Adabas SQL Server

Installation and Operations Manual (VSE)

Manual Order Number: ESQ143-010VSE

This document applies to Adabas SQL Server Version 1.4.3 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 or to the following e-mail address:

Documentation@softwareag.com

TABLE OF CONTENTS

PREFACE

Using This Manual – Some Basic Information

This manual describes the operating system requirements and procedures for installing and operating Adabas SQL Server, and the Adabas SQL Utilities on the VSE platform.

Who Should Read This Manual

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

How This Manual Is Organized

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

Preface	contains a brief overview of the contents of this manual.
Chapter 1	gives an overview of the system architecture and configuration as well as the concepts of the security system. Furthermore, special attention is drawn to those issues which needs to be considered and/or decided before the actual installation procedure is performed.
Chapter 2	contains description of the Adabas SQL Server Interfaces such as Entire Service Manager (ESG), the VO Parameter Module, the Tuple Manager, the External Security Interface and the User Exits.
Chapter 3	describes the full installation process for Adabas SQL Server and the Adabas SQL Server Utilities and Interfaces, as well as all necessary verification steps.
Chapter 4	contains the information necessary to operate Adabas SQL Server under VSE.
Chapter 5	contains a detailed description of how to work with the Adabas SQL Server Utilities.
Chapter 6	contains a description of the Logging Facility with the various types of logging and the Trace Facility.

Appendix A is valid for all supported platforms (Mainframes and UNIX/OpenVMS). It contains a detailed explanation of the parameter processing language (PPL) which enables the configuration of the various components of Adabas SQL Server.

Appendix B contains lists of helpful information like replacement variable symbols, VO subcodes, CICS response codes and describes the installation aid utility ESQAID.

Other Manuals You May Need

The following Software AG manuals are referred to in this manual; they may be needed when installing/operating Adabas SQL Server:

- Adabas SQL Server Reference Manual (ESQ142-030ALL);
- Adabas SQL Server Programmer's Guide (ESQ142-020ALL);
- Adabas SQL Error Messages (ESQ142-060ALL);
- Entire Service Manager Installation Manual (ESG113-010IBW);
- Entire Service Manager Administration Manual (ESG113-080IBW);
- Entire Service Manager Messages and Codes Manual (ESG113-060IBW);
- Entire Net-Work Installation and Operations Manual (WCP554–010MNF)
- Adabas 5 Implementation and Maintenance Manual (ADA-533-010);
- Adabas 5 DBA Reference (ADA-533-030);
- Adabas 5 Messages and Codes (ADA-533-060);
- Adabas 5 Utilities (ADA-533-808 – two volumes);
- Adabas 5 Operations (ADA-533-110);
- ANSI/ISO Standards SQL (X3.135-1989, ISO 9075).

1

SYSTEM OVERVIEW

This chapter briefly describes the various components of Adabas SQL Server, their purpose and interaction with each other as as well as with other products. Furthermore, major terms of the security features and important points to be considered **before** installing the product are discussed.

Adabas SQL Server provides a standard SQL interface for the Adabas C database management system. It enables SQL access to existing Adabas C data structures and the implementation of SQL-based applications.

Adabas SQL Server provides different means with which data can be accessed and manipulated. Adabas SQL Server supports both static and dynamic SQL statements which may be:

- embedded in a 3rd generation host language application program
- embedded in a Natural application program
- submitted using an interactive interface,
- submitted using the Adabas / ODBC Interface.

Currently, Adabas SQL Server supports SQL statements embedded in either C, COBOL or PL/I. Due to the modular design of Adabas SQL Server, the functionality is identical regardless of the host language chosen.

The remainder of this section provides the reader with an overview of the system architecture and configuration under VSE.

A description of Adabas SQL Server functionality is provided in the *Adabas SQL Server Programmers Guide*, chapter **Introduction to Adabas SQL Server**. This section provides, for example:

- an explanation of how Adabas SQL Server functions,
- a list of supported SQL dialects, and
- an overview of the phases involved in developing an SQL-based application program.

System Architecture

This chapter is valid for IBM mainframe platforms only and provides an overview of the Adabas SQL Server components and their interaction and a very brief description of their functions.

A server consists of the following:

- the catalog files,
- a message file,
- an Adabas SQL Server runtime system, and
- a server name

The server name is used to identify the server.

Precompiler and utility programs are also provided to assist the application development.

System Files

Adabas SQL Server has four system files, which are loaded into an Adabas database during the installation.

The Adabas SQL Server catalog consists of three Adabas files and contains information about the data structures (tables, views, etc.) known to the SQL environment. The message texts used by Adabas SQL Server and its utilities are stored in fourth file.

The catalog files must reside in a single Adabas database. The objects (Adabas files) described in the catalog, which are accessed by Adabas SQL Server, may reside in any accessible Adabas database.

Refer to the chapter **Adabas SQL Server Data Structures** of the *Adabas SQL Server Programmers Guide* for further information.

Runtime System

Each server consists of a catalog file and a runtime system.

The Adabas SQL Server runtime system can be executed in one of two distinct modes:

- LINKED-IN single-user mode, or
- client/server multi-user mode.

LINKED-IN Single-User Mode

If the SQL request is executed in the process context of the client application, it is called LINKED-IN mode. The SQL request is done by the Adabas SQL Server runtime system (ESQLNKBT).

An application, which does not share resources with the other clients, is running in LINKED-IN single-user mode.

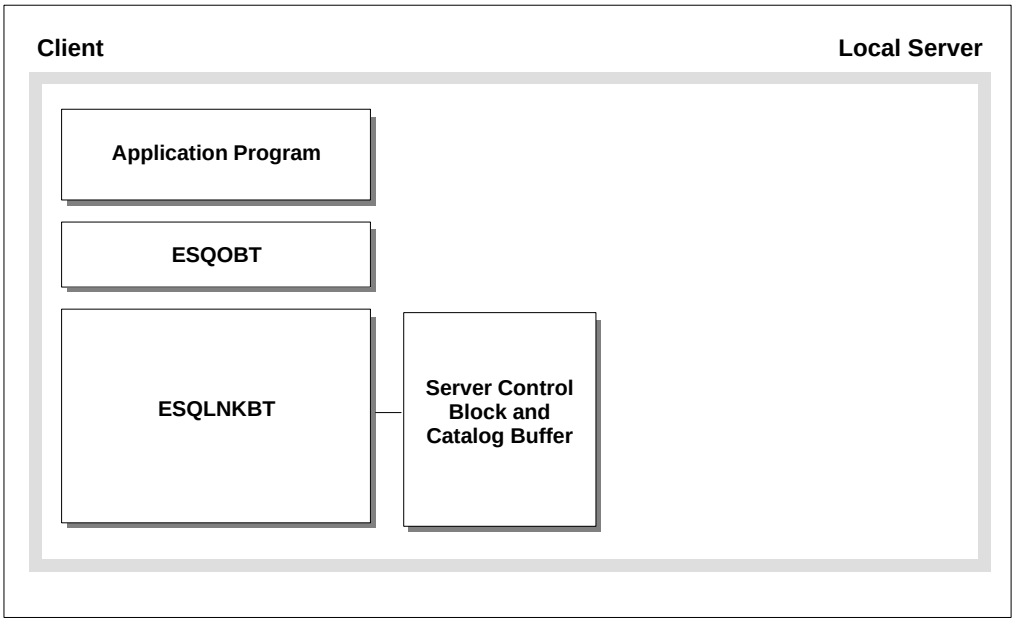


Figure 1-1: LINKED-IN Single-User Mode

During installation the catalog definitions are loaded in LINKED-IN single-user mode using the Adabas SQL Server BASIC Interactive SQL as client application.

