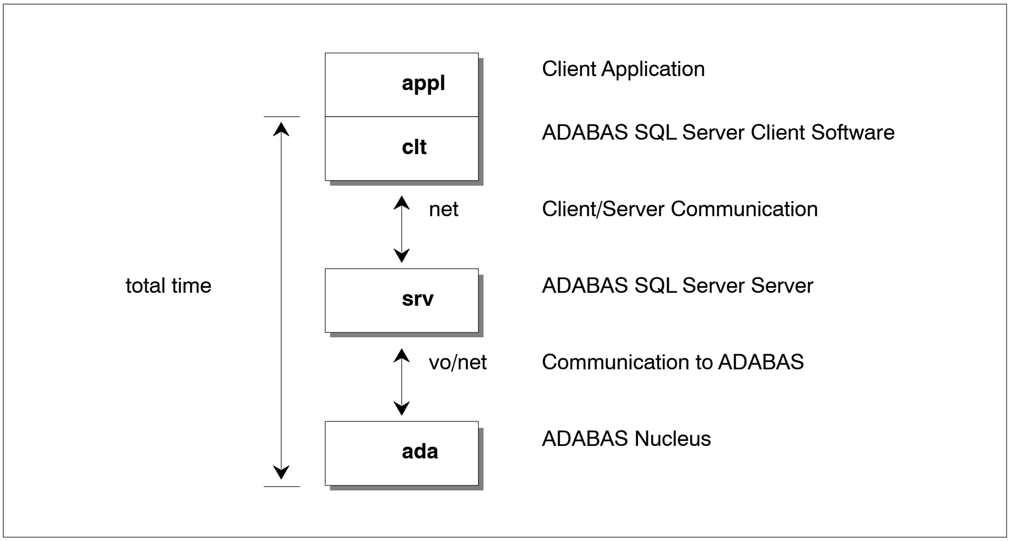
```
%ESQRTS-I-ESQINFO, Th0: < uex5(rc=0)
%ESQRTS-I-ESQINFO, Th0: > uex5(sid:2158, con:2158, cc:LF)
%ESQRTS-I-ESQINFO, Th0: < uex5(rc=0)
%ESQRTS-I-ESQINFO, Th0: > uex5(sid:2158, con:-1, cc:L3)
%ESQRTS-I-ESQINFO, Th0: < uex5(rc=0)
%ESQRTS-I-ESQINFO, Th0: > uex5(sid:2158, con:-1, cc:S1)
%ESQRTS-I-ESQINFO, Th0: < uex5(rc=0)
%ESQRTS-I-ESQINFO, Th0: > uex5(sid:2158, con:-1, cc:L3)
%ESQRTS-I-ESQINFO, Th0: < uex5(rc=0)
%ESQRTS-I-ESQINFO, Th0: > uex5(sid:2158, con:2158, cc:L2)
%ESQRTS-I-ESQINFO, Th0: < uex5(rc=0)

     YACHT_ID

       6230
       6228
       6200
       5001
        157
        156
        155
        154
        153
        152
        151
        150
        149
        148
        147
        146
        145
        144

esqint: q
%ESQRTS-I-ESQINFO, Th0: > uex5(sid:2158, con:2158, cc:BT)
%ESQRTS-I-ESQINFO, Th0: < uex5(rc=0)
%ESQRTS-I-ESQINFO, Th0: > uex5(sid:2158, con:2158, cc:CL)
%ESQRTS-I-ESQINFO, Th0: < uex5(rc=0)
%ESQRTS-I-ESQINFO, Th0: > uex5(sid:2158, con:-1, cc:CL)
%ESQRTS-I-ESQINFO, Th0: < uex5(rc=0)
%ESQINT-I-TERMINATED, 16-APR-1998 15:17:16
```

Elapsed Time Logging

Elapsed Time Logging gives information about the total elapsed time of each SQL request as well as split up into the elapsed time of each software component involved in milliseconds and as a percentage.

The elapsed time values displayed are:

tot	total elapsed time for one SQL request
clt	elapsed time spent by the ADABAS SQL Server client software
net	elapsed time spent by the communication to the server
srv	elapsed time spent by ADABAS SQL Server server
vo/net	elapsed time spent by the communication to ADABAS
ada	elapsed time spent by the ADABAS nucleus



Data Flow and Time Portions

Log lines of the following style are displayed with each SQL request:

```
%ESQRTS-I-Clt: ti-quot     tot     clt     net     srv  vo/net     ada
%ESQRTS-I-Clt:      ms    7599       9      42    4013     640    2895
%ESQRTS-I-Clt:      %      100       0       1      53       8      38
```

Note:
If ESQLOG is set to “ALLTIME” or “TIME=ALL” then additionally the total times spent on each software layer will be displayed.

Example:

```
% set ESQLOG=time
% esqint
%ESQINT-I-STARTED, 16-APR-1998 15:17:57, Version 1.4/2.13 (INTEL-80386/WINDOWS
NT)
ESQ User: john
Password: %ESQRTS-I-ESQINFO, ESQLOG is enabled for 'TIME'
%ESQRTS-I-ESQINFO, Clt: ti-quot     tot     clt     net     srv  vo/net
ada
%ESQRTS-I-ESQINFO, Clt:      ms     261      30       0     200       0
31
%ESQRTS-I-ESQINFO, Clt:      %      100      11       0      77       0
12
%ESQINT-I-CONNECT, Server DEMOSERV connected successfully
esqint: select * from information_schema.tables;
%ESQRTS-I-ESQINFO, Clt: ti-quot     tot     clt     net     srv  vo/net
ada
%ESQRTS-I-ESQINFO, Clt:      ms     390       0       0     200      20
170
%ESQRTS-I-ESQINFO, Clt:      %      100       0       0      51       5
44
%ESQRTS-I-ESQINFO, Clt: ti-quot     tot     clt     net     srv  vo/net
ada
%ESQRTS-I-ESQINFO, Clt:      ms      10       0       0       0       0
10
%ESQRTS-I-ESQINFO, Clt:      %      100       0       0       0       0
100
%ESQRTS-I-ESQINFO, Clt: ti-quot     tot     clt     net     srv  vo/net
ada
%ESQRTS-I-ESQINFO, Clt:      ms       0       0       0       0       0
0
%ESQRTS-I-ESQINFO, Clt:      %      100       0       0       0       0
0
%ESQRTS-I-ESQINFO, Clt: ti-quot     tot     clt     net     srv  vo/net
```

```
ada
%ESQRTS-I-ESQINFO, Clt:      ms      10      10       0       0       0
0
%ESQRTS-I-ESQINFO, Clt:      %      100     100       0       0       0
0
%ESQRTS-I-ESQINFO, Clt: ti-quot     tot     clt     net     srv  vo/net
ada
%ESQRTS-I-ESQINFO, Clt:      ms      31      10       0       0       0
21
%ESQRTS-I-ESQINFO, Clt:      %      100      32       0       0       0
68
%ESQRTS-I-ESQINFO, Clt: ti-quot     tot     clt     net     srv  vo/net
ada
%ESQRTS-I-ESQINFO, Clt:      ms     711       0       0     130      40
541
%ESQRTS-I-ESQINFO, Clt:      %      100       0       0      18       6
76
%ESQRTS-I-ESQINFO, Clt: ti-quot     tot     clt     net     srv  vo/net
ada
%ESQRTS-I-ESQINFO, Clt:      ms       0       0       0       0       0
0
%ESQRTS-I-ESQINFO, Clt:      %      100       0       0       0       0
0
%ESQRTS-I-ESQINFO, Clt: ti-quot     tot     clt     net     srv  vo/net
ada
%ESQRTS-I-ESQINFO, Clt:      ms       0       0       0       0       0
0
%ESQRTS-I-ESQINFO, Clt:      %      100       0       0       0       0
0
%ESQRTS-I-ESQINFO, Clt: ti-quot     tot     clt     net     srv  vo/net
ada
%ESQRTS-I-ESQINFO, Clt:      ms       0       0       0       0       0
0
%ESQRTS-I-ESQINFO, Clt:      %      100       0       0       0       0
0
%ESQRTS-I-ESQINFO, Clt: ti-quot     tot     clt     net     srv  vo/net
ada
%ESQRTS-I-ESQINFO, Clt:      ms       0       0       0       0       0
0
%ESQRTS-I-ESQINFO, Clt:      %      100       0       0       0       0
0
%ESQRTS-I-ESQINFO, Clt: ti-quot     tot     clt     net     srv  vo/net
ada
%ESQRTS-I-ESQINFO, Clt:      ms       0       0       0       0       0
0
%ESQRTS-I-ESQINFO, Clt:      %      100       0       0       0       0
```

```
0
%ESQRTS-I-ESQINFO, Clt: ti-quot     tot     clt     net     srv  vo/net
ada
%ESQRTS-I-ESQINFO, Clt:      ms       0       0       0       0       0
0
%ESQRTS-I-ESQINFO, Clt:      %      100       0       0       0       0
0
%ESQRTS-I-ESQINFO, Clt: ti-quot     tot     clt     net     srv  vo/net
ada
%ESQRTS-I-ESQINFO, Clt:      ms       0       0       0       0       0
0
%ESQRTS-I-ESQINFO, Clt:      %      100       0       0       0       0
0
%ESQRTS-I-ESQINFO, Clt: ti-quot     tot     clt     net     srv  vo/net
ada
%ESQRTS-I-ESQINFO, Clt:      ms       0       0       0       0       0
0
%ESQRTS-I-ESQINFO, Clt:      %      100       0       0       0       0
0
%ESQRTS-I-ESQINFO, Clt: ti-quot     tot     clt     net     srv  vo/net
ada
%ESQRTS-I-ESQINFO, Clt:      ms       0       0       0       0       0
0
%ESQRTS-I-ESQINFO, Clt:      %      100       0       0       0       0
0
%ESQRTS-I-ESQINFO, Clt: ti-quot     tot     clt     net     srv  vo/net
ada
%ESQRTS-I-ESQINFO, Clt:      ms       0       0       0       0       0
0
%ESQRTS-I-ESQINFO, Clt:      %      100       0       0       0       0
0
%ESQRTS-I-ESQINFO, Clt: ti-quot     tot     clt     net     srv  vo/net
ada
%ESQRTS-I-ESQINFO, Clt:      ms       0       0       0       0       0
0
%ESQRTS-I-ESQINFO, Clt:      %      100       0       0       0       0
0
%ESQRTS-I-ESQINFO, Clt: ti-quot     tot     clt     net     srv  vo/net
ada
%ESQRTS-I-ESQINFO, Clt:      ms       0       0       0       0       0
0
%ESQRTS-I-ESQINFO, Clt:      %      100       0       0       0       0
0
%ESQRTS-I-ESQINFO, Clt: ti-quot     tot     clt     net     srv  vo/net
ada
%ESQRTS-I-ESQINFO, Clt:      ms       0       0       0       0       0
```

```
0
%ESQRTS-I-ESQINFO, Clt:       %     100       0       0       0       0
0
%ESQRTS-I-ESQINFO, Clt: ti-quot     tot     clt     net     srv  vo/net
ada
%ESQRTS-I-ESQINFO, Clt:      ms       0       0       0       0       0
0
%ESQRTS-I-ESQINFO, Clt:       %     100       0       0       0       0
0
%ESQRTS-I-ESQINFO, Clt: ti-quot     tot     clt     net     srv  vo/net
ada
%ESQRTS-I-ESQINFO, Clt:      ms       0       0       0       0       0
0
%ESQRTS-I-ESQINFO, Clt:       %     100       0       0       0       0
0
%ESQRTS-I-ESQINFO, Clt: ti-quot     tot     clt     net     srv  vo/net
ada
%ESQRTS-I-ESQINFO, Clt:      ms     110      10       0      20      10
70
%ESQRTS-I-ESQINFO, Clt:       %     100       9       0      18       9
64
%ESQRTS-I-ESQINFO, Clt: ti-quot     tot     clt     net     srv  vo/net
ada
%ESQRTS-I-ESQINFO, Clt:      ms       0       0       0       0       0
0
%ESQRTS-I-ESQINFO, Clt:       %     100       0       0       0       0
0
%ESQRTS-I-ESQINFO, Clt: ti-quot     tot     clt     net     srv  vo/net
ada
%ESQRTS-I-ESQINFO, Clt:      ms       0       0       0       0       0
0
%ESQRTS-I-ESQINFO, Clt:       %     100       0       0       0       0
0
%ESQRTS-I-ESQINFO, Clt: ti-quot     tot     clt     net     srv  vo/net
ada
%ESQRTS-I-ESQINFO, Clt:      ms       0       0       0       0       0
0
%ESQRTS-I-ESQINFO, Clt:       %     100       0       0       0       0
0

 TABLE_SCHEMA                     TABLE_NAME                       TABLE_TYPE

 INFORMATION_SCHEMA               SERVER_INFO                      VIEW
 INFORMATION_SCHEMA               INFORMATION_SCHEMA_CATALOG_NAME  VIEW
 INFORMATION_SCHEMA               SCHEMATA                         VIEW
 INFORMATION_SCHEMA               CLUSTERS                         VIEW
 INFORMATION_SCHEMA              TABLES                           VIEW
 INFORMATION_SCHEMA              BASE_TABLES                      VIEW
 INFORMATION_SCHEMA              VIEWS                            VIEW
 INFORMATION_SCHEMA              COLUMNS                          VIEW
 INFORMATION_SCHEMA              TABLE_PRIVILEGES                 VIEW
 INFORMATION_SCHEMA              COLUMN_PRIVILEGES                VIEW
 INFORMATION_SCHEMA              TABLE_CONSTRAINTS                VIEW
 INFORMATION_SCHEMA              TABLE_INDEXES                    VIEW
 INFORMATION_SCHEMA              ALL_TABLE_INDEX_ELEMENTS         VIEW
 INFORMATION_SCHEMA              KEY_COLUMN_USAGE                 VIEW
 INFORMATION_SCHEMA              VIEW_TABLE_USAGE                 VIEW
 INFORMATION_SCHEMA              VIEW_COLUMN_USAGE                VIEW
 INFORMATION_SCHEMA              CONSTRAINT_TABLE_USAGE           VIEW
 INFORMATION_SCHEMA              CONSTRAINT_COLUMN_USAGE          VIEW
 INFORMATION_SCHEMA              REFERENTIAL_CONSTRAINTS          VIEW
 INFORMATION_SCHEMA              DATABASES                        VIEW

output (col 1-79 of 107): q
esqint: q
%ESQRTS-I-ESQINFO, Clt: ti-quot     tot     clt     net     srv  vo/net
ada
%ESQRTS-I-ESQINFO, Clt:      ms       0       0       0       0       0
0
%ESQRTS-I-ESQINFO, Clt:      %      100       0       0       0       0
0
%ESQRTS-I-ESQINFO, Clt: ti-quot     tot     clt     net     srv  vo/net
ada
%ESQRTS-I-ESQINFO, Clt:      ms      70       0       0      10      10
50
%ESQRTS-I-ESQINFO, Clt:      %      100       0       0      14      14
71
%ESQRTS-I-ESQINFO, Clt: elapsed time for ESQ session    1593 ms
%ESQINT-I-TERMINATED, 16-APR-1998 15:18:12
%ESQRTS-I-ESQINFO, Clt: ti-quot     tot     clt     net     srv  vo/net
ada
%ESQRTS-I-ESQINFO, Clt:      ms       0       0       0       0       0
0
%ESQRTS-I-ESQINFO, Clt:      %      100       0       0       0       0
0
%ESQRTS-I-ESQINFO, Clt: elapsed time for ESQ session       0 ms
```

Sort Logging

When using ORDER BY or GROUP BY clauses, ADABAS SQL Server is in some cases forced to perform an explicit internal sorting of data. The sorting itself is done by the VO layer of the ADABAS SQL Server. This implies that on different platforms different sorting algorithms are used. This sorting may become a bottleneck during processing. To allow monitoring and analysis of the sort step, a special sort logging is offered by ADABAS SQL Server.

If the environment variable ESQLOG is set to “SORT”, the sort step will log statistical information about the resource usage. These statistics can be analyzed. Based on the analysis, default sort buffer sizes may be newly assigned, thus improving the performance.

Example:

```
% set ESQLOG=SORT%ESQINT-I-STARTED, 16-APR-1998 17:16:25, Version 1.4/2.13
(INTEL-80386/WINDOWS N
T)
ESQ User: esq
Password: %ESQRTS-I-ESQINFO, ESQLOG is enabled for 'SORT'
%ESQINT-I-CONNECT, Server DEMOSERV connected successfully
esqint: select sailor_name,sailor_name from sagtours.sailor order by 1,2;
%SAGVO-I-SORTBYTES, sum  29248, core  16384, file  12864, core_size 16KB
%SAGVO-I-SORTRECS,  sum    457, core    256, file    201, ratio 56:44
%SAGVO-I-SORTTIME,  sum     20, load      0, sort     20, unload      0

 SAILOR_NAME                    SAILOR_NAME

 ADAM DERFLER                   ADAM DERFLER
 ADAM KING                      ADAM KING
 AGNES GRGOIRE                  AGNES GRGOIRE
 ALAIN                          ALAIN
 ALAIN ASTIER                   ALAIN ASTIER
 ALEX BARTON                    ALEX BARTON
 ALEX FINLEY                    ALEX FINLEY
 ALEXANDRE GOETSCHEL            ALEXANDRE GOETSCHEL
 ALFI DORSCH                    ALFI DORSCH
 ALFRED SCHNEIDER               ALFRED SCHNEIDER
 ALFRED WOODS                   ALFRED WOODS
 ALFREDO MARTINEZ               ALFREDO MARTINEZ
 ALVAREZ, YOLANDA               ALVAREZ, YOLANDA
 AMANDA STEVENS                 AMANDA STEVENS
 AMELY DILGER                   AMELY DILGER
 ANDRE                          ANDRE
 ANGELA PEREZ                   ANGELA PEREZ
 ANJA                           ANJA
 ANN WOODS                      ANN WOODS
 ANNE GREY                      ANNE GREY

output (col 1-79 of 62): q
esqint: quit
%ESQINT-I-TERMINATED, 16-APR-1998 17:17:12
```

Discussion:

```
SORTBYTES:
Total volume sort data   : 29248 Bytes
In buffer                : 16384 Bytes
In temporary file        : 12864 Bytes
Current core area        :    16 KB     (DEFAULT)
```

```
SORTRECS:
Total number of records  :   457
In buffer                :   256
In file                  :   201
Ratio buffer/temp. file  : 56:44

SORTTIME:
Load time                :    0 Milliseconds
Sort time                :   20 Milliseconds
Unload time              :    0 Millisecond
Total sort time          :   20 Milliseconds
```

Note:
Loading/Unloading of records is performed in 16 KB chunks.

If the default sort-buffer size is set to 48 (unit = KB), the sort logging will change as follows.

```
% set ESQSRTBUF=48
%ESQINT-I-STARTED, 16-APR-1998 17:21:56, Version 1.4/2.13 (INTEL-80386/WINDOWS
N
T)
ESQ User: esq
Password: %ESQRTS-I-ESQINFO, ESQLOG is enabled for 'SORT'
%ESQINT-I-CONNECT, Server DEMOSERV connected successfully
esqint: select sailor_name,sailor_name from sagtours.sailor order by 1,2;
%SAGVO-I-SORTBYTES, sum  29248, core  29248, file      0, core_size 48KB
%SAGVO-I-SORTRECS,  sum    457, core    457, file      0, ratio 100:0
%SAGVO-I-SORTTIME,  sum     10, load      0, sort     10, unload      0

 SAILOR_NAME                    SAILOR_NAME

 ADAM DERFLER                   ADAM DERFLER
 ADAM KING                      ADAM KING
 AGNES GRGOIRE                  AGNES GRGOIRE
 ALAIN                          ALAIN
 ALAIN ASTIER                   ALAIN ASTIER
 ALEX BARTON                    ALEX BARTON
 ALEX FINLEY                    ALEX FINLEY
 ALEXANDRE GOETSCHEL            ALEXANDRE GOETSCHEL
 ALFI DORSCH                    ALFI DORSCH
 ALFRED SCHNEIDER               ALFRED SCHNEIDER
 ALFRED WOODS                   ALFRED WOODS
 ALFREDO MARTINEZ               ALFREDO MARTINEZ
 ALVAREZ, YOLANDA               ALVAREZ, YOLANDA
 AMANDA STEVENS                 AMANDA STEVENS
 AMELY DILGER                   AMELY DILGER
 ANDRE                          ANDRE
 ANGELA PEREZ                   ANGELA PEREZ
 ANJA                           ANJA
 ANN WOODS                      ANN WOODS
 ANNE GREY                      ANNE GREY

output (col 1-79 of 62): q
esqint: q
%ESQINT-I-TERMINATED, 16-APR-1998 17:22:16
```

ADABAS Utility Logging

ADABAS SQL Server ADABAS Utility logging gives information about calls from the ADABAS SQL Server run-time system to an ADABAS utility. ADABAS utilities are used by ADABAS SQL Server to execute SQL DDL statements like CREATE TABLE.

Example:

```
% set ESQLOG=adu
% esqint
%ESQINT-I-STARTED, 16-APR-1998 17:02:06, Version 1.4/2.13 (INTEL-80386/WINDOWS
NT)
ESQ User: dba
Password: %ESQRTS-I-ESQINFO, ESQLOG is enabled for 'ADU'
%ESQINT-I-CONNECT, Server DEMOSERV connected successfully
esqint: create schema dba authorization dba;
%ESQINT-I-SUCCESS, SQL statement completed successfully
esqint: create table dba.table1 (col1 char(3));
16-APR 17:02:20.96 Th0: ADU > adafdu
%ESQRTS-I-ESQINFO, Th0: ADU - DBID=5
%ESQRTS-I-ESQINFO, Th0: ADU - FILE=12
%ESQRTS-I-ESQINFO, Th0: ADU - NAME=DBA.TABLE1
%ESQRTS-I-ESQINFO, Th0: ADU - DSSIZE=10B
%ESQRTS-I-ESQINFO, Th0: ADU - NISIZE=10B
%ESQRTS-I-ESQINFO, Th0: ADU - UISIZE=10B
%ESQRTS-I-ESQINFO, Th0: ADU - MAXISN=300
%ESQRTS-I-ESQINFO, Th0: ADU - REUSE=(DS)
%ESQRTS-I-ESQINFO, Th0: ADU - FIELDS
%ESQRTS-I-ESQINFO, Th0: ADU - 01,AA,3,A,NC
;COL1
%ESQRTS-I-ESQINFO, Th0: ADU - END_OF_FIELDS
16-APR 17:02:22.11 Th0: ADU < rsp=0/0/0
%ESQINT-I-SUCCESS, SQL statement completed successfully
esqint: quit
%ESQINT-I-TERMINATED, 16-APR-1998 17:02:23
```

Security Logging

The logging mechanism of ADABAS SQL Server also enables logging of GRANT and REVOKE statements and user system access via CONNECT statements.