Note
A multi-user LINKED-IN mode is not available under VSE.

Client/Server Multi-User Mode

If the SQL request execution is performed by a server process and not by the application program's process, then this is called client/server multi-user mode. The SQL request is shipped by the client application to the server for processing. The Adabas SQL link module (ESQLNKBT) is loaded at runtime and is responsible for packing and unpacking of the SQL requests as well as the client/server communication. The SQL request execution is performed by ESQSRV on the server side.

The application program does not have access to any server shared memories.

When the client and server are on one and the same node, we talk about *local* client/server multi-user mode. If the server is located on a remote node, it is referred to as *remote* client/server multi-user mode.

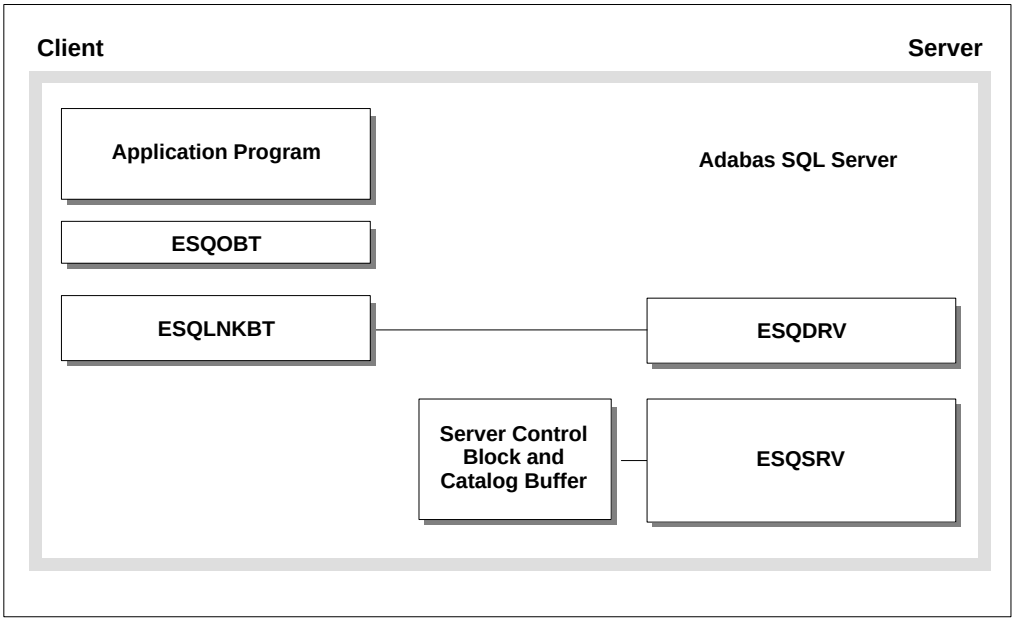


Figure 1-2: Local Client/Server Multi-User Mode

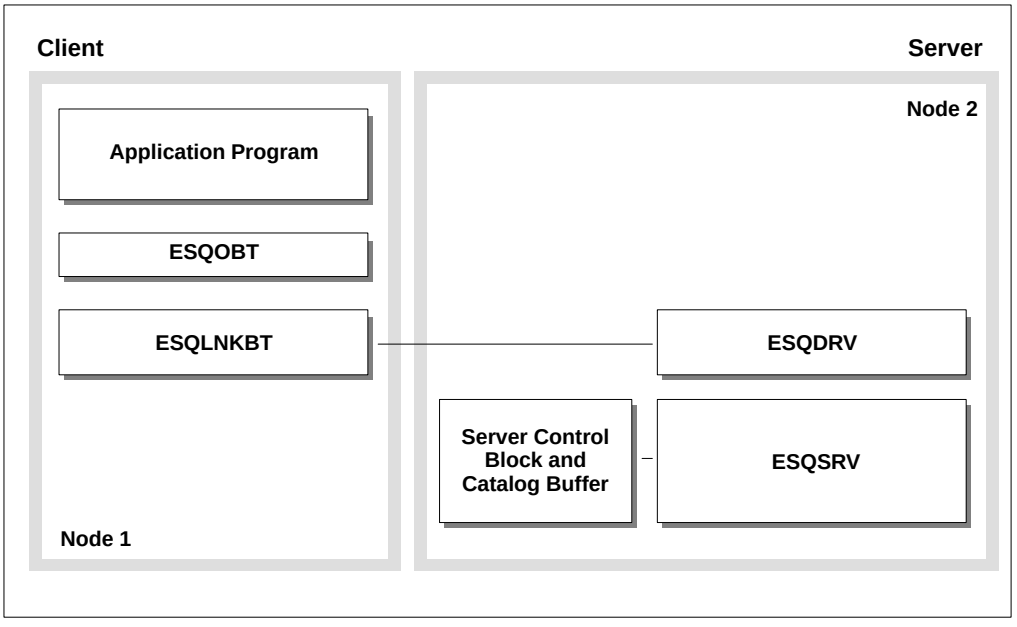


Figure 1-3: Remote Client/Server Multi-user Mode

Communication Protocol

The communication protocol used by Adabas SQL Server for the interprocess communication between client and server is the Client Server Communication Interface (CSCI).

CSCI is the programming interface of a transport service in a client/server environment.

CSCI uses the Adabas SVC to transport requests in local client/server multi-user mode and requires Entire NET-WORK, when operating in remote client/server multi-user mode.

For details about Entire Net-Work, refer to the Software AG documentation. For more information about CSCI, refer to the *Adabas SQL Server Programmers Guide*, **Appendix A**, section: **Adabas SQL Server and Entire CSCI**.

Client Environments

Adabas SQL Server supports the execution of VSE client applications in the following environments:

- Batch
- Com-plete
- CICS

Adabas SQL Server enables ODBC-applications access to the mainframe Adabas C database. The ODBC-applications communicate with the server using Entire Net-Work and the product Adabas/ODBC.

Note
A VSE server also supports Adabas SQL clients executing on other platforms, for example under: UNIX, OpenVMS, Windows NT.

Precompiler

As described in the introductory chapter to the *Adabas SQL Programmers Guide*, application programs containing embedded SQL must be precompiled, prior to their compilation and execution.

Currently, the following 3rd generation host languages are supported by Adabas SQL Server. A separate precompiler is provided for each source language:

Host Language	Precompiler
COBOL I & II	ESQPCCOB
C-Language	ESQPCC
PL/I	ESQPCPLI

Adabas SQL Server provides the Include Facility (ESQPCINC) to expand EXEC SQL INCLUDE statements in COBOL programs, prior to precompilation. This special utility automatically executes the COBOL precompiler after including the copybooks/copycode in the source program.

Utility Programs

The following Adabas SQL Server utilities are available on the mainframe platform:

Utility Name	Module Name
BASIC Interactive Facilities	ESQBIF
ADVANCED Interactive Facilities	NATURAL/SYSESQ
Generate Table Description Utility	ESQGTD
Migration Utility	ESQMIG

The following is a brief description of each utility. A detailed description is provided in chapter: **Operating the Adabas SQL Server Utilities,** later in this manual.

Basic Interactive Facilities (BASIC)

BASIC Interactive Facilities is a line-oriented batch application which executes dynamic SQL statements. It is typically used for establishing data structures or performing isolated SQL statements which do not require further application processing.

BASIC is used to verify the installation by defining, loading and querying the SAGTOURS database.

Advanced Interactive Facilities (ADVANCED)

ADVANCED Interactive Facilities is a Natural application which executes dynamic SQL statements. This application provides the user with a comfortable interactive interface.