Examples:

Example 1:

```
% set ESQLOG=SEC
```

The GRANT/REVOKE statement will be logged. If any errors occur, the corresponding response codes are also displayed.

```
13-APR 13:41:19.93 Th0: VDK: GRANT SELECT, INSERT ON VDK.TABLE1 TO BRU
==> Finished (Rsp: 0)
```

The grant statement has successfully finished.

```
13-APR 13:41:59.79 Th0: BRU: GRANT UPDATE ON VDK.TABLE1 TO TRO
==> Finished (Rsp: -6702)
```

The grant statement has finished with an error, in this case with a security violation, i.e. the grantor is not allowed to grant this privilege.

```
13-APR 13:43:42.37 Th0: VDK: GRANT ALL ON VDK.TABLE1 TO FS WITH GRANT OPTION
==> Finished (Rsp: 0)
```

The grant statement has successfully finished.

The CONNECT statement results in a password verification which will also be logged if ESQLOG is set to SEC.

```
13-APR 13:49:54.88 Th0: User VDK has connected successfully.

13-APR 13:51:28.96 Th0: Authentication error during CONNECT for user BRU.
```

Example 2:

```
% set ESQLOG="SEC=ALL"
```

In this case, additionally, every single GRANT/REVOKE step resulting from a GRANT ALL/ REVOKE ALL is logged.

```
13-APR 13:56:23.41 Th0: VDK: REVOKE ALL ON VDK.TABLE1 FROM AE ==> Started
```

A REVOKE ALL statement has started.

```
13-APR 13:56:23.52 Th0: VDK: REVOKE UPDATE(COL1) ON VDK.TABLE1 FROM AE
==> (generated from ALL)
13-APR 13:56:24.04 Th0: VDK: REVOKE UPDATE(COL2) ON VDK.TABLE1 FROM AE
==> (generated from ALL)
13-APR 13:56:24.11 Th0: VDK: REVOKE UPDATE(COL4) ON VDK.TABLE1 FROM AE
==> (generated from ALL)
13-APR 13:56:24.22 Th0: VDK: REVOKE SELECT ON VDK.TABLE1 FROM AE
==> (generated from ALL)
13-APR 13:56:24.30 Th0: VDK: REVOKE INSERT ON VDK.TABLE1 FROM AE
==> (generated from ALL)
```

The individual REVOKE steps result from the above REVOKE ALL statement.

```
13-APR 13:56:24.38 Th0: VDK: REVOKE ALL ON VDK.TABLE1 FROM AE ==> Finished
(Rsp: 0)
```

A REVOKE ALL statement has sucessfully finished.

Example 3:

```
% set ESQLOG="-SEC"
```

No GRANT/REVOKE statements will be logged. The same is true if nothing is specified.

ESQLNK Call Logging

ESQLNK Call Logging gives information about:

- the point in time where the application is calling ESQLNK, esqerr or esqint and
- the point in time when control is returned to the application

Example:

```
% set ESQLOG=entry
% esqint
%ESQINT-I-STARTED, 16-APR-1998 15:32:34, Version 1.4/2.13 (INTEL-80386/WINDOWS
NT)
ESQ User: john
Password: %ESQRTS-I-ESQINFO, ESQLOG is enabled for 'ENTRY'
%ESQRTS-I-ESQINFO, Clt: > ESQLNK()        cnt=   1
%ESQRTS-I-ESQINFO, Clt: < ESQLNK()        cnt=   1      rsp=0
%ESQRTS-I-ESQINFO, Clt: > esqinf()        cnt=   2
%ESQRTS-I-ESQINFO, Clt: < esqinf()        cnt=   2      rsp=0
%ESQRTS-I-ESQINFO, Clt: > esqinf()        cnt=   3
%ESQRTS-I-ESQINFO, Clt: < esqinf()        cnt=   3      rsp=0
%ESQRTS-I-ESQINFO, Clt: > esqinf()        cnt=   4
%ESQRTS-I-ESQINFO, Clt: < esqinf()        cnt=   4      rsp=0
%ESQRTS-I-ESQINFO, Clt: > esqinf()        cnt=   5
%ESQRTS-I-ESQINFO, Clt: < esqinf()        cnt=   5      rsp=0
%ESQRTS-I-ESQINFO, Clt: > esqinf()        cnt=   6
%ESQRTS-I-ESQINFO, Clt: < esqinf()        cnt=   6      rsp=0
%ESQINT-I-CONNECT, Server DEMOSERV connected successfully
esqint: select * from information_schema.tables;
%ESQRTS-I-ESQINFO, Clt: > ESQLNK()        cnt=   7
%ESQRTS-I-ESQINFO, Clt: < ESQLNK()        cnt=   7      rsp=0
%ESQRTS-I-ESQINFO, Clt: > ESQLNK()        cnt=   8
%ESQRTS-I-ESQINFO, Clt: < ESQLNK()        cnt=   8      rsp=0
%ESQRTS-I-ESQINFO, Clt: > ESQLNK()        cnt=   9
%ESQRTS-I-ESQINFO, Clt: < ESQLNK()        cnt=   9      rsp=0
%ESQRTS-I-ESQINFO, Clt: > ESQLNK()        cnt=  10
%ESQRTS-I-ESQINFO, Clt: < ESQLNK()        cnt=  10      rsp=0
%ESQRTS-I-ESQINFO, Clt: > ESQLNK()        cnt=  11
%ESQRTS-I-ESQINFO, Clt: < ESQLNK()        cnt=  11      rsp=0
%ESQRTS-I-ESQINFO, Clt: > ESQLNK()        cnt=  12
%ESQRTS-I-ESQINFO, Clt: < ESQLNK()        cnt=  12      rsp=0
%ESQRTS-I-ESQINFO, Clt: > ESQLNK()        cnt=  13
%ESQRTS-I-ESQINFO, Clt: < ESQLNK()        cnt=  13      rsp=0
%ESQRTS-I-ESQINFO, Clt: > ESQLNK()        cnt=  14
```

```
%ESQRTS-I-ESQINFO, Clt: < ESQLNK()        cnt=  14       rsp=0
%ESQRTS-I-ESQINFO, Clt: > ESQLNK()        cnt=  15
%ESQRTS-I-ESQINFO, Clt: < ESQLNK()        cnt=  15       rsp=0
%ESQRTS-I-ESQINFO, Clt: > ESQLNK()        cnt=  16
%ESQRTS-I-ESQINFO, Clt: < ESQLNK()        cnt=  16       rsp=0
%ESQRTS-I-ESQINFO, Clt: > ESQLNK()        cnt=  17
%ESQRTS-I-ESQINFO, Clt: < ESQLNK()        cnt=  17       rsp=0
%ESQRTS-I-ESQINFO, Clt: > ESQLNK()        cnt=  18
%ESQRTS-I-ESQINFO, Clt: < ESQLNK()        cnt=  18       rsp=0
%ESQRTS-I-ESQINFO, Clt: > ESQLNK()        cnt=  19
%ESQRTS-I-ESQINFO, Clt: < ESQLNK()        cnt=  19       rsp=0
%ESQRTS-I-ESQINFO, Clt: > ESQLNK()        cnt=  20
%ESQRTS-I-ESQINFO, Clt: < ESQLNK()        cnt=  20       rsp=0
%ESQRTS-I-ESQINFO, Clt: > ESQLNK()        cnt=  21
%ESQRTS-I-ESQINFO, Clt: < ESQLNK()        cnt=  21       rsp=0
%ESQRTS-I-ESQINFO, Clt: > ESQLNK()        cnt=  22
%ESQRTS-I-ESQINFO, Clt: < ESQLNK()        cnt=  22       rsp=0
%ESQRTS-I-ESQINFO, Clt: > ESQLNK()        cnt=  23
%ESQRTS-I-ESQINFO, Clt: < ESQLNK()        cnt=  23       rsp=0
%ESQRTS-I-ESQINFO, Clt: > ESQLNK()        cnt=  24
%ESQRTS-I-ESQINFO, Clt: < ESQLNK()        cnt=  24       rsp=0
%ESQRTS-I-ESQINFO, Clt: > ESQLNK()        cnt=  25
%ESQRTS-I-ESQINFO, Clt: < ESQLNK()        cnt=  25       rsp=0
%ESQRTS-I-ESQINFO, Clt: > ESQLNK()        cnt=  26
%ESQRTS-I-ESQINFO, Clt: < ESQLNK()        cnt=  26       rsp=0
%ESQRTS-I-ESQINFO, Clt: > ESQLNK()        cnt=  27
%ESQRTS-I-ESQINFO, Clt: < ESQLNK()        cnt=  27       rsp=0
%ESQRTS-I-ESQINFO, Clt: > ESQLNK()        cnt=  28
%ESQRTS-I-ESQINFO, Clt: < ESQLNK()        cnt=  28       rsp=0
%ESQRTS-I-ESQINFO, Clt: > ESQLNK()        cnt=  29
%ESQRTS-I-ESQINFO, Clt: < ESQLNK()        cnt=  29       rsp=0
%ESQRTS-I-ESQINFO, Clt: > ESQLNK()        cnt=  30
%ESQRTS-I-ESQINFO, Clt: < ESQLNK()        cnt=  30       rsp=0
%ESQRTS-I-ESQINFO, Clt: > ESQLNK()        cnt=  31
%ESQRTS-I-ESQINFO, Clt: < ESQLNK()        cnt=  31       rsp=0

 TABLE_SCHEMA                     TABLE_NAME                       TABLE_TYPE

 INFORMATION_SCHEMA               SERVER_INFO                      VIEW
 INFORMATION_SCHEMA               INFORMATION_SCHEMA_CATALOG_NAME  VIEW
 INFORMATION_SCHEMA               SCHEMATA                         VIEW
 INFORMATION_SCHEMA               CLUSTERS                         VIEW
 INFORMATION_SCHEMA               TABLES                           VIEW
 INFORMATION_SCHEMA               BASE_TABLES                      VIEW
 INFORMATION_SCHEMA               VIEWS                            VIEW
 INFORMATION_SCHEMA               COLUMNS                          VIEW
 INFORMATION_SCHEMA               TABLE_PRIVILEGES                 VIEW
 INFORMATION_SCHEMA               COLUMN_PRIVILEGES                VIEW
 INFORMATION_SCHEMA               TABLE_CONSTRAINTS                VIEW
 INFORMATION_SCHEMA               TABLE_INDEXES                    VIEW
 INFORMATION_SCHEMA               ALL_TABLE_INDEX_ELEMENTS         VIEW
 INFORMATION_SCHEMA               KEY_COLUMN_USAGE                 VIEW
 INFORMATION_SCHEMA               VIEW_TABLE_USAGE                 VIEW
 INFORMATION_SCHEMA               VIEW_COLUMN_USAGE                VIEW
 INFORMATION_SCHEMA               CONSTRAINT_TABLE_USAGE           VIEW
 INFORMATION_SCHEMA               CONSTRAINT_COLUMN_USAGE          VIEW
 INFORMATION_SCHEMA               REFERENTIAL_CONSTRAINTS          VIEW
 INFORMATION_SCHEMA               DATABASES                        VIEW

output (col 1-79 of 107): q
esqint: quit
%ESQRTS-I-ESQINFO, Clt: > ESQLNK()        cnt=  32
%ESQRTS-I-ESQINFO, Clt: < ESQLNK()        cnt=  32      rsp=0
%ESQRTS-I-ESQINFO, Clt: > ESQLNK()        cnt=  33
%ESQRTS-I-ESQINFO, Clt: < ESQLNK()        cnt=  33      rsp=0
%ESQINT-I-TERMINATED, 16-APR-1998 15:32:47
%ESQRTS-I-ESQINFO, Clt: > ESQLNK()        cnt=  34
%ESQRTS-I-ESQINFO, Clt: < ESQLNK()        cnt=  34      rsp=0
%ESQRTS-I-ESQINFO, Clt: > esqtrm()        cnt=  35
%ESQRTS-I-ESQINFO, Clt: < esqtrm()        cnt=  35      rsp=0
```

Explain Logging

This logging facility provides information about the execution of an SQL statement within ADABAS SQL Server. It provides a pseudo-code representation of the execution path of the statement. Within the pseudo-code the following information is available:

- ADABAS access commands
- Descriptors used for searching the tables
- Restriction evaluation after fetch has occurred
- Group by and order by information
- Predicates and subqueries

Detailed knowledge of ADABAS and SQL is needed in order to be able to fully interpret the resultant logging information.

To turn the logging facility on, the ESQLOG environment variable must be set to EXPL.

example: % setenv ESQLOG EXPL

To produce an output a DML statement must be executed. If the meta-program already exists, then no output will be produced, but the statement needs to be one of the following types:

SELECT	Explains the statements execution fully
DELETE	Explains the searching phase of the delete statement
UPDATE	Explains the searching phase of the update statement
INSERT	Explains the searching phase on nested tables

Note:
If a view is used within the statement, then the statement is "view merged". View merging is where all references to the view are replaced by appropriate references to the base table/tables upon which the view is based. Hence the explain logging output will not contain any view references, rather table references.

Example:

- The created view:

```
create view view_1 as
     select start_harbor, destination_harbor
          from cruise
          where cruise_price < 1000;
```

- The executed statement:

```
select destination_harbor
     from view_1
          where start_harbor = 'MIAMI';
```

- The explained statement:

```
select destination_harbor
     from cruise
          where start_harbor = 'MIAMI'
          and cruise_price < 1000;
```

The result of the explained statement :

```
********   Start of EXPLAIN logging   ********

for (L2 ;SAGTOURS.CRUISE-0; L2 )  /* GET NEXT */
{
  (((SAGTOURS.CRUISE.START_HARBOR = 'MIAMI' )
    (SAGTOURS.CRUISE.CRUISE_PRICE < 1000 )))
}

********    End  of EXPLAIN logging    ********
```

ADABAS Access Commands

In an SQL query statement, there is generally some access to the underlying database. The SQL is translated to ADABAS database access commands. The ADABAS commands that are used in the explained statement are displayed as the actual ADABAS command names, for example: S1, L3, L2. For details on the ADABAS commands, refer to the *ADABAS Command Reference Manual*.

The Explain logging for a fetch is similar to a "FOR" construct in "C". It has two parts for the ADABAS commands: one for get first record, and another for get next record. The S1 command cannot be used in the get next section; one of the other commands, normally an L1(N) is used. S1s and L3s are displayed before the "for" loop header, along with their search buffers, rather than as part of the loop header. Also, there is a section which shows the table name, the schema name and the table level, whether it is LEVEL 0, 1 or 2. With an L1I, additional information of upper and lower searching bounds is displayed before the "for" loop header as with the S1s and L3s.

An example fetch header for a simple table access:

```
select cruise_id from cruise;
********   Start of EXPLAIN logging   ********

S1 (AA > 00AA EOF )
for (   ;SAGTOURS.CRUISE-0; L1N)  /* GET NEXT */
{
}

********    End of EXPLAIN logging    ********

or

select * from contract;
********   Start of EXPLAIN logging   ********

for (L2;SAGTOURS.CONTRACT-0;L2 ) /* GET NEXT */
{
}

********    End of EXPLAIN logging    ********
```

If a query on a clustered table is logged, then a different output is used so that the access on the tables with lower table levels than the requested table can be documented.

Example (buildings is a level 2 table):

```
select building_name, height
from buildings;
********   Start of EXPLAIN logging   ********

for (L2 ;SAGTOURS.BUILDINGS-0; L2 )  /* GET NEXT */
{
  for (every occurrence SAGTOURS.BUILDINGS-1; )
  {
    if (buffer exhausted)
      L1  refill

    for (every occurrence SAGTOURS.BUILDINGS-2; )
    {
      if (buffer exhausted)
        L1  refill

    }
  }
}

********    End of EXPLAIN logging    ********
```

Here an access is made for every subtable below the requested table if the buffer for that subtable access is exhausted. The L1 refill should only occur if the buffering factor is incorrectly set. This factor can be altered via a DDL statement.

Descriptor Searching

Descriptor searching occurs when ADABAS S1 or L3 commands are issued. It shows what descriptor/super descriptors and operands are used for the search criteria. The search buffer is displayed as the ADABAS SQL Server internal format. Prefixing the search buffer is the ADABAS command that is associated with the buffer. So a typical example of a search buffer is:

```
S1 (AA = 0078 EOF )
```

- AA is the ADABAS shortname of the descriptor used.
- The "=" is the comparison operator used for filtering unwanted records.
- The 4-digit hexadecimal number is an offset into an internal ADABAS SQL Server storage table which holds the value that is to be compared against.
- EOF is the end-of-buffer marker, not present with L3 search buffers.

With an L3 it is possible to receive a search buffer like the following:

```
L3 (AA <= {temp value} )
```

The {temp value} is a value that has not been calculated yet, i.e. it is not a constant. This may be a value that has be retrieved from a previous table access, and therefore is not known at compile time.

Also an L1I produces an output similar to S1s and L3s. However, they are not based on a descriptor but on the ADABAS ISN column (SEQNO). Below is an example L1I buffer:

```
L1I (ISN >= 2 AND ISN <= 4 )
```

There may be a case where an S1 may be produced without a loop. This is where a set of ISNs needs to be established for processing multiple times or where the resultant set is accessed by another S1. At most, this command will be executed only once for that specific table. These will then be displayed as:

```
select cruise_id from cruise where exists
(select * from yacht where yacht_id > 0);
********   Start of EXPLAIN logging   ********

if (first execution for SAGTOURS.YACHT-0;)
{
  S1 (AA > 006E EOF )
}
  for (R1 ;SAGTOURS.YACHT-0; L1N)  /* GET NEXT */
  {
  }
    for (L2 ;SAGTOURS.CRUISE-0; L2 )  /* GET NEXT */
    {
    }

********    End  of EXPLAIN logging    ********
```

The other case where an S1 may be produced without a loop is where there will only be one result returned from the S1 (a unique descriptor search). Then the display will look like:

```
select * from yacht where yacht_id = 152;
********   Start of EXPLAIN logging   ********

S1 (AA = 0082 EOF ) on SAGTOURS.YACHT-0;

********    End of EXPLAIN logging    ********
```

Restriction Evaluation

Restriction evaluation is the process of removing unwanted records from the set returned by the ADABAS database access. The format of its display is similar to the same predicate in the SQL statement that is being explained. However, the only logical connector that is displayed is the “OR” operator. No “AND” operators will be displayed. Therefore, between each factor, if an “OR” is not displayed, then it should be assumed that the operator is an “AND”.

If a “NOT” predicate is used, this will not be shown. The statement that is displayed will be the negated version of the original statement according to the rules of DeMorgan’s theorem. The data type BINARY is displayed as its hexadecimal value.

Examples:

```
          SQL statement                    Explain output
-------------------------------------+-----------------------------------
 where NOT(column_1 > 65 );          |    (((column_1 <= 65)))
                                     |
 where bin_col > Y’10101100’;        |    (((bin_col > H’AC’ )))
```

The predicates are split up into expressions, terms and factors. Each section has its opening and closing brackets, to mark the start and end of each. Each expression can have many terms and each term can have many factors.

For example:

```
select   yacht_id, yacht_name
from     yacht
where    yacht_id = :host_variable_1
or       yacht_name = :host_variable_2
and yacht_id = 23 * yacht_length;
********   Start of EXPLAIN logging   ********

for (L2 ;SAGTOURS.YACHT-0; L2 )  /* GET NEXT */
{
  (((SAGTOURS.YACHT.YACHT_ID = ? ))
  OR  ((SAGTOURS.YACHT.YACHT_NAME = ? )
       (SAGTOURS.YACHT.YACHT_ID = 23 * SAGTOURS.YACHT.YACHT_LENGTH )))
      }
********    End of EXPLAIN logging    ********
```

Note:
The “?” is used to indicate a host variable or a parameter marker.

Grouping and Ordering

These actions are caused by the GROUP and ORDER clauses of SQL. An order clause may be obsolete if the equivalent occurs in the group clause or if it can be implemented by an ordered retrieval (L3). In the output, there is an indication of what sorting is done, and then what constraints are done on each of the groups. The constraints are defined in a HAVING clause. Each SQL statement with grouping will, at its most inner point of the query, have a “save tuple” command.

Example of grouping and ordering:

```
select yacht_type from yacht
group by yacht_type
having min(yacht_name) ='AQUIVA';
********   Start of EXPLAIN logging   ********

for (L2 ;SAGTOURS.YACHT-0; L2 )  /* GET NEXT */
{
      save tuple
}
Sort by:
  SAGTOURS.YACHT.YACHT_TYPE
for (every group)
{
   HAVING:
   ((( MIN()  = 'AQUIVA' )))
}

********    End of EXPLAIN logging    ********
```

Predicates and Subqueries

The predicates BETWEEN and IN are not displayed in their original versions. The BETWEEN is transformed to a range consisting of an upper and a lower bound value. IN predicates are transformed to comparison predicates, which are connected by an OR.

For example:

```
SQL statement                                         Explain output
--------------------------------------+-----------------------------------
where column_1 between 3 and 78;      | (((table_name.column_1 >= 3 )
                                      | (table_name.column_1 <= 78 )))
                                      |
where column_1 in (12, 13, 14)        | (((table_name.column_1 = 12 ))
                                      | ((table_name.column_1 = 13 ))
                                      | ((table_name.column_1 = 14 )))
```

Other predicates are not transformed.

There are two sorts of subqueries: there are those that can be solved before the main execution, and then there are those that are solved during the main execution. If a subquery contains a reference to a table declared in the outer query then no distinction is made about which table it actual is if the tables have the same name.

In predicates where a subquery is used, what is displayed depends on the type of subquery. Those that have been solved before the main query execution are replaced by “pre_evaluated subquery”. Those that are to be executed within the main execution because of the need of values obtained from outside the subquery or because there is more than one record returned are displayed at the point where they would be executed.

Example:

```
select id_customer
from contract
where id_cruise = ALL(select cruise_id from cruise);
********   Start of EXPLAIN logging   ********

for (L2 ;SAGTOURS.CONTRACT-0; L2 )  /* GET NEXT */
{
  (((SAGTOURS.CONTRACT.ID_CRUISE = )))

      S1 (AA <> 0096 EOF )
      for (   ;SAGTOURS.CRUISE-0; L1N)  /* GET NEXT */
      {
      }
}

********    End of EXPLAIN logging    ********
```

If the “NOT” predicate is used with an EXISTS, where the subquery has an outer reference, then any table joins will show the inner table columns first.

For example:

```
      select * from person a
      where NOT EXISTS ( select * from yacht b
              where a.person_id >= b.id_owner);

 is explained as:

      for (L2 ;SAGTOURS.PERSON-0; L2 )  /* GET NEXT */
      {
         for (L2 ;SAGTOURS.YACHT-0; L2 )  /* GET NEXT */
         {
         (((SAGTOURS.YACHT.ID_OWNER <= SAGTOURS.PERSON.PERSON_ID)))
         }
      }
```

UNIONs have no special representation within Explain. Each part of the UNION is treated as a separate statement, and the representation of the UNIONs execution path is placed one part after another.

Examples

This is an example of a subquery which can be evaluated before the main execution:

```
select cruise_id, destination_harbor
from cruise
where cruise_price < (select min(cruise_price)
               from cruise
               where id_yacht < 145);
********   Start of EXPLAIN logging   ********

for (L2 ;SAGTOURS.CRUISE-0; L2 )  /* GET NEXT */
{
 (((SAGTOURS.CRUISE.ID_YACHT < 145 )))
}
for (L2 ;SAGTOURS.CRUISE-0; L2 )  /* GET NEXT */
{
  (((SAGTOURS.CRUISE.CRUISE_PRICE < {pre-evaluated subquery})))

}

********    End of EXPLAIN logging    ********
```

This is an example of a level 1 subtable:

```
select city_name, population
from cities;
********   Start of EXPLAIN logging   ********

for (L2 ;SAGTOURS.CITIES-0; L2 )  /* GET NEXT */
{
  for (every occurrence SAGTOURS.CITIES-1; )
  {
    if (buffer exhausted)
      L1  refill

  }
}

********    End of EXPLAIN logging    ********
```

Additional Logging Facilities

The ADABAS SQL Server was designed and implemented to be easily portable to different platforms. This goal was achieved by defining a special software layer named VO (Virtual Operating-system). All ADABAS SQL Server internal systems calls are executed by this layer. In special cases, it might be useful to monitor the response codes of the real platform specific system calls. To allow this, the VO layer offers a special logging facility.

System Call Logging (VO Logging)

System Call Logging provides information about operating system calls carried out by VO that returned an erroneous response code. The operating system messages logged then may just be “normal” responses, for example, if the ADABAS SQL Server checks for the existence of a file and this file does not exist (see example below). But logging may also give precious hints as to operating system problems (for example, incorrect kernel parameters on UNIX) resulting in ADABAS SQL Server response codes such as:

ESQ1360	operating system call failed
ESQ1308	dynamic memory allocation failed
ESQ8002	dynamic memory allocation failed during statement execution
ESQ9657	buffer manager serialization call failed
ESQ9770	buffer manager internal buffer creation failed

or similar errors.

Example:

```
%set VOLOG=ON
%esqint
%SAGVO-I-INFO, pid 174, VO trace enabled via PT_TRACELEV: i9999
%SAGVO-I-INFO, pid 174, VO system call error logging enabled via VOLOG
%ESQINT-I-STARTED, 16-APR-1998 15:35:51, Version 1.4/2.13 (INTEL-80386/WINDOWS
NT)

ESQ User: dba
Password: %SAGVO-I-INFO, > vo_dld
%SAGVO-I-INFO, library <ESQTHS> function <rdfdrv>
%SAGVO-I-INFO,   > dyn_load
%SAGVO-I-INFO, Searching in library : G:\SAG\esq\v142d\bin\esqths.dll
%SAGVO-I-INFO,   < dyn_load
%SAGVO-I-INFO, dyn_load() returned 030C2BA3
%SAGVO-I-INFO, < vo_dld
%ESQINT-I-CONNECT, Server DEMOSERV connected successfully
esqint: read input_file.dat
%SAGVO-I-INFO, pid 174, vo_fof[4]: CreateFile errno=2 The system cannot find
the file specified.

%ESQINT-E-OPENERR, Error opening input file input_file.dat
esqint: quit
%ESQINT-I-TERMINATED, 16-APR-1998 15:36:03
```

ADABAS Command Logging

In terms of ADABAS, the ADABAS SQL Server is an application program which processes SQL requests. To process SQL requests the ADABAS SQL Server makes use of the standard ADABAS call interface. A detailed discussion of the interaction with ADABAS is given in the*ADABAS SQL Server Programmer's Guide*, **Appendix D**.

To allow the monitoring and analysis of an application, the ADABAS SQL Server offers the possibility to log the usage of ADABAS. In addition to this, the ADABAS command logging may be used for trouble shooting.

Due to its internal structure, the ADABAS SQL Server has two software layers which logically interface to ADABAS:

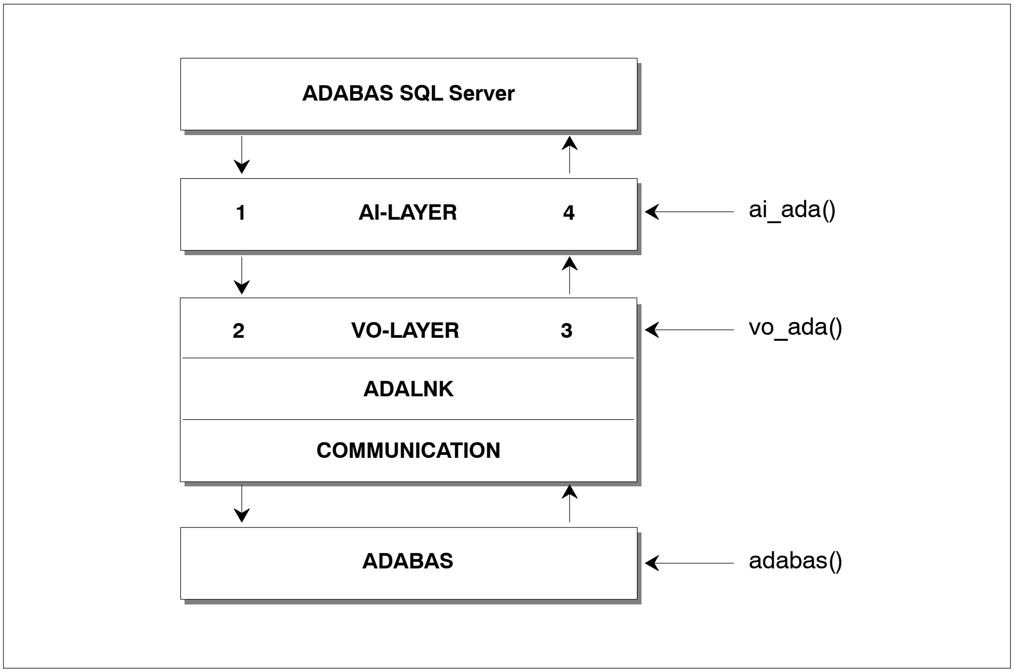
- The AI (ADABAS Interface) layer

  This layer maps CLIENT contexts to the matching ADABAS session contexts ("User queue elements") and performs various optimizations (Multifetch, RC optimization, and so on.)

- The VO (Virtual Operating-system) layer

  This layer performs the real "call ADABAS()" for different contexts.

Based on this, the ADABAS SQL Server offers four logging points:



Log Point(1)	Upon entering the AI layer	(AI IN)
Log Point(2)	Upon entering the VO layer	(VO IN)
Log Point(3)	Upon leaving the VO layer	(VO OUT)
Log Point(4)	Upon leaving the AI layer	(AI OUT)

All the above log points offer the standard ADABAS call interface, consisting of:

Control Block	(CB)
Format Buffer	(FB)
Record Buffer	(RB)
Search Buffer	(SB)
Value Buffer	(VB)
ISN Buffer	(IB)

Each buffer can be displayed in part or full.

Note:
Remember that, in case of Multifetch, the buffers may be really huge (up to 32KB). So great care should be taken if full buffer logging is chosen.

Additionally, there is also an ADABAS command log (CLOG) style logging available. This logging is done a the log point “VO OUT” (right after returning from the real “call ADABAS()”) and provides a compact yet informative logging.

Enabling ADABAS SQL Server ADABAS Command Logging

The ADABAS SQL Server ADABAS command logging is enabled and controlled via the ADABAS SQL Server parameter file. Depending on whether logging is to be enabled for the complete server or just a particular client, it is marked in the server’s or in the client’s parameter file in the GLOBAL section.

The active logging is defined as a union of the server’s parameter settings and the client’s parameter settings.

Example: Enabling ADABAS CLOG-style logging on server side

```
/*#############################################################
*
*  File name  : esqsrv.par
*
*  Description: ADABAS SQL Server Server parameter file
*
 #############################################################*/

    GLOBAL
       BEGIN
       ...
       ADABAS LOG (CLOG)
       ...
    END
```

Syntax examples for ADABAS logging specification:

```
ADABAS LOG (CLOG)
```

Does a compact logging at the “VO OUT” log point.

```
ADABAS LOG (CLOG, VO OUT(CB(FULL)))
```

Does a compact logging at the “VO OUT” log point. Additionally, the ADABAS CB is logged at the “VO IN” log point.

```
ADABAS LOG (AI OUT (CB(10,10),SB(FULL),RB(FULL)))
```

At the log point “AI OUT”, the first 10 and the last 10 Bytes of the CB, and the whole SB and RB are logged.

```
ADABAS LOG (AI IN( FULL ))
```

At the log point “AI IN”, all buffers (CB,FB,RB,SB,IB,VB) are logged in the full length.

```
ADABAS LOG (FULL)
```

At all four log point, all buffers are logged in full length.

Warning:
This may result in a dramatic logging output.

I/O Channel for ADABAS Logging Data

The logging information is written to the logical channel “operator console”, which is mapped to STDOUT. The logging may be redirected either via input/output redirection or by using the REDIRECT setting for ESQLOG. The latter may be used if STDOUT is used for a mask-driven user interface.

Examples of an ADABAS Command Log

1. ADABAS LOG (CLOG)

```
%ESQINT-I-STARTED, 16-APR-1998 15:39:28, Version 1.4/2.13 (INTEL-80386/WINDOWS N
T)
%ESQINT-I-PAR, Using parameter file esqrun.par
ESQ User: john
Password: 16-APR 15:39:34.75 Th0: CLOG 172 *ESQRTS  1 OP..   5/  0 ....   0 RB=”
.”
16-APR 15:39:34.76 Th0: CLOG 172 *ESQRTS  2 LF S   5/ 20        0 ISN:0
16-APR 15:39:34.76 Th0: CLOG 172 *ESQRTS  3 LF S   5/ 21        0 ISN:0
16-APR 15:39:34.77 Th0: CLOG 172 *ESQRTS  4 L2M    5/ 20 IP01+  0 ISN:1
16-APR 15:39:34.78 Th0: CLOG 172 *ESQRTS  1 S1 I   5/ 20 IM00+  0 ISQ:1
16-APR 15:39:34.81 Th0: CLOG 172 *JOHN    2 OP..   5/  0 ....   0 RB=”.”
%ESQINT-I-CONNECT, Server DEMOSERV connected successfully
esqint: select * from information_schema.users;
16-APR 15:39:40.46 Th0: CLOG 172 *ESQRTS  1 L3 V   5/ 22 IRMP+  0 ISN:1
16-APR 15:39:40.47 Th0: CLOG 172 *ESQRTS  2 S1 I   5/ 20 IA00+  0 ISQ:1
16-APR 15:39:40.47 Th0: CLOG 172 *ESQRTS  3 S1 I   5/ 20 IB00   0 ISQ:1
16-APR 15:39:40.47 Th0: CLOG 172 *ESQRTS  4 S2 I   5/ 20 IB00+  0 ISQ:1
16-APR 15:39:40.48 Th0: CLOG 172 *ESQRTS  5 L1MN   5/ 20 IB00+  0 ISN:2313
16-APR 15:39:40.48 Th0: CLOG 172 *ESQRTS  6 RCS    5/ 20 IB00+  0 ISN:0
16-APR 15:39:40.49 Th0: CLOG 172 *ESQRTS  7 S1 I   5/ 20 IA00+  0 ISQ:1
16-APR 15:39:40.50 Th0: CLOG 172 *ESQRTS  8 S1 I   5/ 20 IB00   0 ISQ:113
16-APR 15:39:40.53 Th0: CLOG 172 *ESQRTS  9 S2 I   5/ 20 IB00+  0 ISQ:113
16-APR 15:39:40.54 Th0: CLOG 172 *ESQRTS 10 L1MN   5/ 20 IB00+  0 ISN:10
16-APR 15:39:40.55 Th0: CLOG 172 *ESQRTS 11 L1MN   5/ 20 IB00+  0 ISN:28
16-APR 15:39:40.56 Th0: CLOG 172 *ESQRTS 12 L1MN   5/ 20 IB00+  0 ISN:48
16-APR 15:39:40.58 Th0: CLOG 172 *ESQRTS 13 L1MN   5/ 20 IB00+  0 ISN:68
16-APR 15:39:40.59 Th0: CLOG 172 *ESQRTS 14 L1MN   5/ 20 IB00+  0 ISN:84
16-APR 15:39:40.60 Th0: CLOG 172 *ESQRTS 15 L1MN   5/ 20 IB00+  0 ISN:100
16-APR 15:39:40.62 Th0: CLOG 172 *ESQRTS 16 L1MN   5/ 20 IB00+  0 ISN:116
16-APR 15:39:40.62 Th0: CLOG 172 *ESQRTS 17 L1MN   5/ 20 IB00+  0 ISN:132
16-APR 15:39:40.62 Th0: CLOG 172 *ESQRTS 18 RCS    5/ 20 IB00+  0 ISN:0
16-APR 15:39:40.63 Th0: CLOG 172 *JOHN   19 LF.S   5/ 20 ....   0 ISN:0
16-APR 15:39:40.65 Th0: CLOG 172 *ESQRTS  1 L3 V   5/ 22 IRMP+  0 ISN:2
16-APR 15:39:40.66 Th0: CLOG 172 *ESQRTS  1 L3 V   5/ 22 IRMP+  0 ISN:3
16-APR 15:39:40.66 Th0: CLOG 172 *ESQRTS  1 L3 V   5/ 22 IRMP+  0 ISN:28
16-APR 15:39:40.67 Th0: CLOG 172 *ESQRTS  1 L3 V   5/ 22 IRMP+  0 ISN:29
16-APR 15:39:40.68 Th0: CLOG 172 *ESQRTS  2 S1 I   5/ 20 IH00+  0 ISQ:0
16-APR 15:39:40.68 Th0: CLOG 172 *ESQRTS  3 S1 I   5/ 20 IH00+  0 ISQ:1
16-APR 15:39:40.69 Th0: CLOG 172 *JOHN    1 S1     5/ 20 S001   0 ISQ:4
16-APR 15:39:40.69 Th0: CLOG 172 *JOHN    2 L1MN   5/ 20 S001   0 ISN:2451

 USER_ID
 DBA
 ESQ
 FRED
 JOHN
---EOF---

output (col 1-79 of 33): q
esqint: q
16-APR 15:39:45.49 Th0: CLOG 172 *JOHN    1 BT     5/  0 ....   0 ISN:0
16-APR 15:39:45.51 Th0: CLOG 172 *JOHN    2 CL     5/  0 ....   0 ISN:0
16-APR 15:39:45.55 Th0: CLOG 172 *ESQRTS  3 CL     5/  0 ....   0 ISN:0
%ESQINT-I-TERMINATED, 16-APR-1998 15:39:45
```

Discussion of CLOG-style Logging

Each executed ADABAS call is logged with one clog line. The clog line can be read as follows:

```
30-AUG 13:39:29.81 Th0: CLOG   24934 *ESQRTS  5 L1MN 215/230 IB00+  0 ISN:2232
                     ^  ^      ^     ^        ^ ^ ^^ ^   ^   ^      ^ ^
                     |  |      |     |        | | || |   |   |      | |
        Thread id ---+  |      |     |        | | || |   |   |      | |
        CLOG style  ----+      |     |        | | || |   |   |      | |
        ESQ session id   ------+     |        | | || |   |   |      | |
        ADABAS session context  -----+        | | || |   |   |      | |
        ADABAS call counter/ESQ request  -----+ | || |   |   |      | |
        ADABAS command code   ------------------+ || |   |   |      | |
        (Preceeding ’U’ indicates ADABAS          || |   |   |      | |
        Utility call context)                     || |   |   |      | |
        Command option 1/2  ----------------------++ |   |   |      | |
        Accessed ADABAS File number/ dbid   ---------+---+   |      | |
        Used ADABAS command id    ---------------------------+      | |
        (Succeeding ’+’ marks global Format ID)                     | |
        ADABAS response code on call   -----------------------------+ |
        Either current ISN or ISN quantity or Record Buffer  ---------+
```