Generate Table Description Utility

This utility generates draft CREATE TABLE/CLUSTER DESCRIPTION statements for existing Adabas files, which are to be introduced to Adabas SQL Server.

The utility output can be edited as needed and submitted using BASIC.

Migration Utility

This utility serves as a migration tool designed for the upgrade from Adabas SQL Server Version 1.3 to Adabas SQL Server Version 1.4.

Diagnostic Facilities

Adabas SQL Server provides the following diagnostic facilities.

Adabas SQL Server Logging Facilities

They provide information, which can be used for performance analysis, problem analysis and access monitoring. The following types of logging are available: SQL commands, Adabas commands, Client / Server Communication and Elapsed Time.

The Logging Facilities are described in a separate chapter later in this manual.

Adabas SQL Server Trace Facility

This is only available in the TRACE version of Adabas SQL Server and its components. This version is delivered in a separate sublibrary, and should only be used to diagnose software problems under the guidance of Software AG support.

System Interfaces

The system interface routines provide the Adabas SQL Server runtime system with environment-dependent system services and parameters.

Entire Service Manager

Entire Service Manager (ESG) is used to implement the client/server mode of Adabas SQL Server. It provides the server with multi-user threading capabilities and a CSCI server interface.

VOPRM - Parameter Module

The VO parameter module (VOPRM) contains all Adabas SQL environment dependent parameters. This module contains: the mainframe implementation of environment variables, a concept known to open systems platforms, the Adabas SQL Server routing file and a DB2 code translation table. The module VOPRM is required and is dynamically loaded during the initialization of the Adabas SQL environment.

Tuple Manager

The Tuple Manager is a component of the Adabas SQL Server runtime system. It expands the capabilities of the Adabas nucleus by providing temporary storage, sort and retrieval functions for intermediate result sets. The Tuple manager requires a temporary workfile manager and uses the system sort routines.

Adabas Link Module

The Adabas SQL Server runtime system communicates with the Adabas nucleus using the standard Adabas link module. The Adabas SQL Server utilities use the module ADAUSER.

Adabas Utilities

The Adabas SQL Server runtime system requires Adabas Online Services (AOSASM) during the processing of data definition language (DDL) statements.

Additionally, the runtime system requires the Adabas Invert Utility (ADAINV) during the processing of the CREATE INDEX statement(s).

External Security

The Adabas SQL Server runtime system provides an interface to third party external security products and interacts with Adabas Security products.

A detailed explanation of the topics above is provided in the following chapters of this manual.

System Configuration

Parameters Defining the Server Environment

The following parameters define the server environment:

- Entire Service Manager Parameters (SYSPARM):
  They are processed during the initialization and uses the server name and ACCESS-ID to register/identify the server. Other parameters define the number and size of the client threads, provide limitations for the access and CPU times (timeout parameter).
- VO-Parameter Module (VOPRM):
  It contains the environment variables, routing file/table and the DB2 code translation table. This module is required in both the client and the server environments under VSE.
- Adabas SQL Server Parameters (ESQPARM)
  They are required in the server environment and may optionally be provided in the client environment. These parameters limit the size of the request/reply (transport) buffers and provide a means of setting the request/reply timeouts. Each of these parameters are described in **Appendix A** to this manual.

Parameters influencing the Client Environment

The following parameters influence the Adabas SQL Server Client environment:

- VO-Parameter Module (required)
- Adabas SQL Server Parameter Processing Language/PPL (optional)

Parameters of Other Products which Influence Adabas SQL Server

An SQL application is directly influenced by the parameters of other products. These parameters most often restrict the size of the communication/transport buffer(s) or set a timeout limitation:

- Adabas C Parameters
- Net-Work Parameters
- Adabas / ODBC Parameters

Adabas C Parameters

Additionally, the Adabas C parameters and file sizes restrict the size of database transactions, such limiting factors are the hold queue parameter(s) and the Adabas WORK file size. These parameters should be taken into consideration during the transaction logic design and when processing tables with multiple columns defined as longalpha.

JCL Skeleton and Parameters

Furthermore, the behavior of Adabas SQL Server is influenced by the following JCL skeletons and parameters:

- Job-, Procedure- and Parameter-Skeletons:
  These skeletons were created during the installation process and are to be maintained as needed.
- Job Parameters:
  Some JCL-Parameters directly influence the processing of a job. The SIZE value limits the amount of space available to the job. This can be a limiting factor, when precompiling large application programs or executing in LINKED-IN mode. Time- and Line/Output-Limitations may also hinder the successful completion of an SQL application, particularly when the Adabas SQL diagnostic utilities are in use.

Influences on Natural Applications

Factors which influence the Natural applications are:

- Natural Parameter Module (NATPARM)
- Natural for Adabas SQL Parameters
- Predict Definitions

Security Considerations

This section should provide the reader with an overview of the security features available with Adabas SQL Server.

The *Adabas SQL Server Programmer's Guide* provides the reader with an introduction to the server security concepts. The topics discussed there are platform-independent and thus apply also to other platforms.

The mainframe aspects of the server security topics are briefly discussed in this section. A detailed explanation is provided in the chapter **Adabas SQL Server Interfaces**.

During the installation of Adabas SQL Server, it will be necessary to decide whether the security features are to be activated or not (security server *versus* non-security server). Once made, this decision cannot be reversed without re-installing the server. Therefore, it is of importance, that the security documentation be thoroughly read and understood.

Terminology

The security features provided by Adabas SQL Server are compatible with the ANSI standard, and are discussed in detail in the *Adabas SQL Server Programmer's Guide.* The following terms have been extracted and are briefly explained:

User

In SQL, the concept of user is captured by an *authorization identifier.* Every operation in the SQL system is validated according to the authorization identifier that attempts to invoke the operation. Authorization identifiers in Adabas SQL Server have been implemented as users, which are created using the CREATE USER statement. Users register with Adabas SQL Server using the CONNECT statement.

Privileges

SQL requires an appropriate privilege for every sort of access operation that can be performed on the database. Privileges are stored in the catalog and can be assigned for the following access operations: SELECT, INSERT, DELETE and UPDATE. Privileges are maintained using the GRANT and REVOKE statements.

Authentication

The user identification and password, which are passed using the CONNECT statement, are used to verify the identity of the user.

Authorization

The access to database objects is restricted by privileges, which a user has been granted. Prior to the execution of each SQL statement, the user's privileges are compared with the access operation to be performed. The access operation can be authorized or rejected.

Security Server

This is an Adabas SQL Server with the security features activated. Thus providing SQL2 security functionality using the GRANT and REVOKE statements. Such a server performs authentication and authorization checks.

Non-Security Server

This is an Adabas SQL Server with the security features de-activated. A non-security server provides authentication checks only.

External Security Products

Adabas SQL Server provides an External Authentication Interface to third party security packages using the SAF interface, such as: CA-ACF2(TM), CA-TOP SECRET(TM) and RACF(TM). This interface enables the implementation of a central authentication check and password control.

Currently, authorization checking cannot be delegated to external security packages.

Adabas Security Products

Adabas SQL Server utilizes the standard Adabas interface and thus is subject to the same consideration as any other Adabas application working together with Adabas security facilities. The following Adabas security products can be used to prevent unauthorized access to and/or updating of Adabas database files:

- Adabas Data Encryption (Ciphering)
- Adabas Security and the related utility ADASCR

Further Information

This section only provided a brief introduction to security and Adabas SQL Server. Refer to the chapter **Adabas SQL Server Interface**, section **External Security Interface** of this manual, where the following topics are discussed:

- User Administration
- Usage of External Security Products
- Usage of Adabas Security

Further information may also be found in the appropriate product manuals.