2. ADABAS LOG (CLOG, VO IN(FB(FULL)))

```
%ESQINT-I-STARTED, 16-APR-1998 15:43:36, Version 1.4/2.13 (INTEL-80386/WINDOWS
NT)


%ESQINT-I-PAR, Using parameter file esqrun.par
ESQ User: john
Password: 16-APR 15:43:42.07 Th0: CLOG 174 *ESQRTS  1 OP..   5/  0 ....   0
RB=”
.”
16-APR 15:43:42.08 Th0: CLOG 174 *ESQRTS  2 LF S   5/ 20        0 ISN:0
16-APR 15:43:42.09 Th0: CLOG 174 *ESQRTS  3 LF S   5/ 21        0 ISN:0
%ESQRTS-I-ESQINFO, Th0: ADALOG 174 *ESQRTS > VO_ADA FB FULL (LENGTH: 300)
032BDD80 00000000:   46412C33 322C412C 46422C31 2C412C46    FA,32,A,FB,1,A,F
032BDD90 00000010:   432C382C 422C4644 2C33322C 412C4645    C,8,B,FD,32,A,FE
032BDDA0 00000020:   2C33322C 412C4646 2C312C41 2E000000    ,32,A,FF,1,A....
032BDDB0 00000030:   00000000 00000000 00000000 00000000    ................
                 14 lines identical to line above
032BDEA0 00000120:   00000000 00000000 00000000             ............
16-APR 15:43:42.13 Th0: CLOG 174 *ESQRTS  4 L2M    5/ 20 IP01+  0 ISN:1
%ESQRTS-I-ESQINFO, Th0: ADALOG 174 *ESQRTS > VO_ADA FB FULL (LENGTH: 300)
032BEB40 00000000:   53412C33 322C412C 53422C33 322C422E    SA,32,A,SB,32,B.
032BEB50 00000010:   00000000 00000000 00000000 00000000    ................
                 16 lines identical to line above
032BEC60 00000120:   00000000 00000000 00000000             ............
16-APR 15:43:42.16 Th0: CLOG 174 *ESQRTS  1 S1 I   5/ 20 IM00+  0 ISQ:1
16-APR 15:43:42.19 Th0: CLOG 174 *JOHN    2 OP..   5/  0 ....   0 RB=”.”
%ESQINT-I-CONNECT, Server DEMOSERV connected successfully
select * from information_schema.users;
%ESQRTS-I-ESQINFO, Th0: ADALOG 148 *ESQRTS > VO_ADA FB FULL (LENGTH: 57)
03268A5C 00000000:   41412C33 322C412C 41422C33 322C412C    AA,32,A,AB,32,A,
03268A6C 00000010:   41432C33 2C412C41 442C382C 422C4145    AC,3,A,AD,8,B,AE
03268A7C 00000020:   2C382C42 2C41462C 342C422C 4147312D    ,8,B,AF,4,B,AG1-
03268A8C 00000030:   31302C31 30302C42 2E                   10,100,B.
16-APR 15:45:33.67 Th0: CLOG 148 *ESQRTS  1 L3 V   5/ 22 IRMP+  0 ISN:1
%ESQRTS-I-ESQINFO, Th0: ADALOG 148 *ESQRTS > VO_ADA FB FULL (LENGTH: 300)
032BEA00 00000000:   48412C33 322C412C 48422C33 322C412C    HA,32,A,HB,32,A,
032BEA10 00000010:   48432C34 302C412C 48442C38 2C422C48    HC,40,A,HD,8,B,H
032BEA20 00000020:   452C342C 462C4846 2C342C46 2C48472C    E,4,F,HF,4,F,HG,
032BEA30 00000030:   33322C41 2C48482C 322C462C 32582C48    32,A,HH,2,F,2X,H
032BEA40 00000040:   492C342C 462C484A 432C322C 462C484A    I,4,F,HJC,2,F,HJ
032BEA50 00000050:   312D322C 3235332C 412C484B 432C322C    1-2,253,A,HKC,2,
032BEA60 00000060:   462C484B 312D322C 3235332C 412C484C    F,HK1-2,253,A,HL
032BEA70 00000070:   2C34302C 412C484D 2C312C41 2C484E2C    ,40,A,HM,1,A,HN,
032BEA80 00000080:   312C412C 484F2C31 2C412C48 502C312C    1,A,HO,1,A,HP,1,
032BEA90 00000090:   412C4851 2C312C41 2C33582C 48522C34    A,HQ,1,A,3X,HR,4
032BEAA0 000000A0:   2C462C48 532C342C 462C4854 2C342C46    ,F,HS,4,F,HT,4,F
```

```
032BEAB0 000000B0:   2C48552C 342C462C 48562C34 2C462C48    ,HU,4,F,HV,4,F,H
032BEAC0 000000C0:   572C342C 462E0000 00000000 00000000    W,4,F...........
032BEAD0 000000D0:   00000000 00000000 00000000 00000000    ................
                 4 lines identical to line above
032BEB20 00000120:   00000000 00000000 00000000             ............
16-APR 15:45:33.69 Th0: CLOG 148 *ESQRTS  2 S1 I   5/ 20 IA00+  0 ISQ:1
%ESQRTS-I-ESQINFO, Th0: ADALOG 148 *ESQRTS > VO_ADA FB FULL (LENGTH: 1)
03267E6C 00000000:   2E                                     .
16-APR 15:45:33.69 Th0: CLOG 148 *ESQRTS  3 S1 I   5/ 20 IB00   0 ISQ:1
%ESQRTS-I-ESQINFO, Th0: ADALOG 148 *ESQRTS > VO_ADA FB FULL (LENGTH: 1)
032682AC 00000000:   2E                                     .
16-APR 15:45:33.70 Th0: CLOG 148 *ESQRTS  4 S2 I   5/ 20 IB00+  0 ISQ:1
%ESQRTS-I-ESQINFO, Th0: ADALOG 148 *ESQRTS > VO_ADA FB FULL (LENGTH: 300)
032BDB00 00000000:   49412C33 322C412C 49422C33 322C412C    IA,32,A,IB,32,A,
032BDB10 00000010:   49432C33 322C412C 49442C34 2C462C49    IC,32,A,ID,4,F,I
032BDB20 00000020:   452C322C 462C4946 2C322C41 2C49472C    E,2,F,IF,2,A,IG,
032BDB30 00000030:   34302C41 2C49482C 322C412C 49492C32    40,A,IH,2,A,II,2
032BDB40 00000040:   2C462C49 4A2C3430 2C412C49 4B2C342C    ,F,IJ,40,A,IK,4,
032BDB50 00000050:   462C494C 2C342C46 2C494D43 2C322C46    F,IL,4,F,IMC,2,F
032BDB60 00000060:   2C494D31 2D322C32 35332C41 2C494E2C    ,IM1-2,253,A,IN,
032BDB70 00000070:   312C412C 494F2C31 2C412C49 502C312C    1,A,IO,1,A,IP,1,
032BDB80 00000080:   412C4951 2C312C41 2C49522C 312C412C    A,IQ,1,A,IR,1,A,
032BDB90 00000090:   33582C49 532C342C 462E0000 00000000    3X,IS,4,F.......
032BDBA0 000000A0:   00000000 00000000 00000000 00000000    ................
                 7 lines identical to line above
032BDC20 00000120:   00000000 00000000 00000000             ............
16-APR 15:45:33.70 Th0: CLOG 148 *ESQRTS  5 L1MN   5/ 20 IB00+  0 ISN:2313
16-APR 15:45:33.71 Th0: CLOG 148 *ESQRTS  6 RCS    5/ 20 IB00+  0 ISN:0
%ESQRTS-I-ESQINFO, Th0: ADALOG 148 *ESQRTS > VO_ADA FB FULL (LENGTH: 300)
032BEA00 00000000:   48412C33 322C412C 48422C33 322C412C    HA,32,A,HB,32,A,
032BEA10 00000010:   48432C34 302C412C 48442C38 2C422C48    HC,40,A,HD,8,B,H
032BEA20 00000020:   452C342C 462C4846 2C342C46 2C48472C    E,4,F,HF,4,F,HG,
032BEA30 00000030:   33322C41 2C48482C 322C462C 32582C48    32,A,HH,2,F,2X,H
032BEA40 00000040:   492C342C 462C484A 432C322C 462C484A    I,4,F,HJC,2,F,HJ
032BEA50 00000050:   312D322C 3235332C 412C484B 432C322C    1-2,253,A,HKC,2,
032BEA60 00000060:   462C484B 312D322C 3235332C 412C484C    F,HK1-2,253,A,HL
032BEA70 00000070:   2C34302C 412C484D 2C312C41 2C484E2C    ,40,A,HM,1,A,HN,
032BEA80 00000080:   312C412C 484F2C31 2C412C48 502C312C    1,A,HO,1,A,HP,1,
032BEA90 00000090:   412C4851 2C312C41 2C33582C 48522C34    A,HQ,1,A,3X,HR,4
032BEAA0 000000A0:   2C462C48 532C342C 462C4854 2C342C46    ,F,HS,4,F,HT,4,F
032BEAB0 000000B0:   2C48552C 342C462C 48562C34 2C462C48    ,HU,4,F,HV,4,F,H
032BEAC0 000000C0:   572C342C 462E0000 00000000 00000000    W,4,F...........
032BEAD0 000000D0:   00000000 00000000 00000000 00000000    ................
                 4 lines identical to line above
032BEB20 00000120:   00000000 00000000 00000000             ............
16-APR 15:45:33.72 Th0: CLOG 148 *ESQRTS  7 S1 I   5/ 20 IA00+  0 ISQ:1
%ESQRTS-I-ESQINFO, Th0: ADALOG 148 *ESQRTS > VO_ADA FB FULL (LENGTH: 1)
03267E6C 00000000:   2E                                     .
```

```
16-APR 15:45:33.73 Th0: CLOG 148 *ESQRTS  8 S1 I   5/ 20 IB00   0 ISQ:113
%ESQRTS-I-ESQINFO, Th0: ADALOG 148 *ESQRTS > VO_ADA FB FULL (LENGTH: 1)
032682AC 00000000:   2E                                     .
16-APR 15:45:33.76 Th0: CLOG 148 *ESQRTS  9 S2 I   5/ 20 IB00+  0 ISQ:113
%ESQRTS-I-ESQINFO, Th0: ADALOG 148 *ESQRTS > VO_ADA FB FULL (LENGTH: 300)
032BDB00 00000000:   49412C33 322C412C 49422C33 322C412C    IA,32,A,IB,32,A,
032BDB10 00000010:   49432C33 322C412C 49442C34 2C462C49    IC,32,A,ID,4,F,I
032BDB20 00000020:   452C322C 462C4946 2C322C41 2C49472C    E,2,F,IF,2,A,IG,
032BDB30 00000030:   34302C41 2C49482C 322C412C 49492C32    40,A,IH,2,A,II,2
032BDB40 00000040:   2C462C49 4A2C3430 2C412C49 4B2C342C    ,F,IJ,40,A,IK,4,
032BDB50 00000050:   462C494C 2C342C46 2C494D43 2C322C46    F,IL,4,F,IMC,2,F
032BDB60 00000060:   2C494D31 2D322C32 35332C41 2C494E2C    ,IM1-2,253,A,IN,
032BDB70 00000070:   312C412C 494F2C31 2C412C49 502C312C    1,A,IO,1,A,IP,1,
032BDB80 00000080:   412C4951 2C312C41 2C49522C 312C412C    A,IQ,1,A,IR,1,A,
032BDB90 00000090:   33582C49 532C342C 462E0000 00000000    3X,IS,4,F.......
032BDBA0 000000A0:   00000000 00000000 00000000 00000000    ................
                 7 lines identical to line above
032BDC20 00000120:   00000000 00000000 00000000             ............
16-APR 15:45:33.77 Th0: CLOG 148 *ESQRTS 10 L1MN   5/ 20 IB00+  0 ISN:10
%ESQRTS-I-ESQINFO, Th0: ADALOG 148 *ESQRTS > VO_ADA FB FULL (LENGTH: 300)
032BDB00 00000000:   49412C33 322C412C 49422C33 322C412C    IA,32,A,IB,32,A,
032BDB10 00000010:   49432C33 322C412C 49442C34 2C462C49    IC,32,A,ID,4,F,I
032BDB20 00000020:   452C322C 462C4946 2C322C41 2C49472C    E,2,F,IF,2,A,IG,
032BDB30 00000030:   34302C41 2C49482C 322C412C 49492C32    40,A,IH,2,A,II,2
032BDB40 00000040:   2C462C49 4A2C3430 2C412C49 4B2C342C    ,F,IJ,40,A,IK,4,
032BDB50 00000050:   462C494C 2C342C46 2C494D43 2C322C46    F,IL,4,F,IMC,2,F
032BDB60 00000060:   2C494D31 2D322C32 35332C41 2C494E2C    ,IM1-2,253,A,IN,
032BDB70 00000070:   312C412C 494F2C31 2C412C49 502C312C    1,A,IO,1,A,IP,1,
032BDB80 00000080:   412C4951 2C312C41 2C49522C 312C412C    A,IQ,1,A,IR,1,A,
032BDB90 00000090:   33582C49 532C342C 462E0000 00000000    3X,IS,4,F.......
032BDBA0 000000A0:   00000000 00000000 00000000 00000000    ................
                 7 lines identical to line above
032BDC20 00000120:   00000000 00000000 00000000             ............
16-APR 15:45:33.79 Th0: CLOG 148 *ESQRTS 11 L1MN   5/ 20 IB00+  0 ISN:28
%ESQRTS-I-ESQINFO, Th0: ADALOG 148 *ESQRTS > VO_ADA FB FULL (LENGTH: 300)
032BDB00 00000000:   49412C33 322C412C 49422C33 322C412C    IA,32,A,IB,32,A,
032BDB10 00000010:   49432C33 322C412C 49442C34 2C462C49    IC,32,A,ID,4,F,I
032BDB20 00000020:   452C322C 462C4946 2C322C41 2C49472C    E,2,F,IF,2,A,IG,
032BDB30 00000030:   34302C41 2C49482C 322C412C 49492C32    40,A,IH,2,A,II,2
032BDB40 00000040:   2C462C49 4A2C3430 2C412C49 4B2C342C    ,F,IJ,40,A,IK,4,
032BDB50 00000050:   462C494C 2C342C46 2C494D43 2C322C46    F,IL,4,F,IMC,2,F
032BDB60 00000060:   2C494D31 2D322C32 35332C41 2C494E2C    ,IM1-2,253,A,IN,
032BDB70 00000070:   312C412C 494F2C31 2C412C49 502C312C    1,A,IO,1,A,IP,1,
032BDB80 00000080:   412C4951 2C312C41 2C49522C 312C412C    A,IQ,1,A,IR,1,A,
032BDB90 00000090:   33582C49 532C342C 462E0000 00000000    3X,IS,4,F.......
032BDBA0 000000A0:   00000000 00000000 00000000 00000000    ................
                 7 lines identical to line above
032BDC20 00000120:   00000000 00000000 00000000             ............
```

```
16-APR 15:45:33.81 Th0: CLOG 148 *ESQRTS 12 L1MN   5/ 20 IB00+  0 ISN:48
%ESQRTS-I-ESQINFO, Th0: ADALOG 148 *ESQRTS > VO_ADA FB FULL (LENGTH: 300)
032BDB00 00000000:   49412C33 322C412C 49422C33 322C412C    IA,32,A,IB,32,A,
032BDB10 00000010:   49432C33 322C412C 49442C34 2C462C49    IC,32,A,ID,4,F,I
032BDB20 00000020:   452C322C 462C4946 2C322C41 2C49472C    E,2,F,IF,2,A,IG,
032BDB30 00000030:   34302C41 2C49482C 322C412C 49492C32    40,A,IH,2,A,II,2
032BDB40 00000040:   2C462C49 4A2C3430 2C412C49 4B2C342C    ,F,IJ,40,A,IK,4,
032BDB50 00000050:   462C494C 2C342C46 2C494D43 2C322C46    F,IL,4,F,IMC,2,F
032BDB60 00000060:   2C494D31 2D322C32 35332C41 2C494E2C    ,IM1-2,253,A,IN,
032BDB70 00000070:   312C412C 494F2C31 2C412C49 502C312C    1,A,IO,1,A,IP,1,
032BDB80 00000080:   412C4951 2C312C41 2C49522C 312C412C    A,IQ,1,A,IR,1,A,
032BDB90 00000090:   33582C49 532C342C 462E0000 00000000    3X,IS,4,F.......
032BDBA0 000000A0:   00000000 00000000 00000000 00000000    ................
                  7 lines identical to line above
032BDC20 00000120:   00000000 00000000 00000000             ............
16-APR 15:45:33.83 Th0: CLOG 148 *ESQRTS 13 L1MN   5/ 20 IB00+  0 ISN:68
%ESQRTS-I-ESQINFO, Th0: ADALOG 148 *ESQRTS > VO_ADA FB FULL (LENGTH: 300)
032BDB00 00000000:   49412C33 322C412C 49422C33 322C412C    IA,32,A,IB,32,A,
032BDB10 00000010:   49432C33 322C412C 49442C34 2C462C49    IC,32,A,ID,4,F,I
032BDB20 00000020:   452C322C 462C4946 2C322C41 2C49472C    E,2,F,IF,2,A,IG,
032BDB30 00000030:   34302C41 2C49482C 322C412C 49492C32    40,A,IH,2,A,II,2
032BDB40 00000040:   2C462C49 4A2C3430 2C412C49 4B2C342C    ,F,IJ,40,A,IK,4,
032BDB50 00000050:   462C494C 2C342C46 2C494D43 2C322C46    F,IL,4,F,IMC,2,F
032BDB60 00000060:   2C494D31 2D322C32 35332C41 2C494E2C    ,IM1-2,253,A,IN,
032BDB70 00000070:   312C412C 494F2C31 2C412C49 502C312C    1,A,IO,1,A,IP,1,
032BDB80 00000080:   412C4951 2C312C41 2C49522C 312C412C    A,IQ,1,A,IR,1,A,
032BDB90 00000090:   33582C49 532C342C 462E0000 00000000    3X,IS,4,F.......
032BDBA0 000000A0:   00000000 00000000 00000000 00000000    ................
                  7 lines identical to line above
032BDC20 00000120:   00000000 00000000 00000000             ............
16-APR 15:45:33.84 Th0: CLOG 148 *ESQRTS 14 L1MN   5/ 20 IB00+  0 ISN:84
%ESQRTS-I-ESQINFO, Th0: ADALOG 148 *ESQRTS > VO_ADA FB FULL (LENGTH: 300)
032BDB00 00000000:   49412C33 322C412C 49422C33 322C412C    IA,32,A,IB,32,A,
032BDB10 00000010:   49432C33 322C412C 49442C34 2C462C49    IC,32,A,ID,4,F,I
032BDB20 00000020:   452C322C 462C4946 2C322C41 2C49472C    E,2,F,IF,2,A,IG,
032BDB30 00000030:   34302C41 2C49482C 322C412C 49492C32    40,A,IH,2,A,II,2
032BDB40 00000040:   2C462C49 4A2C3430 2C412C49 4B2C342C    ,F,IJ,40,A,IK,4,
032BDB50 00000050:   462C494C 2C342C46 2C494D43 2C322C46    F,IL,4,F,IMC,2,F
032BDB60 00000060:   2C494D31 2D322C32 35332C41 2C494E2C    ,IM1-2,253,A,IN,
032BDB70 00000070:   312C412C 494F2C31 2C412C49 502C312C    1,A,IO,1,A,IP,1,
032BDB80 00000080:   412C4951 2C312C41 2C49522C 312C412C    A,IQ,1,A,IR,1,A,
032BDB90 00000090:   33582C49 532C342C 462E0000 00000000    3X,IS,4,F.......
032BDBA0 000000A0:   00000000 00000000 00000000 00000000    ................
                  7 lines identical to line above
032BDC20 00000120:   00000000 00000000 00000000             ............
16-APR 15:45:33.86 Th0: CLOG 148 *ESQRTS 15 L1MN   5/ 20 IB00+  0 ISN:100
%ESQRTS-I-ESQINFO, Th0: ADALOG 148 *ESQRTS > VO_ADA FB FULL (LENGTH: 300)
032BDB00 00000000:   49412C33 322C412C 49422C33 322C412C    IA,32,A,IB,32,A,
032BDB10 00000010:   49432C33 322C412C 49442C34 2C462C49    IC,32,A,ID,4,F,I
032BDB20 00000020:   452C322C 462C4946 2C322C41 2C49472C    E,2,F,IF,2,A,IG,
032BDB30 00000030:   34302C41 2C49482C 322C412C 49492C32    40,A,IH,2,A,II,2
032BDB40 00000040:   2C462C49 4A2C3430 2C412C49 4B2C342C    ,F,IJ,40,A,IK,4,
032BDB50 00000050:   462C494C 2C342C46 2C494D43 2C322C46    F,IL,4,F,IMC,2,F
032BDB60 00000060:   2C494D31 2D322C32 35332C41 2C494E2C    ,IM1-2,253,A,IN,
032BDB70 00000070:   312C412C 494F2C31 2C412C49 502C312C    1,A,IO,1,A,IP,1,
032BDB80 00000080:   412C4951 2C312C41 2C49522C 312C412C    A,IQ,1,A,IR,1,A,
032BDB90 00000090:   33582C49 532C342C 462E0000 00000000    3X,IS,4,F.......
032BDBA0 000000A0:   00000000 00000000 00000000 00000000    ................
                 7 lines identical to line above
032BDC20 00000120:   00000000 00000000 00000000             ............
16-APR 15:45:33.88 Th0: CLOG 148 *ESQRTS 16 L1MN   5/ 20 IB00+  0 ISN:116
%ESQRTS-I-ESQINFO, Th0: ADALOG 148 *ESQRTS > VO_ADA FB FULL (LENGTH: 300)
032BDB00 00000000:   49412C33 322C412C 49422C33 322C412C    IA,32,A,IB,32,A,
032BDB10 00000010:   49432C33 322C412C 49442C34 2C462C49    IC,32,A,ID,4,F,I
032BDB20 00000020:   452C322C 462C4946 2C322C41 2C49472C    E,2,F,IF,2,A,IG,
032BDB30 00000030:   34302C41 2C49482C 322C412C 49492C32    40,A,IH,2,A,II,2
032BDB40 00000040:   2C462C49 4A2C3430 2C412C49 4B2C342C    ,F,IJ,40,A,IK,4,
032BDB50 00000050:   462C494C 2C342C46 2C494D43 2C322C46    F,IL,4,F,IMC,2,F
032BDB60 00000060:   2C494D31 2D322C32 35332C41 2C494E2C    ,IM1-2,253,A,IN,
032BDB70 00000070:   312C412C 494F2C31 2C412C49 502C312C    1,A,IO,1,A,IP,1,
032BDB80 00000080:   412C4951 2C312C41 2C49522C 312C412C    A,IQ,1,A,IR,1,A,
032BDB90 00000090:   33582C49 532C342C 462E0000 00000000    3X,IS,4,F.......
032BDBA0 000000A0:   00000000 00000000 00000000 00000000    ................
                 7 lines identical to line above
032BDC20 00000120:   00000000 00000000 00000000             ............
16-APR 15:45:33.89 Th0: CLOG 148 *ESQRTS 17 L1MN   5/ 20 IB00+  0 ISN:132
16-APR 15:45:33.89 Th0: CLOG 148 *ESQRTS 18 RCS    5/ 20 IB00+  0 ISN:0
16-APR 15:45:33.91 Th0: CLOG 148 *JOHN   19 LF.S   5/ 20 ....   0 ISN:0
%ESQRTS-I-ESQINFO, Th0: ADALOG 148 *ESQRTS > VO_ADA FB FULL (LENGTH: 57)
03268A5C 00000000:   41412C33 322C412C 41422C33 322C412C    AA,32,A,AB,32,A,
03268A6C 00000010:   41432C33 2C412C41 442C382C 422C4145    AC,3,A,AD,8,B,AE
03268A7C 00000020:   2C382C42 2C41462C 342C422C 4147312D    ,8,B,AF,4,B,AG1-
03268A8C 00000030:   31302C31 30302C42 2E                   10,100,B.
16-APR 15:45:33.93 Th0: CLOG 148 *ESQRTS  1 L3 V   5/ 22 IRMP+  0 ISN:2
%ESQRTS-I-ESQINFO, Th0: ADALOG 148 *ESQRTS > VO_ADA FB FULL (LENGTH: 57)
03268A5C 00000000:   41412C33 322C412C 41422C33 322C412C    AA,32,A,AB,32,A,
03268A6C 00000010:   41432C33 2C412C41 442C382C 422C4145    AC,3,A,AD,8,B,AE
03268A7C 00000020:   2C382C42 2C41462C 342C422C 4147312D    ,8,B,AF,4,B,AG1-
03268A8C 00000030:   31302C31 30302C42 2E                   10,100,B.
16-APR 15:45:33.93 Th0: CLOG 148 *ESQRTS  1 L3 V   5/ 22 IRMP+  0 ISN:3
%ESQRTS-I-ESQINFO, Th0: ADALOG 148 *ESQRTS > VO_ADA FB FULL (LENGTH: 57)
03268A5C 00000000:   41412C33 322C412C 41422C33 322C412C    AA,32,A,AB,32,A,
03268A6C 00000010:   41432C33 2C412C41 442C382C 422C4145    AC,3,A,AD,8,B,AE
03268A7C 00000020:   2C382C42 2C41462C 342C422C 4147312D    ,8,B,AF,4,B,AG1-
03268A8C 00000030:   31302C31 30302C42 2E                   10,100,B.
16-APR 15:45:33.94 Th0: CLOG 148 *ESQRTS  1 L3 V   5/ 22 IRMP+  0 ISN:28
```

```
%ESQRTS-I-ESQINFO, Th0: ADALOG 148 *ESQRTS > VO_ADA FB FULL (LENGTH: 57)
03268A5C 00000000:   41412C33 322C412C 41422C33 322C412C    AA,32,A,AB,32,A,
03268A6C 00000010:   41432C33 2C412C41 442C382C 422C4145    AC,3,A,AD,8,B,AE
03268A7C 00000020:   2C382C42 2C41462C 342C422C 4147312D    ,8,B,AF,4,B,AG1-
03268A8C 00000030:   31302C31 30302C42 2E                   10,100,B.
16-APR 15:45:33.95 Th0: CLOG 148 *ESQRTS  1 L3 V   5/ 22 IRMP+  0 ISN:29
%ESQRTS-I-ESQINFO, Th0: ADALOG 148 *ESQRTS > VO_ADA FB FULL (LENGTH: 300)
032BE500 00000000:   4F412C33 322C412C 4F422C33 322C412C    OA,32,A,OB,32,A,
032BE510 00000010:   4F432C33 322C412C 4F442C33 322C412C    OC,32,A,OD,32,A,
032BE520 00000020:   4F452C34 302C412C 4F462C31 2C412E00    OE,40,A,OF,1,A..
032BE530 00000030:   00000000 00000000 00000000 00000000    ................
                 14 lines identical to line above
032BE620 00000120:   00000000 00000000 00000000             ............
16-APR 15:45:33.95 Th0: CLOG 148 *ESQRTS  2 S1 I   5/ 20 IH00+  0 ISQ:0
%ESQRTS-I-ESQINFO, Th0: ADALOG 148 *ESQRTS > VO_ADA FB FULL (LENGTH: 300)
032BE500 00000000:   4F412C33 322C412C 4F422C33 322C412C    OA,32,A,OB,32,A,
032BE510 00000010:   4F432C33 322C412C 4F442C33 322C412C    OC,32,A,OD,32,A,
032BE520 00000020:   4F452C34 302C412C 4F462C31 2C412E00    OE,40,A,OF,1,A..
032BE530 00000030:   00000000 00000000 00000000 00000000    ................
                 14 lines identical to line above
032BE620 00000120:   00000000 00000000 00000000             ............
16-APR 15:45:33.96 Th0: CLOG 148 *ESQRTS  3 S1 I   5/ 20 IH00+  0 ISQ:1
%ESQRTS-I-ESQINFO, Th0: ADALOG 148 *JOHN   > VO_ADA FB FULL (LENGTH: 8)
01102904 00000000:   53412C33 322C412E                      SA,32,A.
16-APR 15:45:33.97 Th0: CLOG 148 *JOHN    1 S1     5/ 20 S001   0 ISQ:4
%ESQRTS-I-ESQINFO, Th0: ADALOG 148 *JOHN   > VO_ADA FB FULL (LENGTH: 8)
01102904 00000000:   53412C33 322C412E                      SA,32,A.
16-APR 15:45:33.97 Th0: CLOG 148 *JOHN    2 L1MN   5/ 20 S001   0 ISN:2451

 USER_ID

 DBA
 ESQ
 FRED
 JOHN
---EOF---

output (col 1-33 of 33): 16-APR 15:45:37.34 Th0: CLOG 148 *JOHN    1 BT     5/
0 ....   0 ISN:0
16-APR 15:45:37.36 Th0: CLOG 148 *JOHN    2 CL     5/  0 ....   0 ISN:0
16-APR 15:45:37.45 Th0: CLOG 148 *ESQRTS  3 CL     5/  0 ....   0 ISN:0
%ESQINT-I-TERMINATED, 16-APR-1998 15:45:37
```

APPENDIX A — THE PARAMETER PROCESSING LANGUAGE (PPL)

The Parameter Processing Language (PPL) has been developed to enable suitable settings of individual parts of the ADABAS SQL Server system. The list on the following two pages shows in which mode each parameter can be set and what the effects are.

This version of the PPL is presented in 5 logical sections. These sections are: PRECOMPILER, COMPILER, RUNTIME, GLOBAL and SERVER.

Syntax

PPL works via a text parameter file, which can be of any size (including lines/record lengths) and can contain comments.

The comments can be in the form of “C’ comments, starting with “/*’ and ending with “*/’.

Comments can also start with “<<’ and end with “>>’ or start with “––’ and end with “––’.

The file must not contain line numbers. Statements in the file can span multiple lines and there can be multiple statements on a line. If supported by the given operating system, some parameters can be specified directly in the command line.

Each logical unit of parameter settings is individually described in a block which starts with either the keyword **BEGIN** or **(** (parenthesis) with one or more settings for that block then being defined. The block is then terminated with either the keyword **END** or **)** (parenthesis). A block for the same logical section can occur several times within the parameter file.

What Happens When These Parameters Are Set?

The following table shows in which mode each parameter can be set and what the effects are.

For example, PRECOMPILER COBOL LANGUAGE SETTINGS (COBOL II = ON), if set for a client, affects that particular client while in Client/Server Mode, and, if set on the server side in the same mode, sets the default for all clients. If set while in Precompiler mode, it affects both single-user and multi-user applications. This particular setting has no effect in Runtime Mode.

Individual settings of the same parameter (for example the GLOBAL ERROR Setting with DBID, FNR and LANGUAGE) may in some cases affect the client, in other cases the server. Therefore, these parameters have been broken down to reflect each possibility.

One of the following statuses will apply:

D	=	Default for the Client
Y	=	Used in this mode
–	=	No effect / is ignored or issues a warning

Parameter Settings	**Client-Server Mode**		**Precompiler Mode**		**Runtime-Linked-In Mode**	
	Client	**Server**	**Single-user**	**Multi-user**	**Single-user**	**Multi-user**
PRECOMPILER						
C LANGUAGE	Y	D	Y	Y	–	–
COBOL LANGUAGE	Y	D	Y	Y	–	–
COMPILATION UNIT IDENTIFIER	Y	D	Y	Y	Y	Y
HOST LANGUAGE	Y	D	Y	Y	–	–
COMPILER						
MAXIMUM	Y	D	Y	Y	Y	Y
EXPECTED UPDATE	Y	D	Y	Y	Y	Y
MODE	Y	D	Y	Y	Y	Y

Parameter Settings	Client-Server Mode		Precompiler Mode		Runtime-Linked-In Mode	
	Client	Server	Single-user	Multi-user	Single-user	Multi-user
RUNTIME						
MAXIMUM	Y	D	–	–	Y	Y
ROLLBACK ON ERROR	Y	D	–	–	Y	Y
LOCK WHEN READING	Y	D	–	–	Y	Y
SERVER	Y	D	–	–	–	Y
GLOBAL						
ADABAS Settings						
– ADABAS MULTIFETCH	–	Y	Y	–	Y	Y
– ADABAS HOLD ON/OFF	Y	D	Y	Y	Y	Y
– ADABAS FREE FILE SEARCH R.	Y	D	–	–	Y	Y
– ADABAS EXU/EXCLUSIVE UPDATE	Y	D	–	–	Y	Y
MULTIFETCH	Y	D	–	–	Y	Y
BUFFER MANAGER	–	Y	Y	–	Y	–
CATALOG	–	Y	Y	–	Y	–
ERROR Settings						
– ERROR DBID	–	Y	Y	–	Y	–
– ERROR FNR	–	Y	Y	–	Y	–
– ERROR LANGUAGE	Y	Y	Y	Y	Y	Y
PREDICT						
– PREDICT DBID	–	Y	Y	–	Y	–
– PREDICT FNR	–	Y	Y	–	Y	–
– PREDICT CROSS REFERENCE	Y	D	Y	Y	Y	Y
– PREDICT CROSS REFERENCE LIB.	Y	D	Y	Y	Y	Y

Parameter Settings	**Client-Server Mode**		**Precompiler Mode**		**Runtime-Linked-In Mode**	
	Client	**Server**	**Single-user**	**Multi-user**	**Single-user**	**Multi-user**
DEFAULT SCHEMA IDENTIFIER	Y	–	Y	Y	Y	Y
FILE	Y	D	Y	Y	Y	Y
SERVER						
NAME	–	Y	–	–	–	–
THREADS	–	Y	–	–	–	–
TYPE	–	Y	–	–	–	–
MAXIMUM	–	Y	–	–	–	–

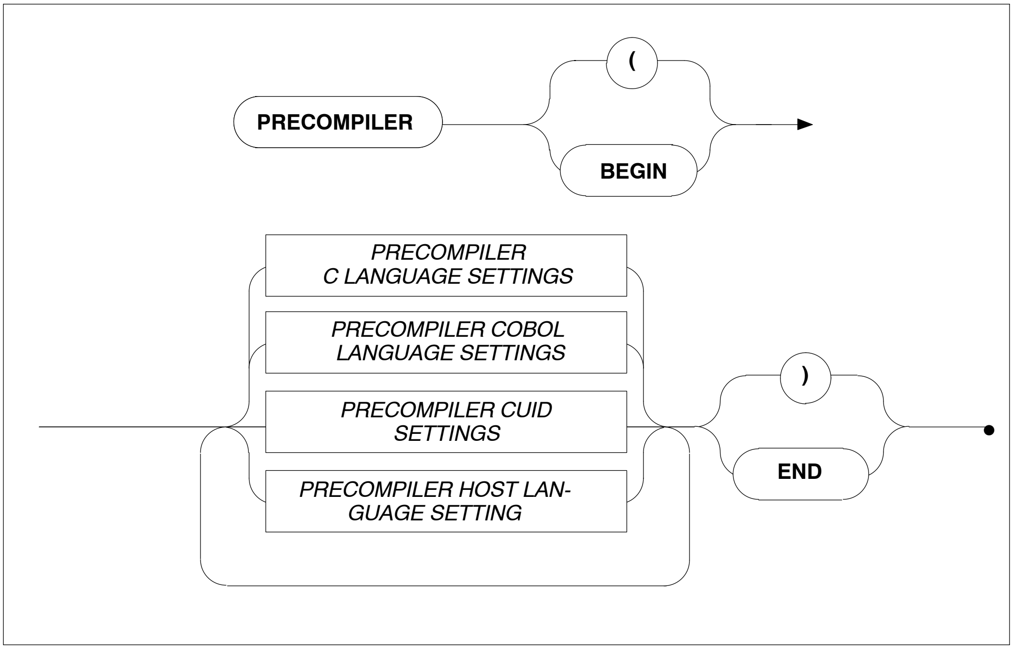
PRECOMPILER Settings

The precompiler-specific settings influence the behavior of the ADABAS SQL Server precompiler.

The types of settings for the precompiler are:

- C LANGUAGE SETTINGS
- COBOL LANGUAGE SETTINGS
- COMPILATION UNIT IDENTIFIER SETTINGS
- HOST LANGUAGE SETTINGS

SYNTAX

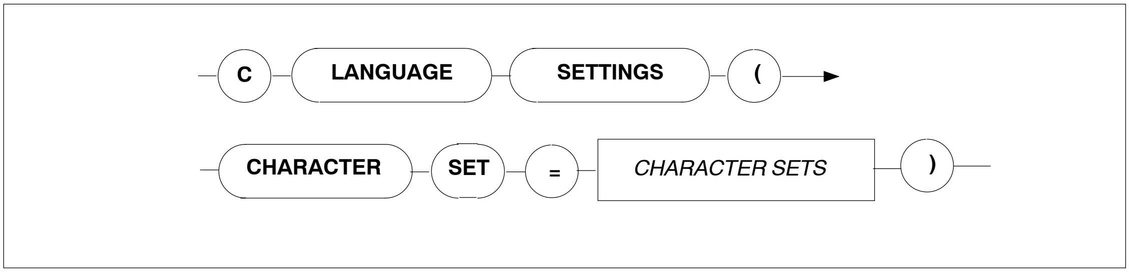

PRECOMPILER
(
BEGIN
PRECOMPILER C LANGUAGE SETTINGS
PRECOMPILER COBOL LANGUAGE SETTINGS
PRECOMPILER CUID SETTINGS
PRECOMPILER HOST LANGUAGE SETTING
)
END

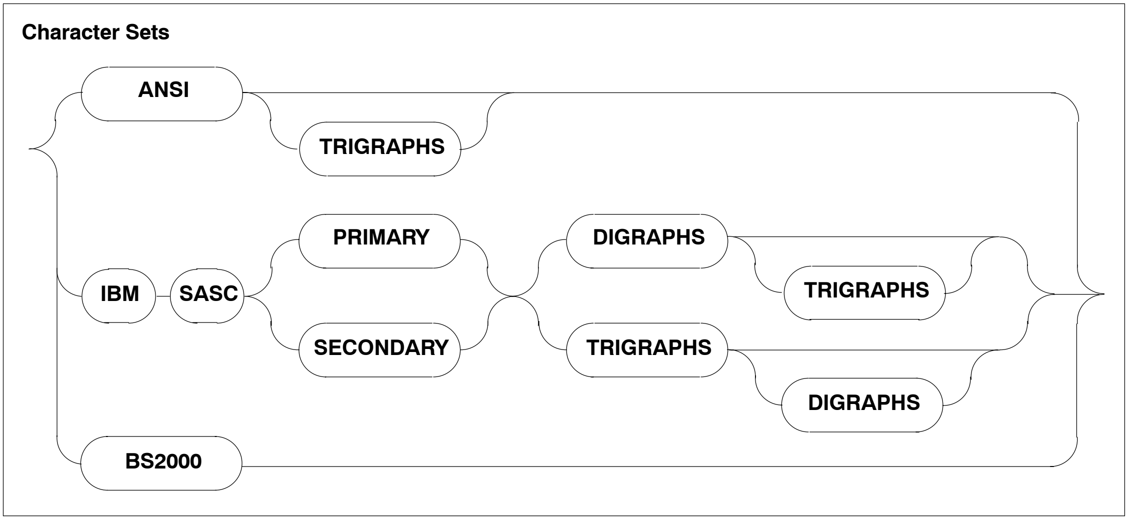
PRECOMPILER C LANGUAGE Settings

Function

To specify the C language environment.

Syntax





Description

These settings allow you to specify the character set used The character sets available are:

ANSI	Normal ASCII character set. There is an added option of TRIGRAPHS which some versions of C have.
IBM SASC PRIMARY	IBM's C compilers primary character set. There are the added options of TRIGRAPHS and DIGRAPHS.
IBM SASC SECONDARY	IBM's C compilers secondary character set. Also has TRIGRAPHS and DIGRAPHS.
BS2000	Siemens BS2000 series C compiler character set.

Limitations

None.

Examples

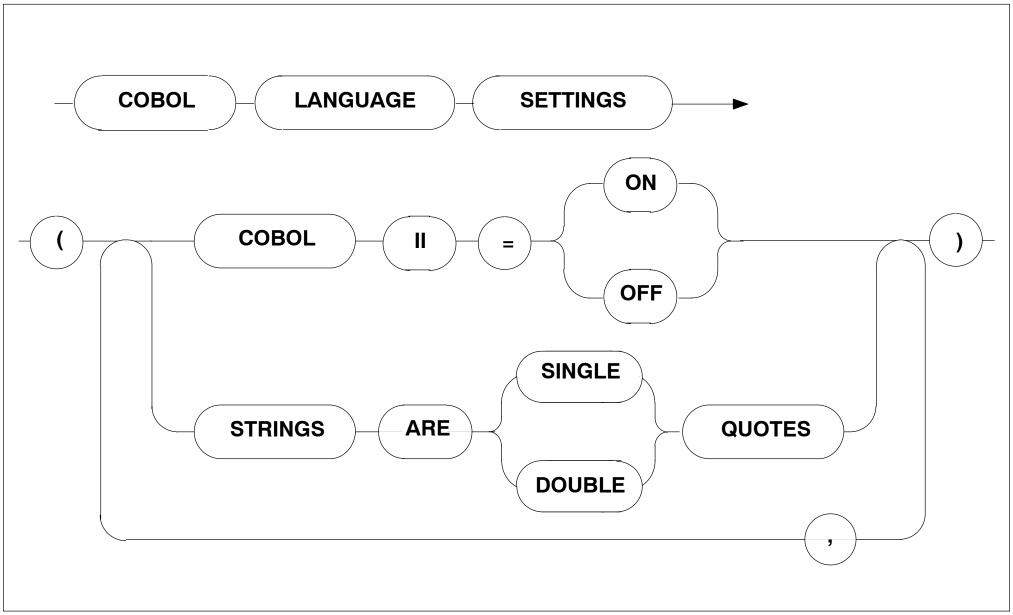
```
PRECOMPILER
BEGIN
    C LANGUAGE SETTINGS ( CHARACTER SET = ANSI TRIGRAPHS)
END

PRECOMPILER
BEGIN
    C LANGUAGE SETTINGS ( CHARACTER SET = IBM SASC PRIMARY )
END
```

PRECOMPILER COBOL LANGUAGE Settings

Function

To set whether the COBOL compiler used is COBOL II compatible or not.

Syntax



Description

If the COBOL compiler used is not COBOL II compatible, it is mandatory to set the COBOL language setting to OFF. In this case, the code generated by the ADABAS SQL SERVER COBOL Precompiler will not include the END-IF or END-PERFORM, etc. clauses and is thus compatible with COBOL as well as COBOL II. If the COBOL II compiler is used, this setting may be set to ON or may be omitted because this is also the default setting.

Furthermore, setting the STRINGS option will determine whether a COBOL string is single or double quoted. The default setting is: DOUBLE

Limitations

None

Examples

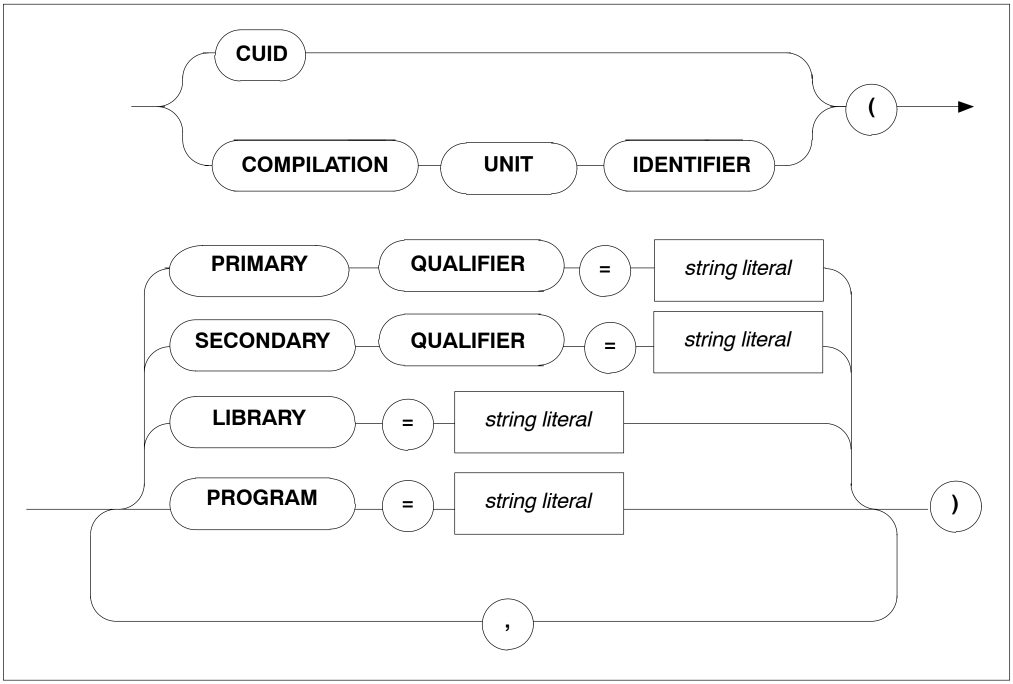
```
PRECOMPILER
BEGIN
    COBOL LANGUAGE SETTINGS ( COBOL II = ON, STRINGS ARE SINGLE QUOTES )
END
```

PRECOMPILER COMPILATION UNIT IDENTIFIER Settings

Function

To set the compilation unit ID. Specification of program is mandatory.

Syntax



Description

The compilation unit identifier is stored in a key field and is used to identify a meta program. This field consists of four user-definable elements and one host language element predefined by the precompiler:

PRIMARY QUALIFIER	any user-defined string of up to 5 characters.
SECONDARY QUALIFIER	any user-defined string of up to 5 characters.
LIBRARY	any user-defined string of up to 8 characters.
PROGRAM	any user-defined string of up to 8 characters.

Note:
The specification of the PROGRAM setting is mandatory.

The default values for the above elements are: blank

Strict naming conventions should be established to make sure that the entire key field amounts to a unique value. A new meta program with the same compilation unit identifier will overwrite an existing one in the catalog.

Limitations

None.

Examples

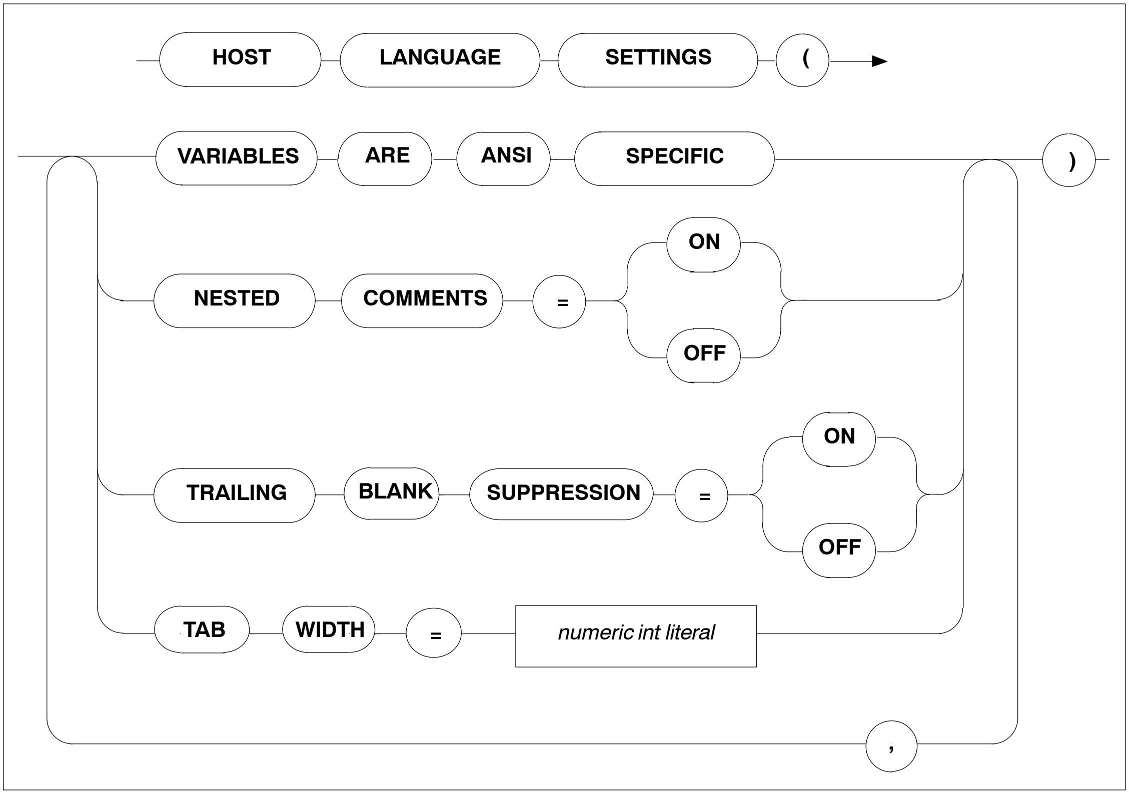
```
PRECOMPILER
BEGIN
    CUID ( LIBRARY = ’ESQ’, PROGRAM = ’CR_BASE’ )
END

PRECOMPILER
BEGIN
    CUID ( PROGRAM = “CR_ACC”, LIBRARY = ’BANKS’ )
END
```

PRECOMPILER HOST LANGUAGE Setting

Function

To specify the host language environment.

Syntax



Description

VARIABLES ARE ANSI SPECIFIC

This setting defines that the Precompiler only searches for host variables within an SQL statement when used according to the ANSI specification, i.e. :<host_var>. This is the default condition and can presently not be altered.

NEST COMMENTS

Offsetting the default, this setting defines that the host language compiler allows nested comments. The default setting is OFF.

TRAILING BLANK the SUPPRESSION

This setting defines whether trailing blank strings returned by ADABAS SQL Server are suppressed or not. The default = OFF is also the only valid setting in ANSI mode.

TAB WIDTH

This setting defines the width of a TAB character. This is particularly important for COBOL compilers when a TAB character is used to get to a specific column, i.e. column 8. The default value is 8 characters.

Limitations

None.

Examples

```
PRECOMPILER
BEGIN
    HOST LANGUAGE SETTINGS (VARIABLES ARE ANSI SPECIFIC, TAB WIDTH = 2)
END
```

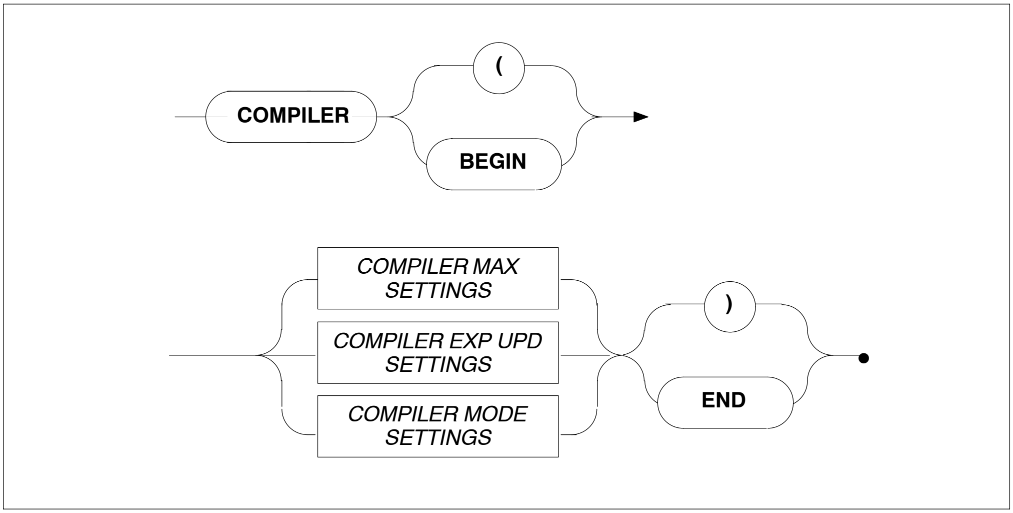
COMPILER Settings

The compiler-specific settings influence the behavior of the ADABAS SQL Server compiler.