- Adabas Security Manual
- Security Product Documentation (third party)

Installation Considerations

The following points are to be considered **PRIOR** to installing Adabas SQL Server:

- Upgrading From A Previous Version
- Installing a Security / Non-Security Server
- Using External Security Products
- Using Adabas Security Products
- Administering User Identifications and Passwords

Upgrading From A Previous Version

The Migration Utility is provided to assist in the conversion of the Adabas SQL Server Version 1.3 directory file to the Adabas SQL Version 1.4 catalog.

The Migration Utility requires that a complete installation of Adabas SQL Server Version 1.4 be performed, before the upgrade can be attempted.

The Migration Utility and a description of its usage is provided in chapter **Operating the Adabas SQL Server Utilities** later in this manual.

Installing a Security / Non-Security Server

The decision to install a security server must be made **prior** to initializing the Adabas SQL catalog.

During Installation Step **A.6** the reader must decide on the type of server to be installed. This is done by replacing the variable $ESQSEC with one of the following values:

ON	Security Server
OFF	Non-Security Server

Warning:
Once the server has been installed as a security server, it can only be changed into a non-security server by completely reinstalling it. The same is true the other way around.

Using External Security Products

It is recommended that Adabas SQL Server be installed – initially – without the added complexity of external security products. The security interfaces can be activated after the installation has been verified.

Using Adabas Security Products

Adabas SQL Server utilizes the standard Adabas interface and thus is subject to the same considerations as any other Adabas application working together with Adabas security products.

Please refer to the *Adabas Security Manua*l for further information.

Administering User IDs and Passwords

Before activating the external security products, the reader should give some thought to the administration of user identification and passwords.

As described in the *Adabas SQL Server Programmer's Guide*, user identifiers (userID) are unique within the Adabas SQL Server catalog and must be defined by the user DBA. The user DBA is the only person authorized to execute the CREATE USER statement. User identifiers may optionally be assigned a password.

Prior to activating the external security product the following points must be considered:

– **User Identifier Maintenance**:
  A unique user identifier must be defined for each user that is to connect to Adabas SQL Server. This is the value passed to Adabas SQL Server using the CONNECT-Statement and which in turn is passed to the external authentication interface for checking.
– **Password Maintenance**:
  It is recommended, that the user identifiers be defined in the Adabas SQL Server catalog WITHOUT a password, thus delegating the password maintenance to the external security system.

Above points apply to all user identifiers with the exception of the administration user identifier DBA. Authentication and Authorization checking for the user identifier DBA is performed solely by Adabas SQL Server. External security products are **not** called for this task.

Therefore, upon completion of the installation and verification of Adabas SQL Server, it is recommended that the user identifier DBA be assigned a password. This can be performed using the BASIC Interactive Facility to execute an ALTER USER statement with the SET PASSWORD clause. For example:

```
ALTER USER DBA SET PASSWORD 'password_identifier' ;
```

Note
Although Adabas SQL Server supports password identifiers of up to 32 characters, the use of external security products limits the length of the password identifier to a maximum of 8 characters.

The verification jobs / programs use the following user identifiers, neither is assigned a password:

- DBA to create sample schema and user/owner,
- ESQ to create sample application SAGTOURS.

Prior to verifying the external authentication interface the following points must be considered:

- Define the user identifier ESQ in the external security system,
- Modify the verification jobs to reflect any password changes (user identifier: DBA)

Note
It is not recommended to assign a password to the user identifier ESQ. Doing so, will cause the verification programs to fail with an SQL code indicating that a security violation (password exception) has occurred.

2

ADABAS SQL SERVER INTERFACES

This chapter contains the description of the Adabas SQL Server Interfaces such as:

- Entire Service Manager (ESG),
- the VO Parameter Module,
- the Tuple Manager,
- the External Security Interface,
- the User Exits.

Entire Service Manager

Entire Service Manager (ESG) provides Adabas SQL Server in client/server mode with the following functionality:

- CSCI Communication Services
- Multi-user and Threading capabilities.

This section briefly describes the Adabas SQL Server environment in client/server mode, and covers the following topics:

- server architecture,
- server identification and registration,
- server configuration including product parameter interaction,
- resource requirements and limitations,
- database interfaces and usage.

This section also contains information concerning:

- multi-user considerations,
- performance considerations, and
- client/server communication.

For information about TP-Monitor support and ODBC-Communication in client/server mode, please refer to the Phase D “Installation Steps for Clients” in the chapter **VSE System Installation**.

Server Architecture

This chapter is valid for the VSE platform only and provides an overview of Adabas SQL Server operating under Entire Service Manager (ESG).

Modules

The following diagram provides an overview of the Adabas SQL Server and Entire Service Manager (ESG) modules. The individual modules and their usage are described below.

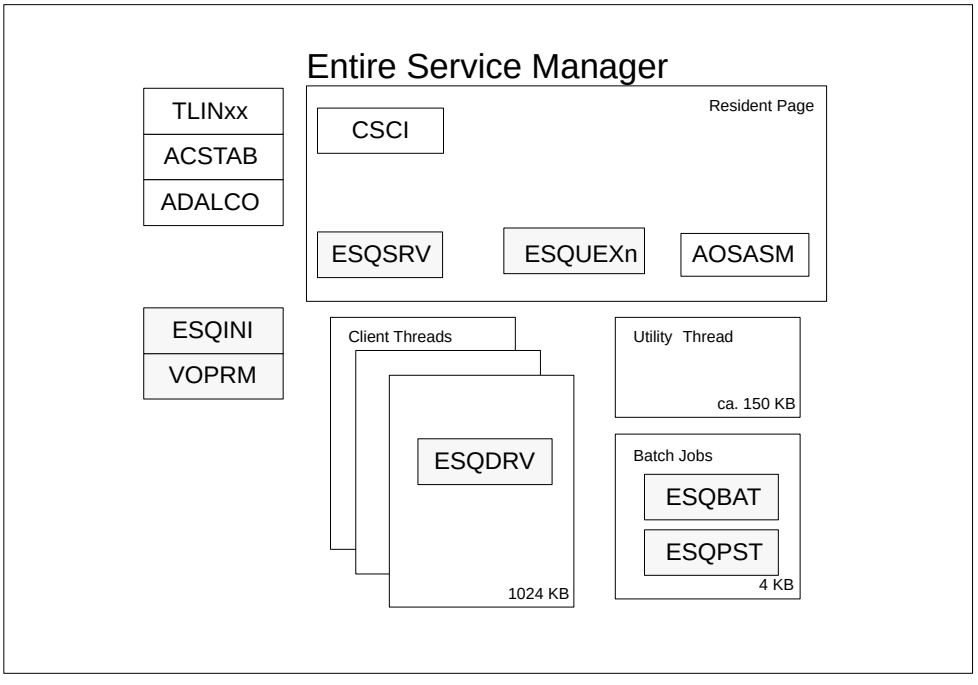


Figure 2-1: Server Components – Modules

Installation Modules

The following modules are created during the installation:

Module	Description/Action
ACSTAB	This module provides a batch interface to Entire Service Manager (ESG) and is created in installation Step **B.14**.
ADALCO	Entire Service Manager (ESG) issues Adabas calls through this module. It must be linked into Entire Service Manager user sublibrary, as described in installation Step **B.6**.
VOPRM	This module contains the Adabas SQL Server VO-Parameter and is created in installation Step **B.11**.

Sublibrary Definitions

The following Adabas SQL modules must be defined to Entire Service Manager (ESG) either using the sublibrary utility *ULIB (on-line) or TULIB (in batch).