The types of COMPILER settings are:

- MAXIMUM Settings
- EXPECTED UPDATE Setting
- MODE Settings

Syntax

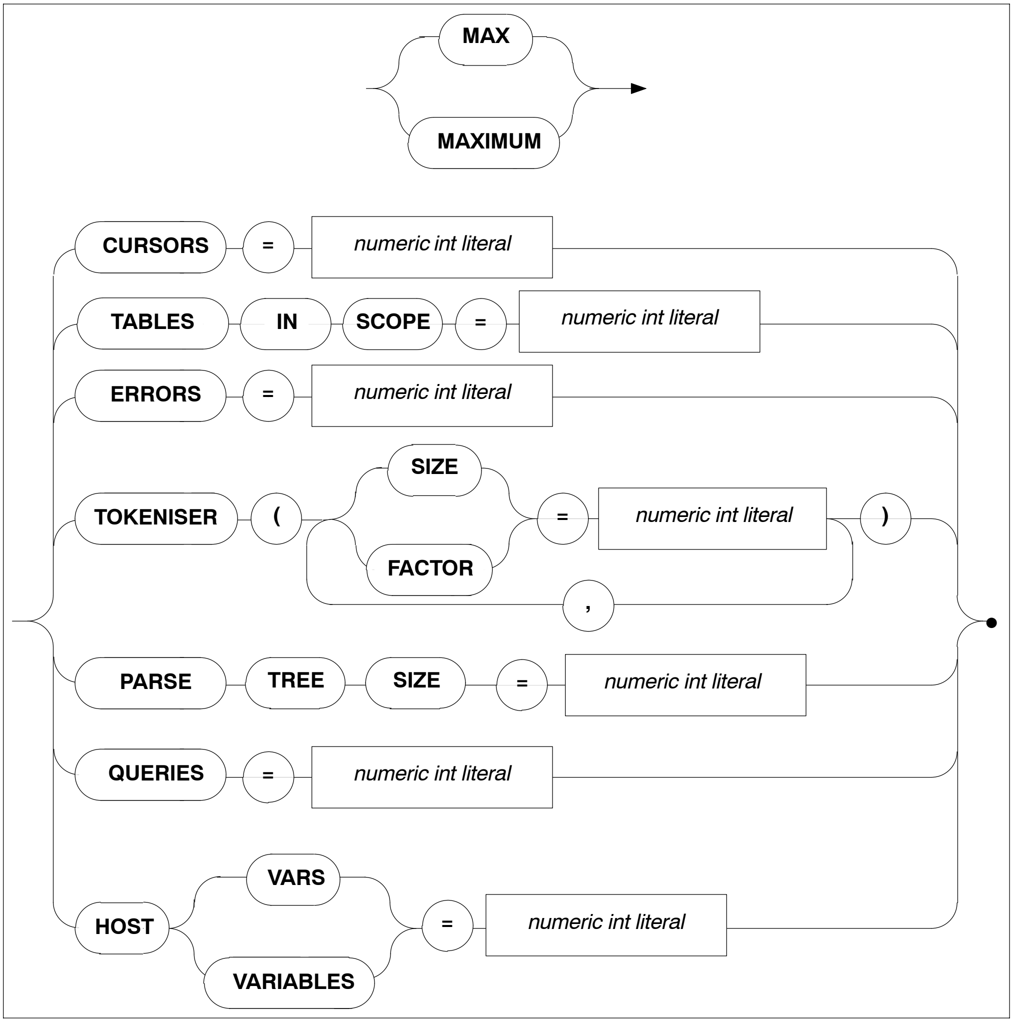


COMPILER MAXIMUM Settings

Function

To set the compile-time system's maximum values.

Syntax



Description

Allows the setting of maximum compile-time system values. The minimum value to be specified is usually 1, with the exception of maximum parse tree size whose minimum setting is 10000 bytes. The maximum value depends on the underlying hardware. Default values are as follows:

CURSORS
The number of different cursors which can be declared in one compilation unit. Default: 32

TABLES IN SCOPE
The number of tables that may be in scope within an SQL statement. Default: 32

ERRORS
The number of errors that can be logged in a compilation unit. Default: 100

PARSE TREE SIZE
The number of bytes in memory reserved for each parse tree generated by the SQL Compiler.
Default (as well as minimum value): 10,000.

TOKENISER SIZE
The size of a tokeniser buffer in bytes. Default: 4000

TOKENISER FACTOR
The number of identifiers, strings constants etc... per 100 tokens. Default: 10

QUERIES
The number of subqueries allowed within an SQL statement. Default: 32

HOST VARIABLES
The number of host variables allowed in a compilation unit. Default: 250

Limitations

The higher these values are set, the more memory will be required by the compile-time system.

If the PARSE TREE SIZE is set too small, the compiler attempts to acquire more buffer space and start parsing again, which will result in reduced performance. If the compiler fails in acquiring a larger buffer, then it will abort with a fatal error.

Examples

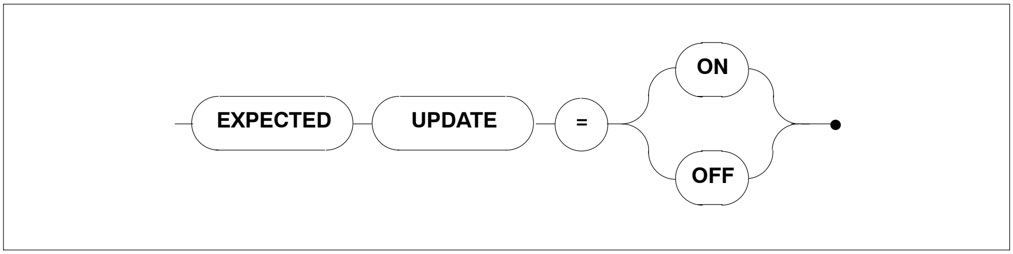
```
COMPILER
BEGIN
    MAX TOKENISER ( SIZE = 32000 )
    MAX CURSORS = 32
    MAX TABLES IN SCOPE = 15
    MAX ERRORS = 50
    MAX QUERIES = 5
    MAX HOST VARS = 32
    MAX PARSE TREE SIZE = 20000
END
```

COMPILER EXPECTED UPDATE (Cursor) Setting

Function

To set whether the default is an updatable cursor or a read-only cursor.

Syntax



Description

If a cursor is defined in one compilation unit without the FOR UPDATE clause and without any UPDATE or DELETE statements, the default results in a read-only cursor. If this cursor is referred to in another compilation unit, a runtime error will be raised. This clause alters the default for cursors.

Limitations

None.

Examples

```
COMPILER
BEGIN
     EXPECTED UPDATE = ON
END
```

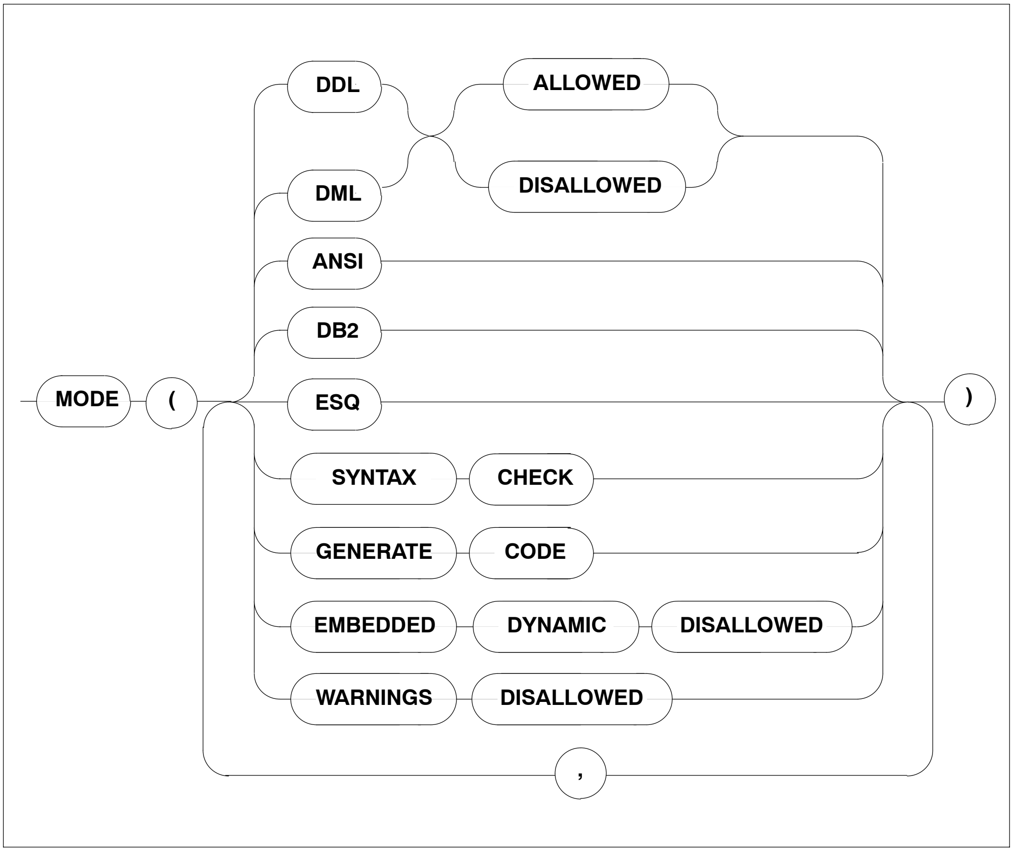
```
COMPILER
BEGIN
     EXPECTED UPDATE = OFF
END
```

COMPILER MODE Settings

Functions

To set the particular mode of operation for the compiler.

Syntax



Description

When a particular mode is set, only statements or options which conform to that mode will be permitted. For example, if DDL is disallowed, it will not be possible to compile DDL statements like CREATE TABLE or DCL statements like GRANT or REVOKE. If the default mode ESQ is set, warnings will still be issued if the ANSI standard is violated. To suppress these warnings, use the option WARNINGS DISALLOWED.

DDL
ALLOWED/DISALLOWED

all DDL/DCL statements can be explicitly permitted or forbidden. There is no special keyword for DCL but DDL implicitly includes DCL.

DML
ALLOWED/DISALLOWED

DML statements can be explicitly permitted or forbidden.

ANSI

in this mode, only statements which conform to the ANSI standard will be permitted. All deviations from the standard, i.e., ADABAS SQL Server extensions, will result in compilation errors. The same is true for embedded dynamic statements.

DB2

in this mode, only statements which conform to the DB2 syntax will be permitted. All deviations from the standard, i.e., ADABAS SQL Server extensions, will result in compilation errors.
Under CICS, implicit COMMIT or ROLLBACK statements are used at runtime at the end of each CICS task:
COMMIT in case of a normal task end and ROLLBACK in case of an abnormal task end.

ESQ

this is the ADABAS SQL Server default mode. The use of all extensions is permitted.

SYNTAX CHECK

the output of object code is suppressed and only a list of syntax errors (if any) is displayed.

GENERATE CODE

object code is produced and a list of syntax errors (if any) is displayed.

EMBEDDED DYNAMIC DISALLOWED
this setting is only valid for the default mode and explicitly excludes embedded dynamic statements.

WARNINGS DISALLOWED
warning messages are suppressed, actual error messages will be displayed.

Limitations

DDL and DML must not both be disallowed at the same time. In ANSI mode, DML and DDL must not both be allowed at the same time, while in ADABAS SQL Server and DB2 modes, this is possible.

The specification of only one mode, ANSI, DB2 or ADABAS SQL Server, is permitted per statement/ compilation unit.

Examples

```
COMPILER
BEGIN
    MODE ( DDL ALLOWED, DML ALLOWED, ESQ )
END

COMPILER
BEGIN
    MODE ( ANSI, SYNTAX CHECK )
END

COMPILER
BEGIN
    MODE ( DDL ALLOWED, DML ALLOWED, DB2, EMBEDDED DYNAMIC DISALLOWED )
END
```

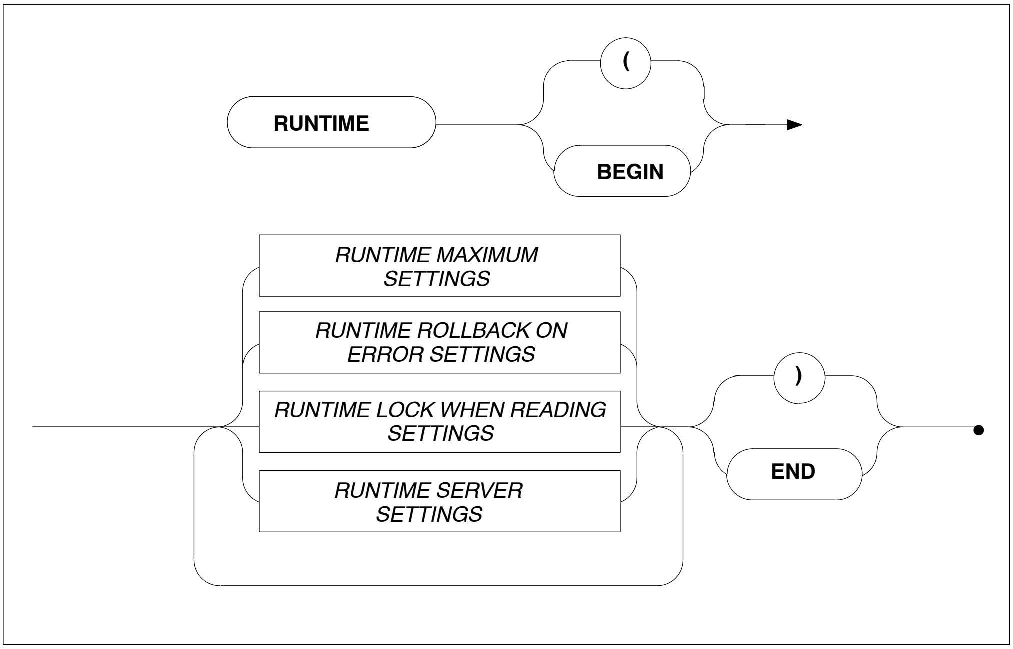
RUNTIME Settings

The Runtime System-specific settings influences the behavior of the Runtime System.

The types of settings for the Runtime System are:

- MAXIMUM RUNTIME SETTINGS
- ROLLBACK ON ERROR SETTING
- LOCK WHEN READING SETTING
- SERVER SETTINGS

Syntax

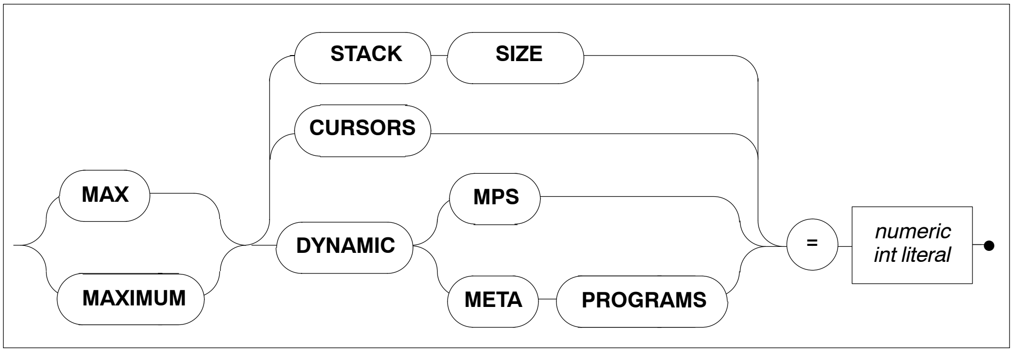

RUNTIME
(
BEGIN
RUNTIME MAXIMUM SETTINGS
RUNTIME ROLLBACK ON ERROR SETTINGS
RUNTIME LOCK WHEN READING SETTINGS
RUNTIME SERVER SETTINGS
)
END

RUNTIME MAXIMUM Settings

Function

To set the maximum Runtime System parameters.

Syntax



Description

These settings are designed so that applications with abnormal requirements do not affect other applications, for example, in memory required.

MAX STACK SIZE — The maximum size of the runtime system stack. Default value is 100, minimum is 1.

MAX CURSORS — The maximum number of cursors expected to be opened. Default value is 3.

MAX DYNAMIC MPS — The maximum number of DYNAMIC meta programs which are expected to be executed. Dynamic meta programs are only in existence while an application is running. Default value is 2.

Limitations

The higher these values are set, the more memory is required. If they are set too low, the Runtime System will terminate with errors.

Examples

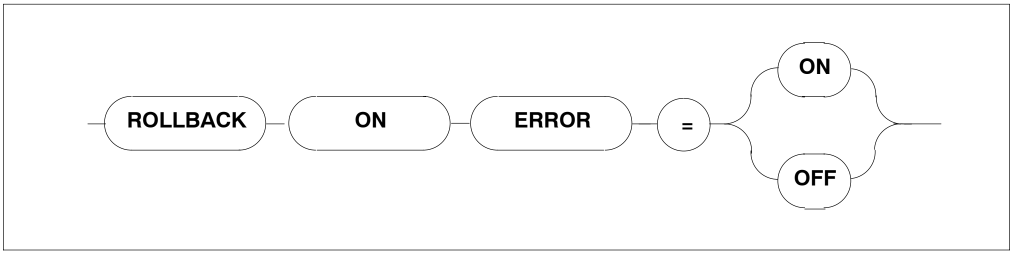
```
RUNTIME
BEGIN
    MAX STACK SIZE = 64 MAX CURSORS = 32
END
```

RUNTIME ROLLBACK ON ERROR Setting

Function

To set that a ROLLBACK occurs when an error is encountered.

Syntax



Description

If this setting is set to ON, the Runtime System issues a ROLLBACK command (backout transaction) whenever an error occurs. The default setting is OFF.

Limitations

None.

Examples

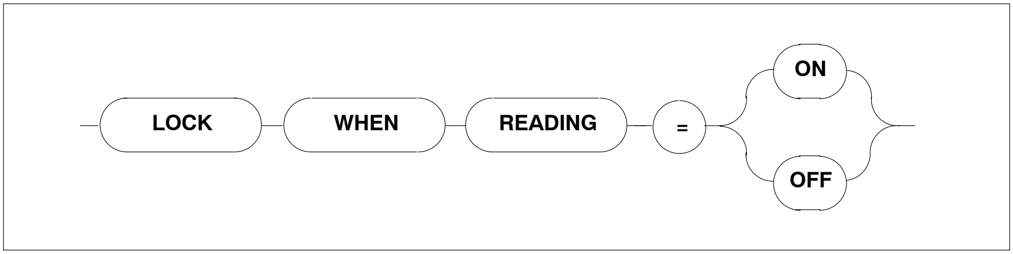
```
RUNTIME
BEGIN
     ROLLBACK ON ERROR = ON
END
```

RUNTIME LOCK WHEN READING Setting

Function

To set whether table rows are locked while they are being retrieved from the database.

Syntax



Description

If LOCK WHEN READING is set to ON, each time a table row is retrieved from the database, that particular row is locked.

If it is set to the default value = OFF, the row will not be locked.

Limitations

None.

Examples

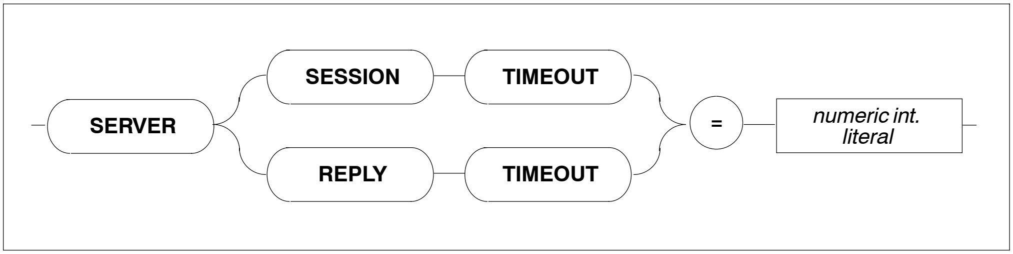
```
RUNTIME
BEGIN
    LOCK WHEN READING = ON
END

RUNTIME
BEGIN
    MAX STACK SIZE = 128 MAX CURSORS = 48 MAX DYNAMIC MPS = 32
END
```

RUNTIME SERVER Settings

Function

To set the length of a timeout for a particular client.

Syntax



Description

This setting defines the server's timeout for a particular client. Each client may set how long it takes for a timeout in minutes.

Zero minutes is infinite (∞).

SESSION TIMEOUT — The length of time a session must be idle before timing out. Default value: 15 minutes.

REPLY TIMEOUT — The length of time a reply may take before timing out. Default value: 15 minutes.

Limitations

For use in client/server mode only.

Examples

```
RUNTIME
BEGIN
    SERVER SESSION TIMEOUT = 5
END
```

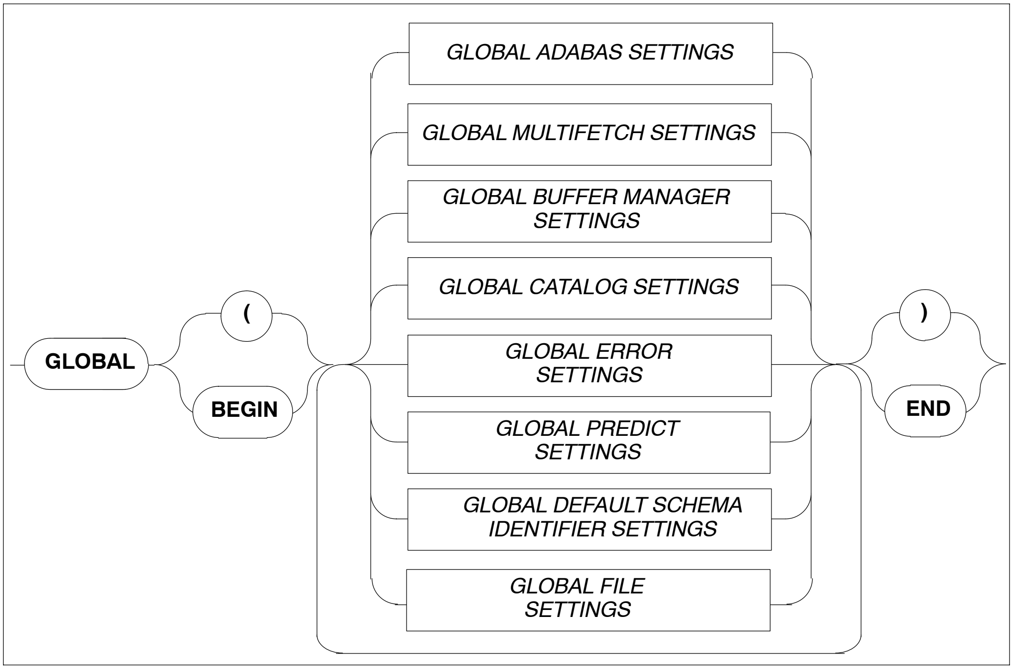
GLOBAL Settings

Within the global section, all parameters which are not limited to the precompiler, compiler, runtime system or trace facility are described. Options set here are effective throughout the entire ADABAS SQL Server system.

The types of settings for the GLOBAL section are:

- ADABAS SETTINGS
- MULTIFETCH SETTINGS
- BUFFER MANAGER SETTINGS
- CATALOG SETTINGS
- ERROR SETTINGS
- PREDICT SETTINGS
- DEFAULT SCHEMA IDENTIFIER SETTINGS
- FILE SETTINGS

Syntax

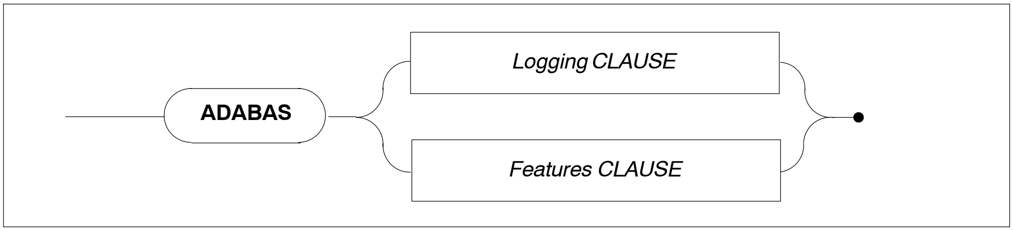

GLOBAL
(
BEGIN
GLOBAL ADABAS SETTINGS
GLOBAL MULTIFETCH SETTINGS
GLOBAL BUFFER MANAGER SETTINGS
GLOBAL CATALOG SETTINGS
GLOBAL ERROR SETTINGS
GLOBAL PREDICT SETTINGS
GLOBAL DEFAULT SCHEMA IDENTIFIER SETTINGS
GLOBAL FILE SETTINGS
)
END

GLOBAL ADABAS Settings

Function

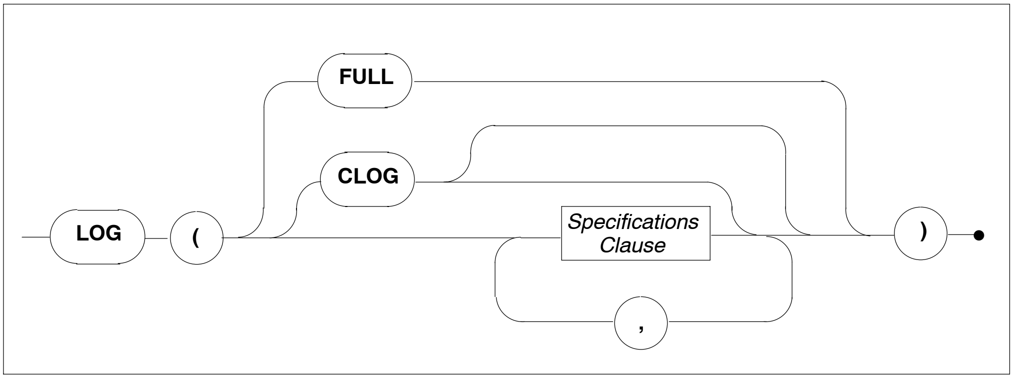
To set ADABAS-specific values that are valid throughout all phases of processing (from precompilation to runtime functions).

Syntax



For information on the individual clauses refer to the description section directly below the relevant syntax diagrams.

Logging Clause



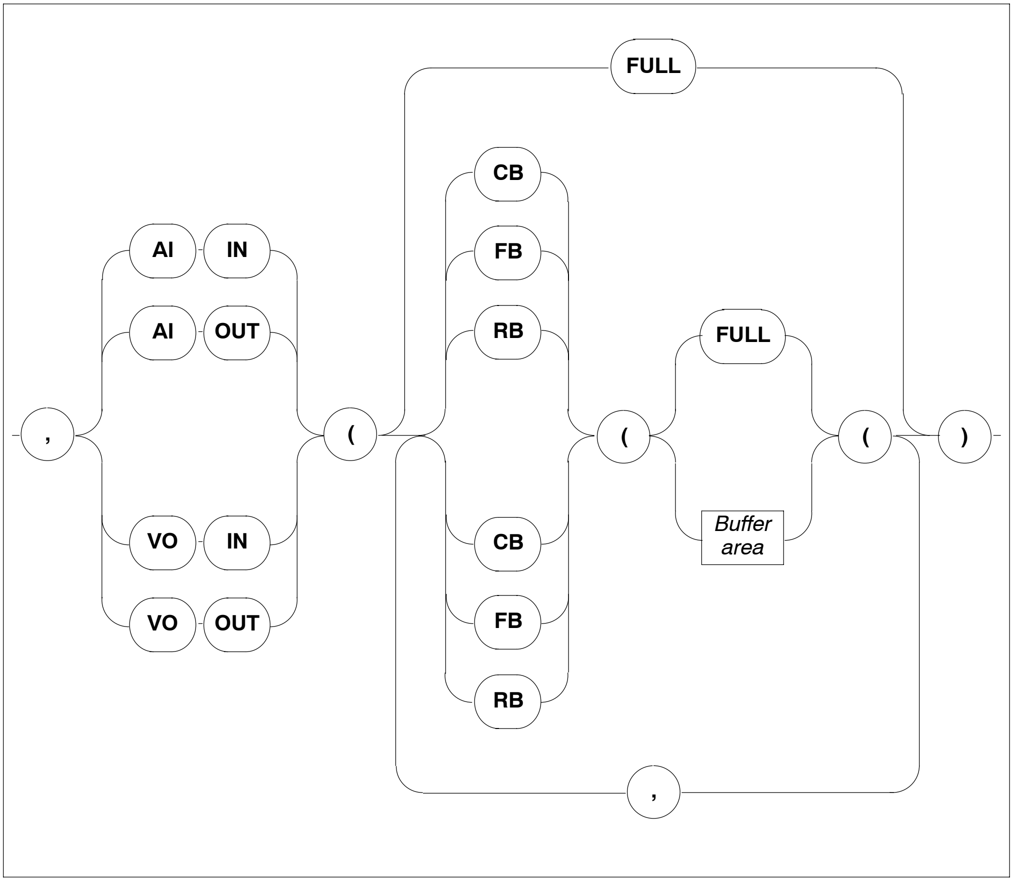
FULL — Full logging of all ADABAS buffers at all log points will be performed, which may be very extensive.

CLOG — If no other details are specified, a compact command logging at the VO−OUT point will be performed. For details regarding the specifications clause see below.

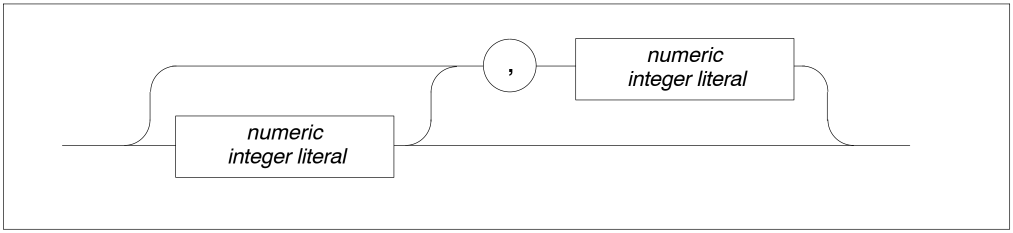
Description

Within the ADABAS logging clause it can be specified at which of the four log points ADABAS calls can be logged. Furthermore, it can be specified which buffer is to be displayed and to what extent. For details on ADABAS Command Logging refer to the chapter **Logging Facilities** earlier in this manual.

Specifications Clause



Buffer Area



AI IN/OUT – VO IN/OUT	Indicates the log point.
CB – SB FULL	Indicates the buffer to be displayed in full.
CB – SB *buffer area*	Indicates the buffer area to be displayed.

Limitations

None

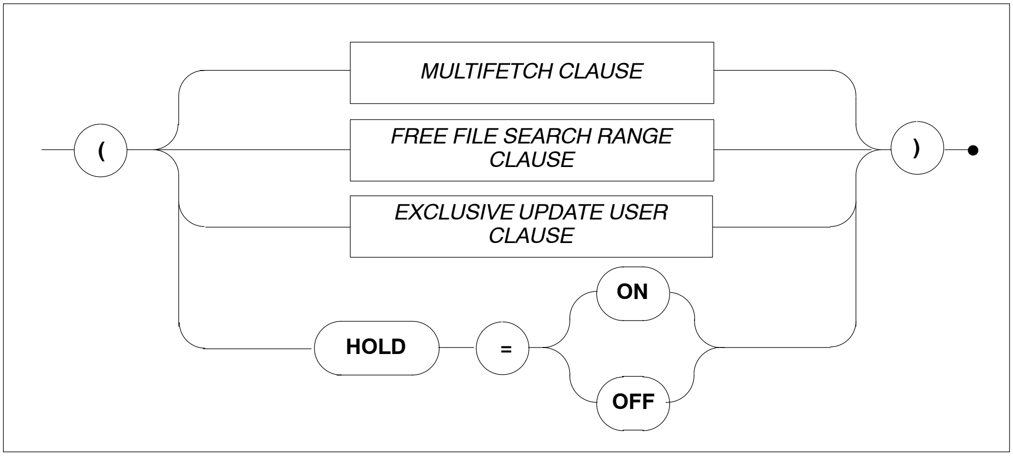
Example

```
GLOBAL BEGIN
ADABAS LOG (AI OUT (CB(10,10),SB(FULL),RB(FULL)))
END

ADABAS LOG (CLOG)
```

Does a compact logging at the “VO OUT” log point.

Features Clause

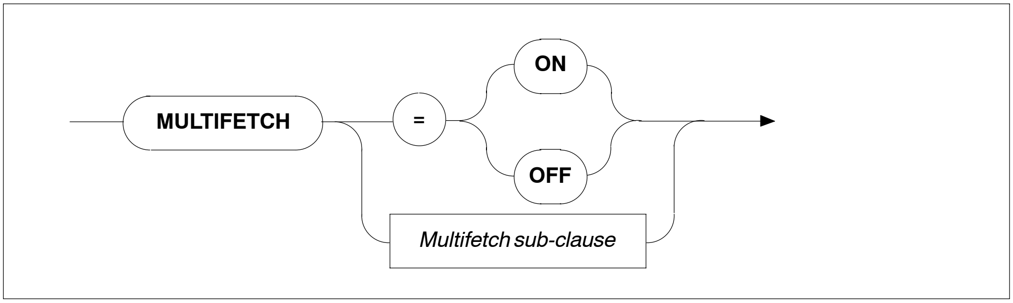
For information on the individual clauses refer to the description section directly below the relevant syntax diagrams.



HOLD = ON — Indicates that if a record is locked by another user, the application will wait until the record becomes available.

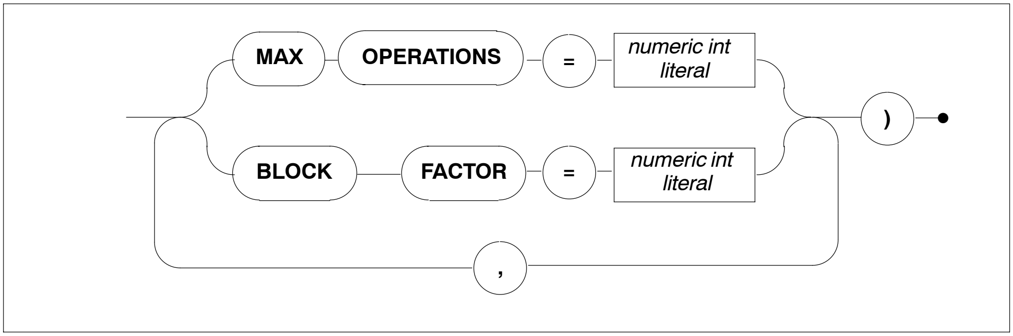
HOLD = OFF — Indicates that if a record is locked by another user, the application will NOT wait, but will get an appropriate response code.

Multifetch Clause



MULTIFETCH ON/OFF — Turns ON/OFF the ADABAS MULTIFETCH feature between ADABAS and the ADABAS SQL Server. If multifetching is not used, severe performance losses can be expected.

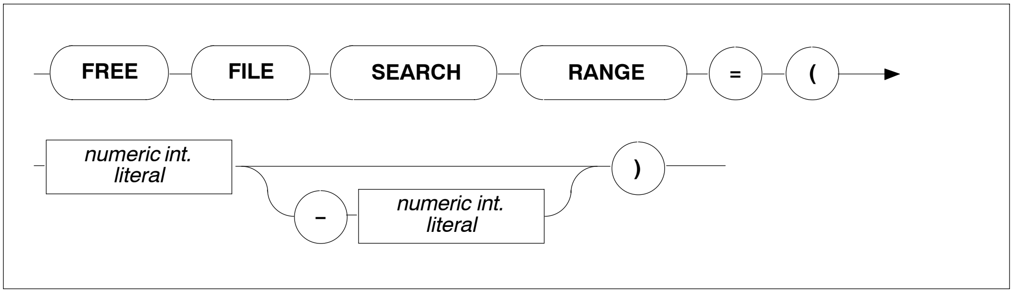
Multifetch Sub-clause



MULTIFETCH MAX OPERATIONS — Defines the number of ADABAS commands which can be used simultaneously with the MULTIFETCH logic.
Minimum value : 4
Maximum value : 32
Default value : 8

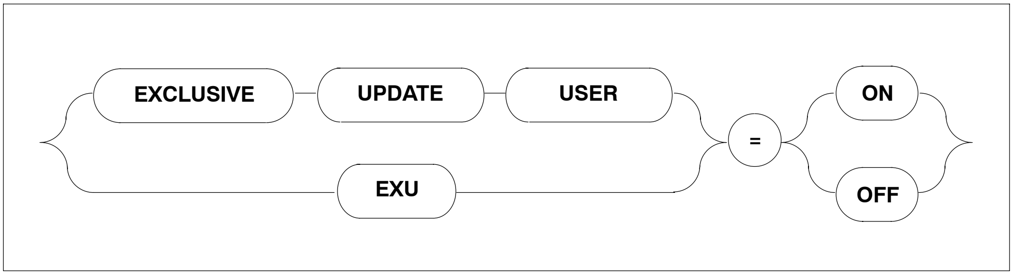
MULTIFETCH BLOCK FACTOR — To optimize the data transfer between the ADABAS SQL Server and ADABAS, the MULTIFETCH logic of ADABAS is used. The BLOCK FACTOR defines how many records should be retrieved with one ADABAS call.
Minimum value: 8
Maximum value: 512
Default value: 16

Free File Search Range Clause



FREE FILE SEARCH RANGE Defines the range of file numbers that will be checked when creating a table or cluster. No files reserved for security should be within the specified range. This setting is used when the file number is not specified in the tablespace for a CREATE TABLE/CREATE CLUSTER statement or when one of the two default tablespaces is used (hardcoded/schema default tablespace). Specifying only the first value will result in a search starting at that number up to the highest file number supported by the particular ADABAS version.

Exclusive Update User Clause



EXCLUSIVE UPDATE USER If set to ON, the user will have exclusive update rights according to the predefined EXU-List. See the original ADABAS manuals or the *ADABAS SQL Server Programmer's Guide*, **The ADABAS SQL Server and other SOFTWARE AG Products**. The default value is OFF.

Limitations

The ADABAS MAXIMUM OPERATIONS setting is only necessary in client/server mode.

Examples

```
GLOBAL BEGIN
ADABAS ( MULTIFETCH ( BLOCK FACTOR = 256 ) )
END

GLOBAL BEGIN
ADABAS ( HOLD = ON )
END
```

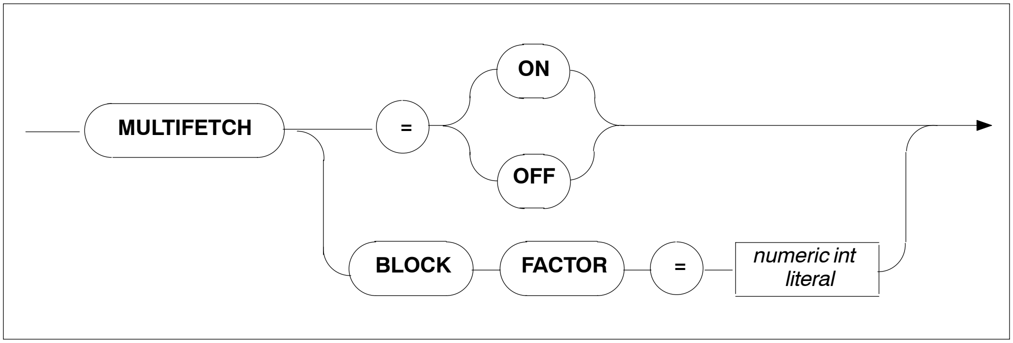
GLOBAL MULTIFETCH Settings

Function

Allows the setting of the following client-specific MULTIFETCH logic:

Syntax

Multifetch Clause



MULTIFETCH ON/OFF — Turns ON/OFF the MULTIFETCH feature of the client.
Default value: ON

MULTIFETCH BLOCK FACTOR — Optimizes the data transfer between the client and the ADABAS SQL Server. The BLOCK FACTOR defines how many rows will be retrieved from the ADABAS SQL Server with one FETCH statement.
Minimum value: 8
Maximum value: 512
Default value: 16

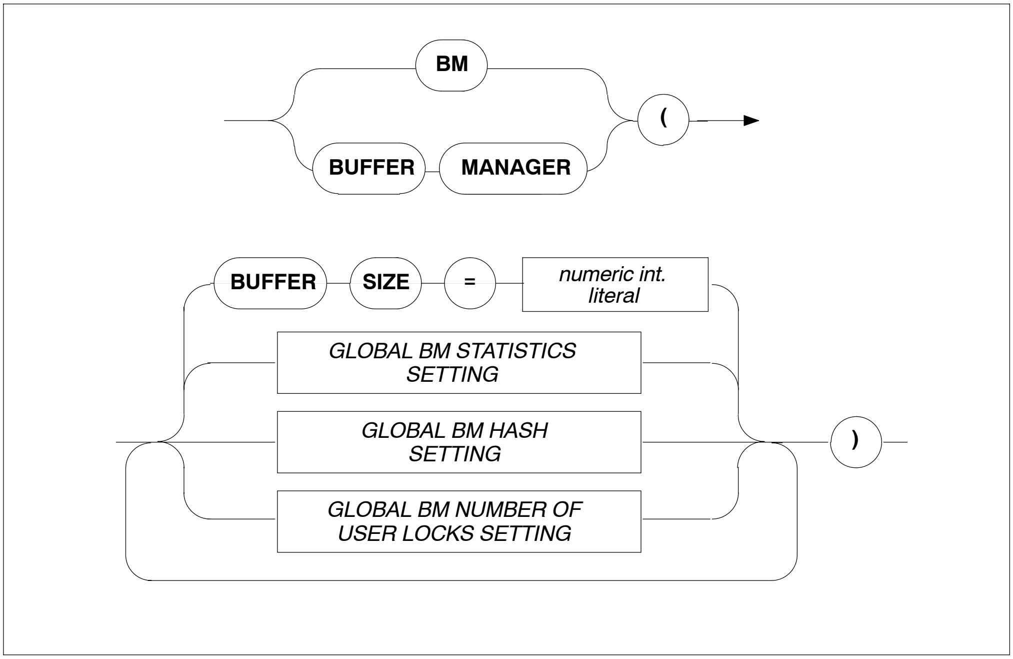
Note:
If a cursor has been defined to be updatable, the use of MULTIFETCH logic will be turned off automatically by the client.

GLOBAL BUFFER MANAGER (BM) Settings

Function

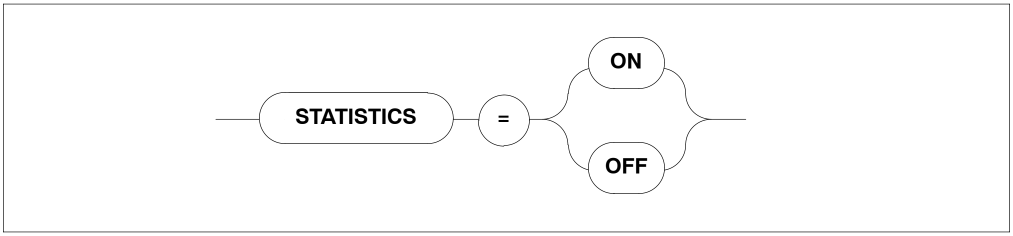
The ADABAS SQL Server catalog buffer stores the objects of the catalog. The buffer is a shared memory in a multi-user environment and a process local memory in a single-user environment.

Syntax



BUFFER SIZE	The size of the catalog buffer in bytes. Default: 262 144 bytes. Minimum: 32 768 bytes (also depends on the length of the hash table), for details refer to the section **GLOBAL BM HASH SETTINGS** below.

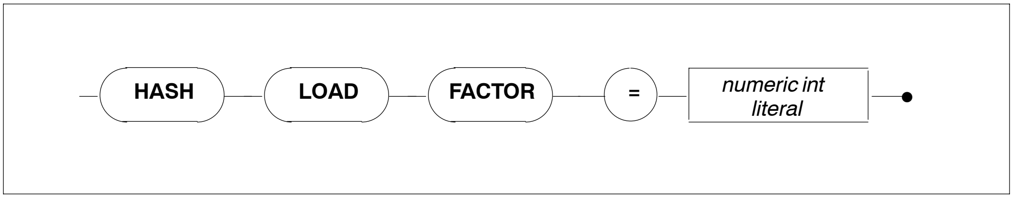
GLOBAL BM STATISTICS SETTING



STATISTICS

Defines whether statistics are collected concerning the operation of the buffer manager (slows the system down). Default: OFF

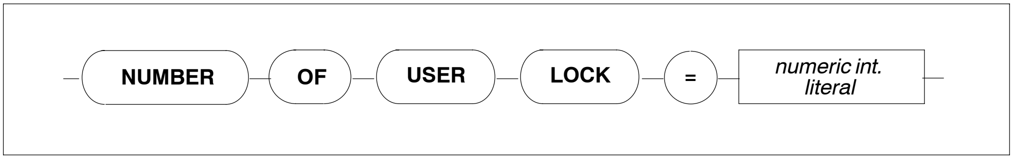
GLOBAL BM HASH SETTING



HASH LOAD FACTOR

Defines the size of the hash table in the catalog buffer. The higher the hash load factor, the faster BM's access to the objects in the buffer.
Default: 20 %
20 % equals 4 KB if the buffer size is smaller than 64 KB
and 8 KB if the buffer size is larger than 64KB.

GLOBAL BM NUMBER OF USER LOCKS SETTING



NUMBER OF USER LOCKS

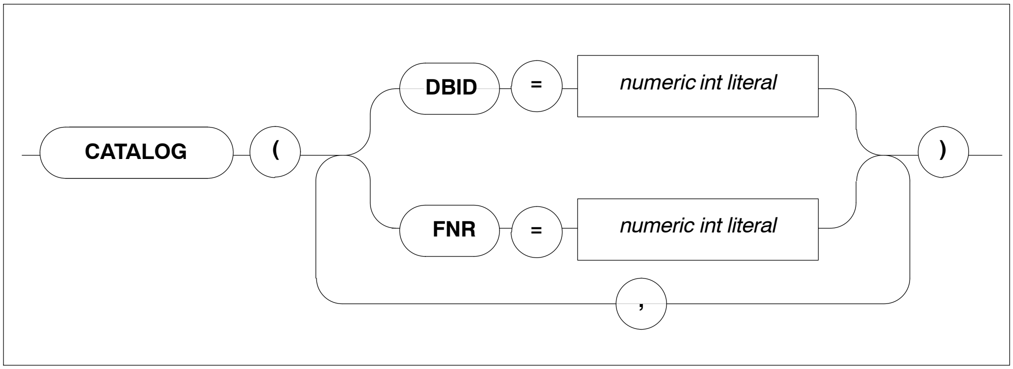
Defines the default number of locks per user if no parameter was specified via BM_R_LOGON(). Default value is 256, minimum is 1 and maximum is 32767.