Module	Description/Action
ESQDRV	is loaded into the client thread at the time of execution.
ESQPST	is used to post the client thread during CREATE INDEX processing.
ESQBAT	calls the post routine to notify the client thread.

Any of the modules listed above, when not installed correctly, may cause the server to malfunction and/or terminate abnormally. It is therefore important, that the modules be installed as described.

Resident Page Entries

The modules, which are to be loaded by Entire Service Manager (ESG) into the resident page, require a RESIDENTPAGE-parameter in the SYSPARMS parameter file, as described in the installation Step **B.12**.

Module	Description
CSCI	CSCI Communication Interface
AOSASM	Adabas On-line Services
ESQSRV	Adabas SQL runtime system
ESQUEX5	*(Optional)* Adabas SQL User Exit 5
ESQUEX6	*(Optional)* Adabas SQL User Exit 6

Adabas SQL User Exits 5 and 6 are optional. When defined in the VOPRM module, the user exits should always be loaded into the resident page.

For more information concerning the usage and naming conventions of user exits, please refer to the section “User Exits” of this chapter.

Server Initialization

The module ESQINI is loaded during Entire Service Manager (ESG) startup. This module initializes Adabas SQL Server and loads the VO-parameter module (VOPRM) form the user sublibrary.

Upon completion of the initialization tasks, ESQINI returns control to Entire Service Manager (ESG) for registration. Server registration is discussed in greater detail below.

Dataset Requirements

The following diagram provides an overview of the datasets used by Adabas SQL Server and Entire Service Manager (ESG). A brief description of the individual datasets is provided below. For more information concerning Adabas SQL dataset requirements, please refer to the chapter **Operating Adabas SQL Server**.

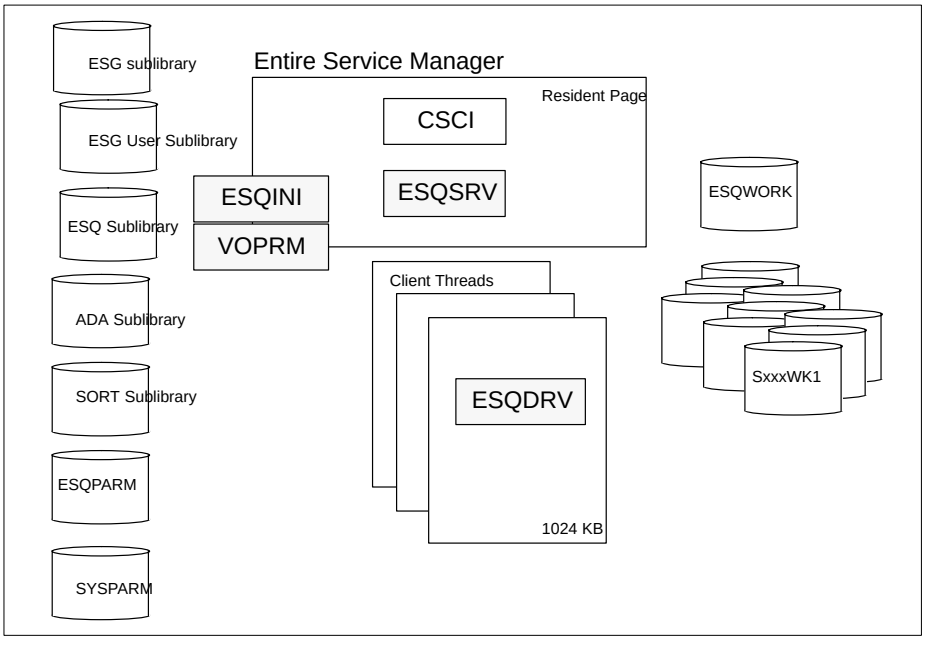


Figure 2-2: Server Components – Datasets

Sublibrary Requirements

Entire Service Manager (ESG) loads modules from the following sublibrary:

- Entire Service Manager sublibrary

Adabas SQL Server requires, that the following sublibraries to be added to the LIBDEF search chain in Entire Service Manager (ESG) startup JCL. This is described in the installation **Step B.17**:

- Adabas SQL sublibrary
- Adabas sublibrary
- Entire Service Manager user sublibrary
- External sort sublibrary

Note:
Please ensure that the sublibrary containing the VOPRM module has been included in the LIBDEF search chain of your Entire Service Manager startup JCL.

Parameter Files

Adabas SQL Server takes parameters from the following files:

- Entire Service Manager (ESG) parameter file (SYSPARM, embedded in ESG startup JCL)
- Adabas SQL parameter file (ESQPARM)

The VO parameter module (VOPRM) can also be used to configure Adabas SQL Server; e.g. using environment variables and the VO parameters.

Tuple Manager Work Files

The Tuple Manager, a component of Adabas SQL Server, utilizes temporary files for storing and sorting intermediate results. Even though, a detailed explanation of the Tuple Manager is provided later in this chapter, a reference to the dataset requirements should be made here.

The Tuple Manager requires a work file for storing the intermediate results. This file is accessed using the dataset name ESQWORK.

The external sort routine, which is called by the Tuple Manager for sorting larger amounts of data, requires one or more work file triplets for sorting. The sort work files triplets are assigned using the dataset names SxxxWK1, SxxxWK2 and SxxxWK3; where the prefix “Sxxx” represents the triplet number. The triplets start with “S001” and must be numbered in ascending order.

The number of sort work file triplets, which are to be assigned, is dependent upon the number of Entire Service Manager (ESG) threads. The minimum number of sort work file triplets is “one”, the maximum number should not exceed “the number of Entire Service Manager (ESG) threads minus one”.

The following dataset names must be included in the Entire Service Manager (ESG) startup procedure:

Dataset Name	Description
ESQWORK	Tuple Manager work file
SxxxWK1–3	Sort work file triplets

The Tuple Manager dataset requirements and naming conventions are described in the section **Tuple Manager Interface** of this chapter.

Diagnostic Output

Adabas SQL Server in client/server mode produces diagnostic output, when the logging or trace facilities have been activated. In such cases, include the following dataset names in the Entire Service Manager (ESG) startup procedure:

Member	Description
ESQCLOG	Adabas SQL Server Adabas Command Log File
ESQDMPG	Adabas SQL Meta-Program Dump File
ESQTRAC	Adabas SQL Trace File

The assignment of these dataset names is optional.

Please refer to the chapter **Logging Facilities**, for further information about the usage of these datasets.

Job Procedures

In the mainframe environment, Adabas SQL Server generates and submits batch jobs when performing INDEX maintenance (i.e. to process CREATE/ALTER INDEX statements). Jobs generated then use the ADAINV Utility (**INVddddd.J**) created in Step **C.3** of the installation.

Thread Requirements

Adabas SQL Server Threads

Entire Service Manager (ESG) provides Adabas SQL Server with threading capabilities.

The number of threads, which are required:

- is dependent upon the number of clients, which are to process concurrently and
- is limited by the amount of system resources which are available.

A minimum of three (3) threads in a production environment is recommended.

Adabas SQL Server requires a minimum of 1024KB per thread.

Special Considerations

Adabas SQL Server requires that the threads be located below the 16MB line.

The maximum thread size available to Adabas SQL Server is restricted to 1024KB.

The definition of one additional thread is recommended. This enables the execution of Entire Service Manager utility programs parallel to Adabas SQL Server. This thread can be defined with minimal size of 150KB.

The number of Adabas SQL Server threads should exceed the number of sort work file triplet, which are assigned using the dataset names SxxxWK1 through SxxxWK3. This is an important performance consideration, as the sort processing temporarily blocks the server thread.