GLOBAL CATALOG Setting

Function

To set where the catalog can be found.

Syntax



Description

Defines in which ADABAS database and in which ADABAS file the catalog can be found. The catalog contains details about which tables and meta programs are available for the execution of SQL statements.

Default values are DBID = 1 and FNR = 13.
Minimum value is 1, maximum value is 255.

Limitations

For precompiler and runtime files and only necessary in LINKED-IN mode. In client/server mode, this setting is ignored.

The ADABAS database specified by DBID and the file specified by FNR must contain a valid catalog.

If more than one DBID/FNR is specified, only the last specified value is recognized.

Examples

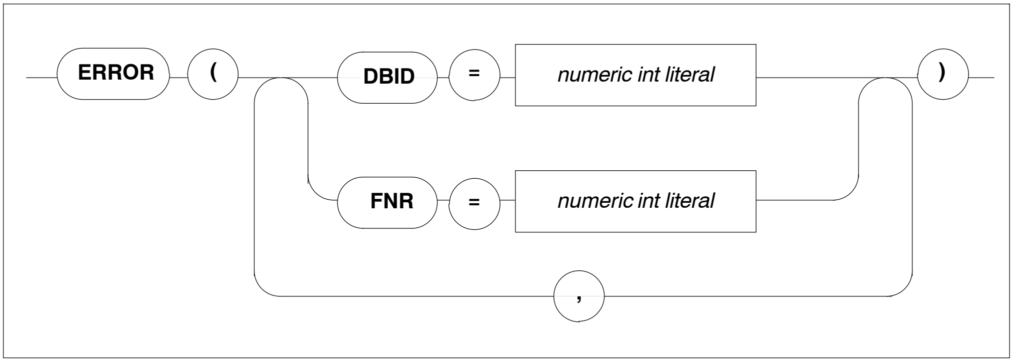
```
GLOBAL
BEGIN
    CATALOG ( DBID = 1, FNR = 13 )
END
```

GLOBAL ERROR Settings

Function

To set where the error messages can be found and to specify the desired language.

Syntax



Description

Defines in which ADABAS database and which ADABAS file the SQL error messages can be found.

Limitations

DBID and FNR are only necessary in client/server mode.

The ADABAS database specified by DBID and the file specified by the file number must contain the valid error messages.

If more than one DBID/FNR is specified, only the last specified value is recognized.

Examples

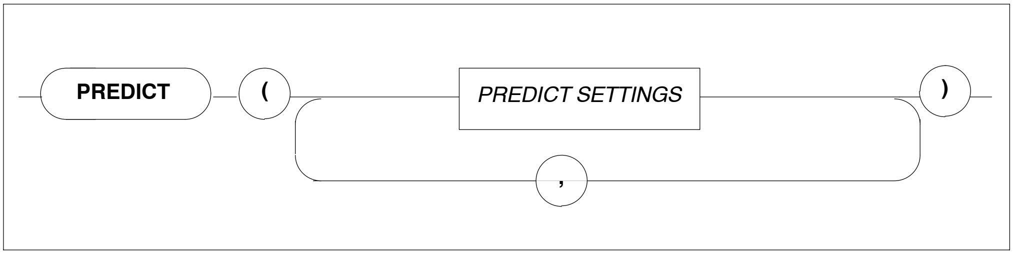
```
GLOBAL
BEGIN
    ERROR ( DBID = 13, FNR = 3 )
END
```

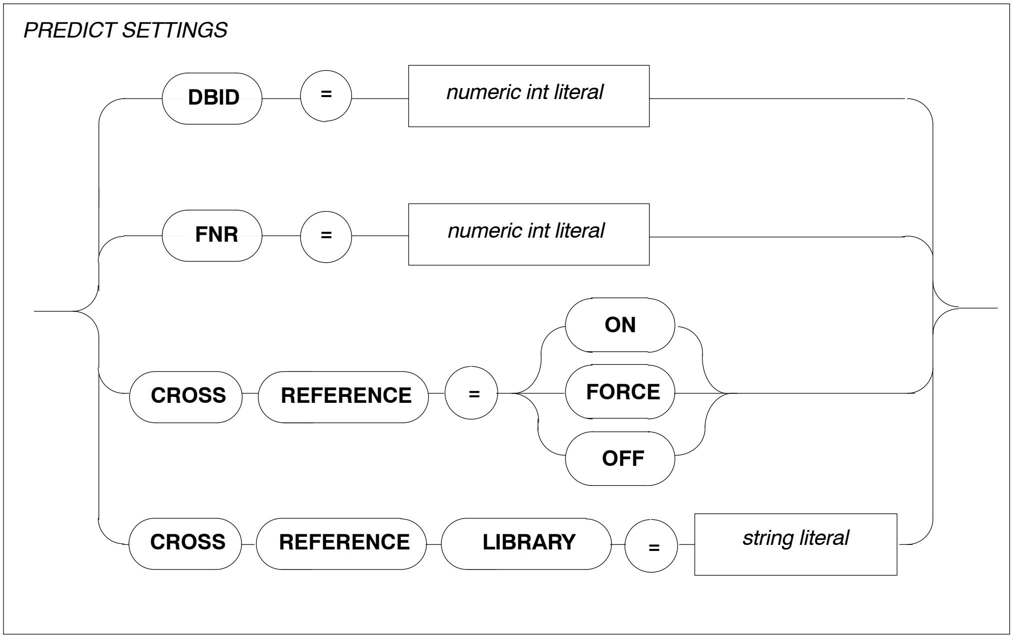
GLOBAL PREDICT Settings

Function

To set PREDICT-specific values which are valid across all parts of the system (from precompilation time to runtime).

Syntax





Description

DBID

Defines in which ADABAS database the PREDICT Cross Reference Library can be found.
Minimum value: 1
Maximum value: 255
Default value: 1

FNR

Defines in which ADABAS file the PREDICT Cross Reference Library can be found.
Minimum value: 1
Maximum value: 255
Default value: 113

CROSS REFERENCE = OFF/ON

Defines whether the active cross reference feature is to be used If set to ON, cross reference data for the host program will be stored in the appropriate PREDICT entries at the end of a precompilation process.

CROSS REFERENCE LIBRARY

Defines the cross reference library name. If a name is entered, this name has to be documented in PREDICT as well. If no name is entered, a default name will be taken.

Limitations

Note:
Interaction with PREDICT is not possible with all ADABAS SQL Server 1.4 versions.

DBID and FNR are only necessary in client/server mode.

The maximum length of a PREDICT Cross Reference Library name is 8 characters.

Examples

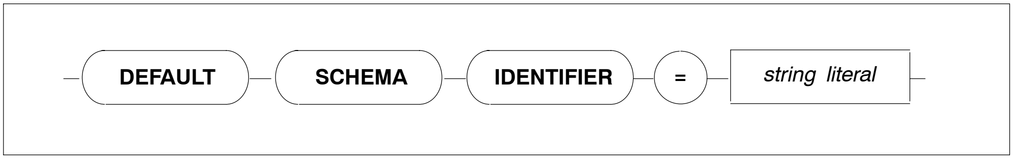
```
GLOBAL
BEGIN
    PREDICT (CROSS REFERENCE = ON)
END
```

GLOBAL DEFAULT SCHEMA IDENTIFIER Setting

Function

Sets the default schema identifier.

Syntax



Description

A table identified by a table identifier only is called an unqualified table specification. The default schema identifier is used to uniquely identify each unqualified table specification occurring in an SQL statement within a compilation unit. This is effective at precompile time for static embedded SQL statements and at runtime for statements processed dynamically via PREPARE or EXECUTE IMMEDIATE statements. The default schema identifier setting has to appear in the precompiler or runtime parameter files, respectively.

If a default schema identifier has not been set explicitly, the unqualified table name will be implicitly qualified by the user identifier set by a CONNECT statement. In the precompiler environment, this user identifier is derived from the operating system user name.

The string containing the default schema identifier is not case-sensitive and must be defined according to the rules of SQL identifiers.

Limitations

The length is limited to 32 characters.

Example

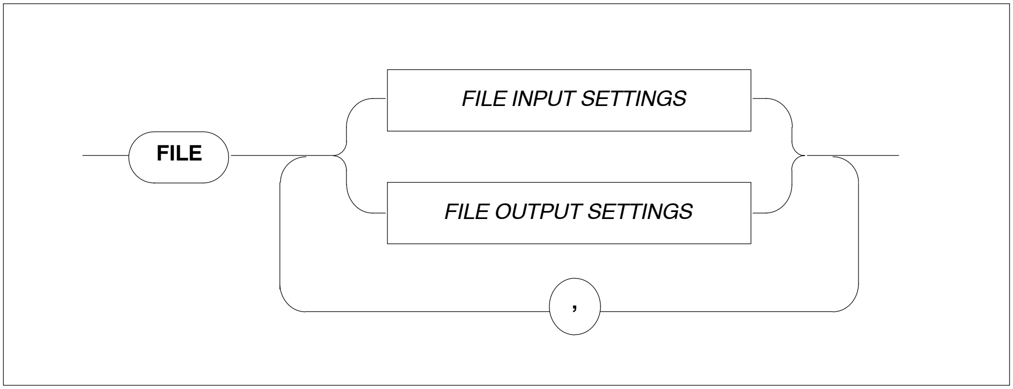
```
GLOBAL
BEGIN
     DEFAULT SCHEMA IDENTIFIER = “ESQ”
END

GLOBAL
BEGIN
     DEFAULT SCHEMA IDENTIFIER = “robert”
END
```

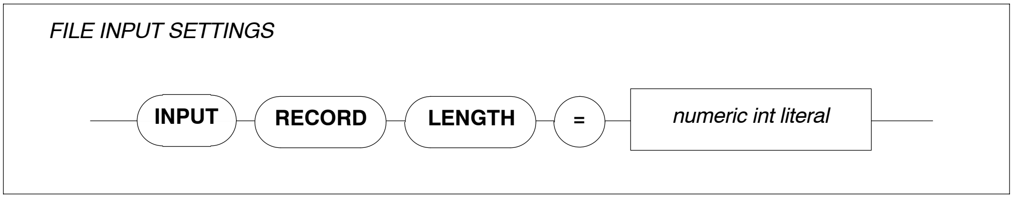
GLOBAL FILE Settings

Function

Sets the input/output record length.

Syntax





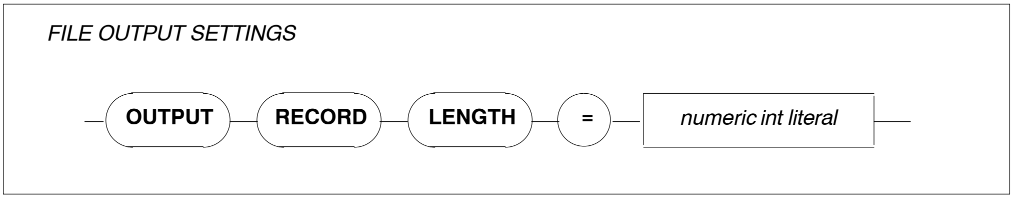
INPUT RECORD LENGTH Maximum record length (integer).

The input record length is used by PPL and the precompiler. The default value is 72 characters.

Example

Sets the record length to 132 characters.

```
FILE INPUT RECORD LENGTH = 132
```



OUTPUT RECORD LENGTH   Maximum record length (integer).

The output record length is used by the precompiler. Default values: 256 characters for non-mainframe environments and 80 characters for mainframe environments.

Note:
On mainframe platforms, the record length of the output file can be specified beforehand and is not overwritten by the specification of the GLOBAL FILE parameter setting. This means that truncation occurs if the predefined output record length is shorter than the input record length.

Example

Sets the record length to 72 characters.

```
FILE OUTPUT RECORD LENGTH = 72
```

SERVER Settings

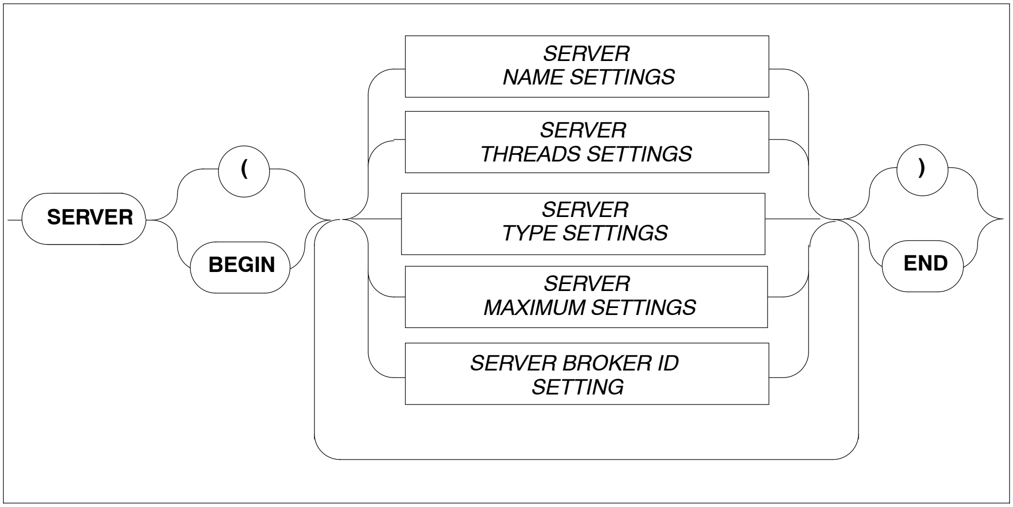
Function

The SERVER settings define the server-specific environment and are needed when running in client/server mode only. They are used during start-up of a server.

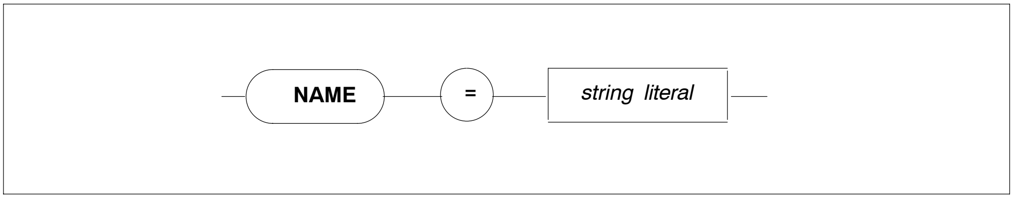
The types of settings for the SERVER section are:

- SERVER NAME SETTINGS
- SERVER THREAD SETTINGS
- SERVER TYPE SETTINGS
- SERVER MAXIMUM SETTINGS
- SERVER BROKER ID SETTING

Syntax

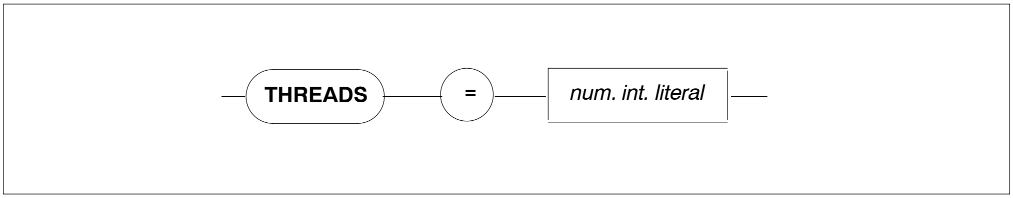


SERVER NAME Setting



Specifies the the name of the server, which must not be longer than 8 characters. Note that this parameter is not used on IBM platforms.

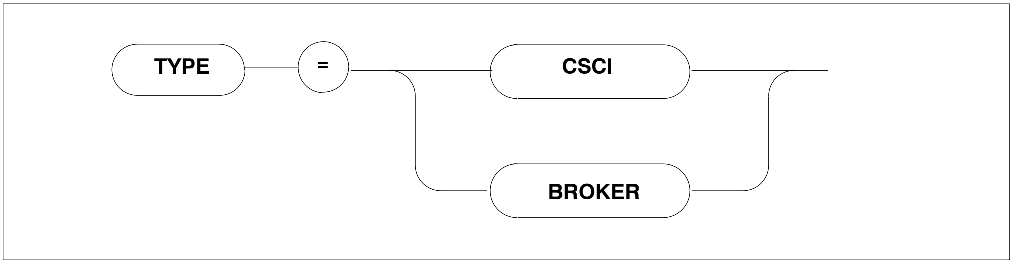
SERVER THREADS Setting



Specifies the number of threads per server. More threads mean more resources required.

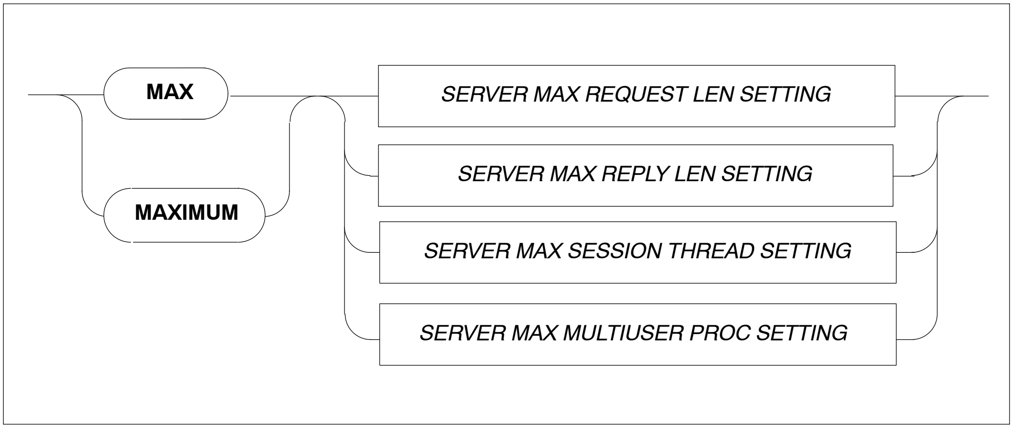
Default value: 5, minimum value: 0.

SERVER TYPE Setting



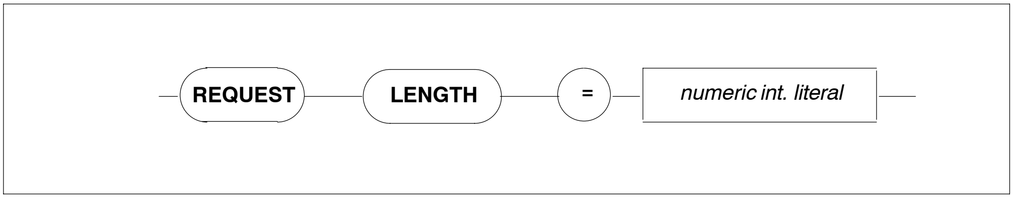
Specifies the type of client-server communications mechanism. The default communication mechanism is CSCI. Must be set to BROKER if the SERVER BROKER ID Setting is to be specified.

SERVER MAXIMUM Settings


MAX
MAXIMUM
SERVER MAX REQUEST LEN SETTING
SERVER MAX REPLY LEN SETTING
SERVER MAX SESSION THREAD SETTING
SERVER MAX MULTIUSER PROC SETTING

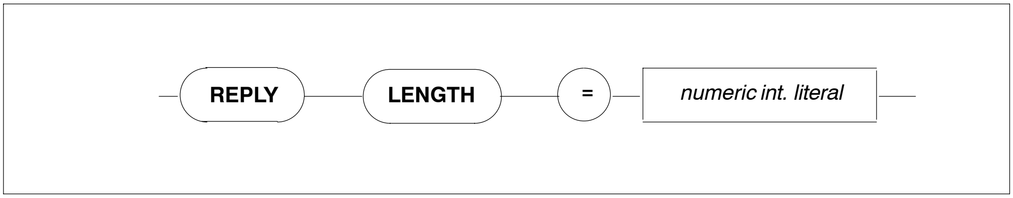
Sets the maximum values for the server.

SERVER MAXIMUM REQUEST LENGTH Setting



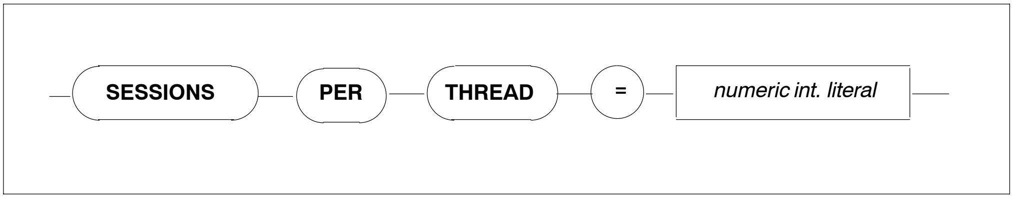
Specifies the maximum request length for the client-server communications. The default value is 8000 bytes, maximum length is 64000 bytes. Recommended minimum value for UNIX only: 32000 bytes.

SERVER MAXIMUM REPLY LENGTH Setting



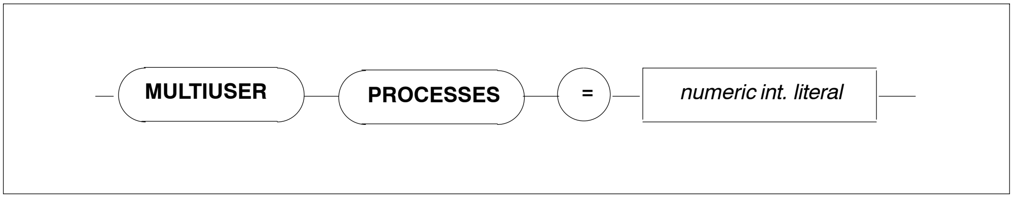
Specifies the reply length for the client/server communications. The default value is 8000 bytes, maximum length is 64000 bytes.

SERVER MAXIMUM SESSION THREAD Setting



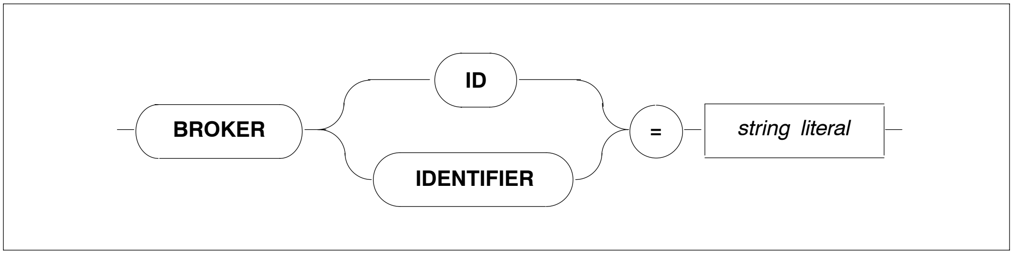
Specifies the maximum number of sessions per thread; each user process is one session. Zero means infinite (∞). For Version 1.4 the maximum number of sessions per thread must be 1.

SERVER MAXIMUM MULTIUSER PROCESSES Setting



Specifies the number of additional processes which may use the server e.g., precompiler. The default value is 64.

SERVER BROKER ID Settings



BROKER ID — Defines the name under which the communication between the ADABAS SQL Server and SOFTWARE AG's middleware communication protocol BROKER will take place. This setting will only take effect when the SERVER TYPE Setting is defined as BROKER.

INDEX

Notes

Notes