The CREATE INDEX statement processing temporarily can cause the blockage of server threads. These threads are posted by the ESQPST module upon completion of the batch processing.

Thread Definition

The number and size of threads, which are to be started, are specified in the Entire Service Manager (ESG) parameter file (SYSPARMS) using the THREADS parameter:

```
THREADS=(t0,t1,t2,...,tn)
```

Where:

- the number of threads is determined by the number of *tn* parameters, and
- the thread size is determined by the value of the *tn* parameter.

The THSIZEABOVE parameter may not be used to define Adabas SQL Server threads. For example, if the parameter is given as follows, four threads are initialized:

```
THREADS=(150,1024,1024,1024)
```

- one thread with 150KB for Entire Service Manager utility programs, and
- three threads with 1024KB for Adabas SQL Server.

Troubleshooting

If the thread size be insufficient (smaller than 1024KB) or the module ESQDRV is not cataloged with a thread size of 1008 KB, the server session will issue the following console message and terminate abnormally:

```
ESQDRV – RESPONSE CODE: 0778
```

Entire Service Manager (ESG) does not post the client session in such circumstances, which causes the client session to hang.

Server Identification

Adabas SQL Server is identified using the server name. It is used to register the server with the communication services, in this case Net-Work / CSCI.

The server name is a string literal consisting of maximum 8 characters.

The server name is specified by the Entire Service Manager (ESG) SERVER-Parameter.

```
SERVER=(server_name, ... )
```

The value of the SERVER-Parameter must be unique. No other servers or services may register with the Net-Work using this name. The name must be unique on the node.

During the server initialization, the value of this parameter is used to set the ESQSRV environment variable.

In order for the client to communicate with the server, the server name must be entered in the server routing file/table of the client environment.

The description of the server routing file/table (ESQSRVRT) and it's usage is provided in the *Adabas SQL Programmer's Guide*. On the mainframe platforms, the server routing file / table is implemented in the VO Parameter Module (VOPRM) using the ESQRTAB macro. A detailed description of VOPRM is provided below.

Note:
The SERVER NAME parameter in the Adabas SQL Server ESQPARMS file is not used on mainframe platform.

The values of the following Entire Service Manager (ESG) parameters must be unique: SERVER, INSTALLATION,VTAMAPPL.

Warning:
Currently, the server name on the mainframe platforms may not contain lowercase characters (Sagsis Problem 162245).

Server Registration

Entire Service Manager registers the server with the Net-Work / CSCI services.

The following Entire Service Parameters are used by Entire Service Manager to register the server:

Parameter	Description
SERVER	the server name
ACCESS-SVC	the number of the Adabas ROUTER SVC
ACCESS-ID	the unique node-id
ACCESS-LOCAL	the scope

Entire Service Manager, irregardless of whether Entire Net-Work is installed or available, attempts to register the server with Net-Work / CSCI services. In the cases where Net-Work is not installed or active, this results in error message(s) during the server initialization and shutdown phases:

```
ESQSVR0014–* Server ESQXXXX Register failed.

ESQSVR0015–* Server ESQXXXX Deregister failed.
```

In spite of the error messages issued, the server is still accessible in LOCAL client/server mode using the Adabas ROUTER services (SVC).

Please note, for REMOTE client/server communication Net-Work is required.

The Server Overview (SO) function of the UCTRL – MONITOR CONTROL utility enables the administrator to query the server status and identification.

The SERV operator command can be used to register and de–register the server. The following commands, for example, de–register and re-register Adabas SQL Server (ESQXXXX) from Entire Service Manager (ESGSRV):

```
reply–id SERV ESQXXXX,DREG

reply–id SERV ESQXXXX,REG
```

A description of both the SERV operator command and the UCTRL utility can be found in the *Entire Service Managers User's Guide and Utilities Manual*.

Configuring the Server

This section provides the system administrator with an overview of those product parameters, which can influence the client/server configuration. These parameters should be modified to reflect the installation and application requirements.

The following topics are covered in this section and should be taken into consideration when configuring the client/server environment:

- Number of Concurrent Users
- Communication Buffer Sizes
- Time-Out Values

This section also lists those product parameters, which restrict the usage of key resources. It also provides a list of those resources, which limit the processing of intermediate results.

For further information on the parameters mentioned in this section, please consult the appropriate product manuals.

Number of Concurrent Users

The following table contains a list of product parameters which may require modification, when increasing the number of concurrent Adabas SQL Server clients.

Product	Parameter	Default Value
Adabas SQL Server	SERVER MAX MULTIUSER PROCESSES	64
Entire Service Manager	ACCESS-NCQE	5
Entire Net-Work	NC	200
Adabas	NU	200

Server Resources

The Adabas SQL Server parameter SERVER MAX MULTIUSER PROCESSES limits the number of client sessions, which can be concurrently managed (processed) by the server. This parameter is used to calculate the size of internal buffers and limits servers resource requirements. Remember, that a client session starts with the processing of a CONNECT statement and ends when either session is timed-out or a DISCONNECT statement is processed.

Specifying a large value for the SERVER MAX MULTIUSER PROCESSES parameter increases the server's resource requirements. To compensate for this increase, please modify the value of the REGION parameter in the Entire Service Manager startup procedure.

Each client, which is to be concurrently processed requires Tuple Manager resources (ESQWORK) for the storage of intermediate results. Please check the sizes and number of the following datasets and adjust the space allocation as deemed necessary:

Dataset	Description
ESQWORK	Adabas SQL Server temporary work file
SxxxWKn	Sort work file triplets

The overall system throughput of Adabas SQL Server could suffer under the additional user load. This performance loss can in part be counter-balanced, by increasing the number of threads and sort work file triplets, which are available to Entire Service Manager for processing.

Entire Service Manager threads are configured using the THREADS parameter in the SYSPARM file and is described above.

The number of sort work file triplets (SxxxWKn) assigned in the Entire Service Manager startup procedure. Please note, that the number of triplets assigned should not exceed the number of available threads.

Communication Resources

The values assigned the ACCESS-NCQE and NC parameters specify the number of command queue entries. The parameters limit the number of clients, which can communicate with Adabas SQL Server. Both of these values should be large enough to handle the expected number of Adabas SQL clients, which are to be concurrently processed.

Database Resources

The Adabas parameters NC (Number of Command Queue Entries) and NU (Number of Users) limit the number of users, which can be serviced by the database. These parameter values should be increased proportionally.

Note, that each Adabas SQL client can have multiple (up to three) unique Adabas session contexts. This is discussed below in the section describing the Entire Service Manager database interface. For information on this subject can be found in **Appendix D, Adabas SQL Server and Adabas** in the *Adabas SQL Server Programmer's Guide*.

Communication Buffers

The following product parameters limit the size of the Adabas SQL client/server communication buffers. The amount and size of data, which must be transported, is application dependent. Ensure that the communication services have sufficient resources to handle the required Adabas SQL request/reply length.

Product	Parameter	Default Value
Adabas SQL Server	SERVER	
	MAX REQUEST LENGTH and	8000
	MAX REPLY LENGTH	8000
Entire Service Manager	ACCESS-NABS	3
Entire Net-Work	NAB	16
Adabas	NAB	16

Server Resources

On the mainframe platforms, the values of the SERVER MAX REQUEST and SERVER MAX REPLY LENGTH parameters are restricted to a maximum of 32000 bytes.

Communication Resources

The values assigned the ACCESS-NABS and NAB parameters specify the number of (internal) attached buffers, which are allocated for interregion communication. The size of the attached buffer is determined by the number of buffers to be defined multiplied by 4112 bytes. The following formula can be used to calculate the value for the number of attached buffers (ACCESS-NABS and NAB):

```
NAB = (max_multiuser * max_request_reply_length ) / 4112
```

Database Resources

The formula above, cannot be used to calculate the value of the Adabas NAB parameter. It does not influence the client/server communication, it does though limit the resources available for the database communication. This parameter has been included here, as a reminder. The value assigned should be large enough to handle the application requirements.

Time-Out Checking

The following product parameters influence the Adabas SQL client/server time-out settings. These parameters are application and system configuration dependent and should be modified as deemed necessary.

Product	Parameter	Default Value
Adabas SQL Server	RUNTIME	
	SERVER SESSION TIMEOUT	15 Minutes
	and	
	SERVER REPLY TIMEOUT	15 Minutes
Entire Net-Work	REPLYTIM	60 Seconds
	CSC_TIMEOUT	60 Seconds

The diagram provided below shows the inter-action of the different product parameters and where the time-out checking is performed:

Server Non-Activity is specified using the SERVER REPLY TIMEOUT and is performed by the client. The value of this parameter should be smaller than the value which is assigned the CSCI communication services parameter: REPLYTIM and CSC_TIMEOUT.

Client Non-Activity is specified using the SERVER SESSION TIMEOUT and is performed by the server.

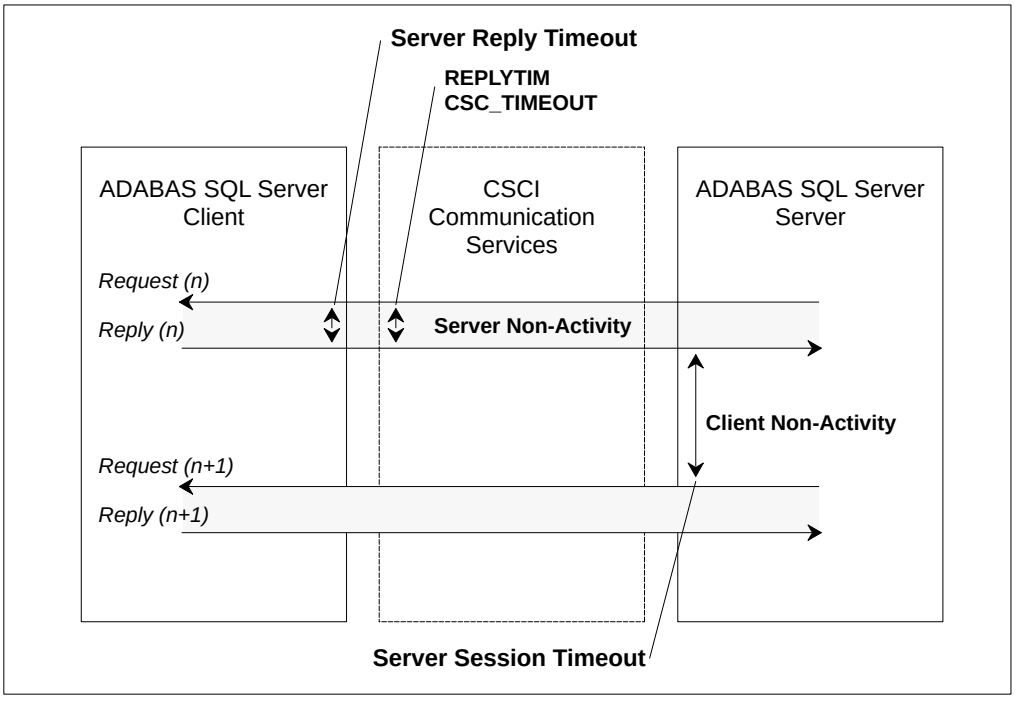


Figure 2-3: Client/server Time-Out Checking

Session / Reply Time-Out

Time-out checking is only performed in client/server mode, and not in LINKED-IN mode. It is performed on both sides of the communication services: by the client and the server.

- SERVER SESSION TIMEOUT specifies the client non-activity time (idle time). It is checked by the server and prevents server threads from being occupied by “lazy” or “dead” clients.
- SERVER REPLY TIMEOUT specifies the server non-activity time-out (response time). It is checked by the client and prevents the client from hanging up if the server does not respond to the client’s request.

The values of the Net-Work CSCI communication parameters:, REPLYTIM and CSC_TIMEOUT ***MUST*** exceed the SERVER REPLY TIMEOUT parameter value.

For further information on the subject the time-out checking, please refer to the chapter “client/server Topics” in the *Adabas SQL Server Programmer’s Guide*.

CSCI Communication

The communications services should not perform any time-out checking. This can be achieved by assigning a (very) large value to the appropriate communication services parameters.

The value used to perform time-out checking is implemented in the CSCI/CPI layer and is specified by the client and can be set:

- using the environment variable CSC_TIMEOUT for Windows / UNIX clients, and
- using the Entire Net-Work parameter REPLYTIM for mainframe clients.

Database Communication

The following Adabas parameters do not directly influence the client/server communication, but they can restrict the server processing. These parameters should be taken into consideration when evaluating the applications time-out requirements.

Parameter	Description
CT	Command time limit
TT	Transaction time limit (User Call)
TTSYN	Transaction time limit (Utility Call)
TNAA	Non-activity time limit (access-only users)
TNAE	Non-activity time limit (ET logic users)
TNAX	Non-activity time limit (exclusive update users)
TLSCMD	Time limit for Sx commands

Restrictions and Processing Limitations

The following product parameters can limit the Adabas SQL client/server processing. These parameters are application and system configuration dependent and should be modified as needed.

Product	Parameter	Default Value
Entire Service Manager	ADACALL	0,5
	ADALIMIT	4096
	ADAROLL	10
	CPUTIME	(2,2,2,...)
	REALTIME	(3,3,3,...)

Recommendations

It is not recommended to limit the Adabas or thread resources, which are required to process a client's request; e.g. those resources required to process a single SQL statement. Therefore, the following values are strongly recommended:

Parameter	Comment
ADAROLL=NO	Specifies that the client session will not be rolled out of the thread will waiting for an Adabas call.
ADACALL=xxxx	If ADAROLL=NO is specified, this parameter is ignored.
ADALIMIT=0	If ADALIMIT=0 is specified, this parameter is ignored.
CPUTIME=(t0,t1,...tn)	Specifies the maximum number of seconds (CPU time) allowed for the processing of a single SQL statement, before terminating the server thread abnormally.
REALTIME=(t0,t1,...tn)	Specifies the maximum number of seconds (elapsed time) allowed for processing of a single SQL statement before issuing warning messages to the console.

Database Resources

The amount of database resources required by an application are dependent upon:

- the type SQL operations which are to be executed,
- the amount of data which are to be selected for processing, and
- the application's transaction logic.

The processing of SQL statements can be very complex. A single SQL statement often results in the execution of many Adabas direct commands. The amount of data processed is solely dependent upon the selection criterion (WHERE-Clause) and the data volume at the time of execution. Poorly formulated or poorly qualified SQL statements can result in the processing of (extremely) large quantities of data.

The Adabas parameters listed below are only a few of the many parameters, which influence or restrict the Adabas SQL processing. A list of the recommended MINIMUM Adabas parameter values is provided in installation step **A.1**.

Parameter	Description
TLSCMD	Time Limit for Sx Commands
LP	Length of the Data Protection Area

The following database resources could restrict the processing of SQL statements:

- Size of the Data Protection Area
- Size of the Intermediate Results Area
  (Work Part 2, Work Part 3 and the Tuple Manger Work File)
- Size of the Unique Descriptor Pool
- Number of User Queue and Command Queue Entries
- Number of Command IDs per User
- Number of Hold Queue Entries per User

For further information concerning the processing of SQL statements, please consult the chapter **Understanding SQL Query Translation** in the *Adabas SQL Server Programmer's Guide.*

Database Interface

Adabas Router SVC

When executing in client/server mode WITHOUT using Entire Net-Work, please ensure that both the database and the server are accessible using the Adabas Router SVC.

Adabas Link Module

Adabas SQL Server communicates with the database using the Adabas link module ADALCO. This module is intended for usage with applications executing under the control of Entire Service Manager.

Adabas SQL Server requires that the module ADALCO be defined with the following attributes: AMODE(31), RMODE(ANY), RENT.

Warning:
The Adabas link module must be link-edited with the attributes AMODE=31, RMODE=ANY and RENT. If this module is not defined with these attributes, the results are unpredictable. When the executable code is loaded above 16MB, the server may terminate abnormally during the server initialization.

Adabas Utilities

The Adabas Online Services module (AOSASM) is used by Adabas SQL Server to interface with the Adabas utilities. This module is called during the processing of DDL statements (Data Definition Language) with one exception. The CREATE INDEX processing is performed by the Adabas Utility ADAINV.

In the mainframe environment, Adabas SQL Server in client/server mode the CREATE INDEX statement processing is performed in batch. To do this, the server creates and submits batch jobs based on the ADAINV utility JCL (**INV*ddddd.J***), which were created in step **C.3** of the installation. The server process, after job submission, waits for completion of the batch job. In the final two steps of the **INV*ddddd.J*** JCL, the module ESQBAT is used to pass the completion code to the waiting server process.

Adabas Interface

Adabas SQL Server uses the following Adabas functionality:

Adabas Session Context: Each client using Adabas SQL Server may require up to three independent Adabas session contexts (Adabas User Queue Elements).

Adabas OP Command: Adabas SQL Server issues an explicit Adabas OP command when the Adabas nucleus is addressed in a client-specific manner for the first time. This is done for each of the above mentioned Adabas session contexts.

ADDITIONS1 Field: The ADDITIONS1 Field of the Adabas control block may be used to ship a user identification to the Adabas nucleus. the usage of '*' in the first byte implies a transparent processing by the addressed Adabas nucleus. The Adabas user exits may not modify this field.

Command IDs: Adabas SQL Server makes use of the Adabas command IDs. Within Adabas SQL Server, there are two types of command IDs: internal and temporary command IDs. A single user may have upwards of 32 command IDs.

Global Format IDs: Adabas SQL Server uses Global Format IDs to optimize the usage of the Adabas format pool.

Further information concerning the server's usage of the Adabas interface and the processing of SQL statements, please consult **Appendix D, Adabas SQL Server and Adabas** and the chapter **Understanding SQL Query Translation** in the *Adabas SQL Server Programmer's Guide.*

Adabas User-Exits

Note:
The Adabas user exits are not allowed to modify the user identification, which is shipped in the ADDITIONS1 field.

The use of Adabas user exits with Adabas SQL Server is generally allowed. But, the user exits should not modify the contents of the Adabas control block of those calls which are issued by Adabas SQL Server.

VO – Parameter Module (VOPRM)

VOPRM Module Overview

The VO parameter module (VOPRM) is required in both the client and the server environment. The module VOPRM provides Adabas SQL Server with the following functionality:

- Environment variables, though common on other platforms, are not available on mainframes and are thus emulated. These variables are used at runtime by Adabas SQL Server and precompiler, for example: to set option switches, to define the server name and to activate the logging facilities.
- Server routing file has been implemented as a table within the module VOPRM and provides routing information for both the client and the server.
- Code conversion table is used in DB2-Mode to convert SQL codes.

Additionally, the module VOPRM contains the VO Parameter settings. These parameters are required and influence the processing:

- CICS client environment parameter determines for example: the mechanism which is to be used to call the Adabas CICS LINK module (ADALNC) and the size of the memory which is available to the Adabas SQL Server client-application.
- Sort processing parameter limit the size of the internal sort buffer. This value influences the performance and determines when external sort processing is required.
- External Security parameter de-/activate the external authentication interface.

Creation and Usage

A VOPRM module is created during the installation process and may be customized to suit the requirements of the site. Typical reasons for customizing the VOPRM module could be:

- to increase/decrease the size of the internal sort buffer,
- to set a default server name, or
- to modify/add server routing file entries.

The VOPRM module is ***required*** in both client and server environments and is loaded during Adabas SQL initialization. It defines the default attributes and provides routing information for both Adabas SQL Server and client applications.

Note:
There is a restriction that only one VOPRM module is allowed within an address space; for example, under TP-environments such as: CICS, Com-plete, Batch etc.

Refer to step **B.11** for information on assembling the VOPRM module.

In those cases, where more than one VOPRM module is required to support the different client/server environments, the different VOPRM modules must be stored in separate sublibraries. Be aware, that the server routing information in both the client and the server environments must be identical.

The following diagram shows the loading of the VOPRM module:

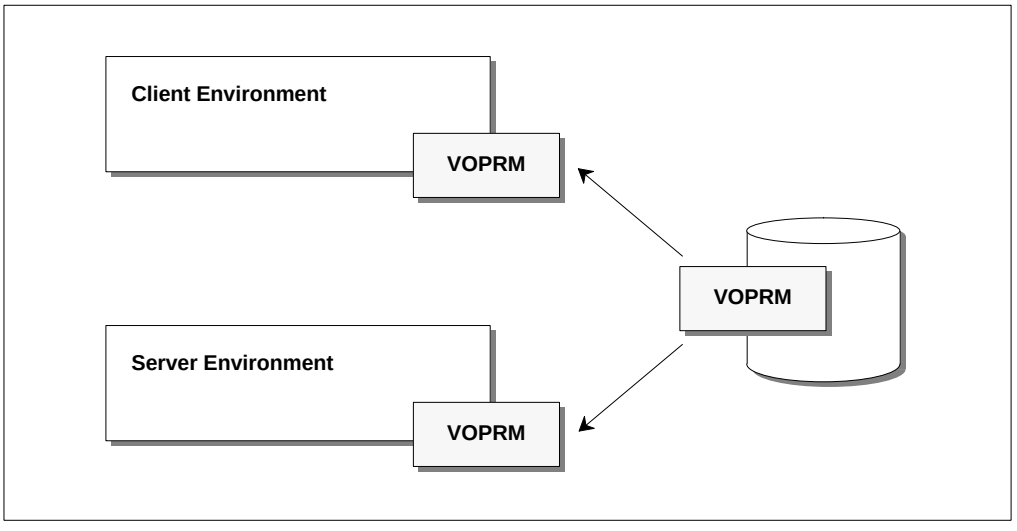


Figure 2-4: Loading the VOPRM Module

Module Layout

The module VOPRM is created from an assembler source and consists of one or more of the following macros:

– **VOPRM Macro**
  Defines the root of the VOPRM module and is *required*. This macro must be the first entry within the source file. It is used to set the VO Parameters.

– **VOENV Macro**
  Defines an entry in the environment variable table. An entry must be made for each environment variable, which is to be added to the table. This macro is *optional*.

– **ESQRTAB Macro**
  Defines an entry in the Server Router file (or rather – table). An entry must be made for each server which is to accessed. This macro is *optional*, but highly recommended.

– **ESQCT Macro**
  Defines an entry in the Code Translation Table. An entry must be made for each Adabas SQL response code, which is to be translated. This macro is *optional*.

The following diagram displays the layout of the VOPRM module:

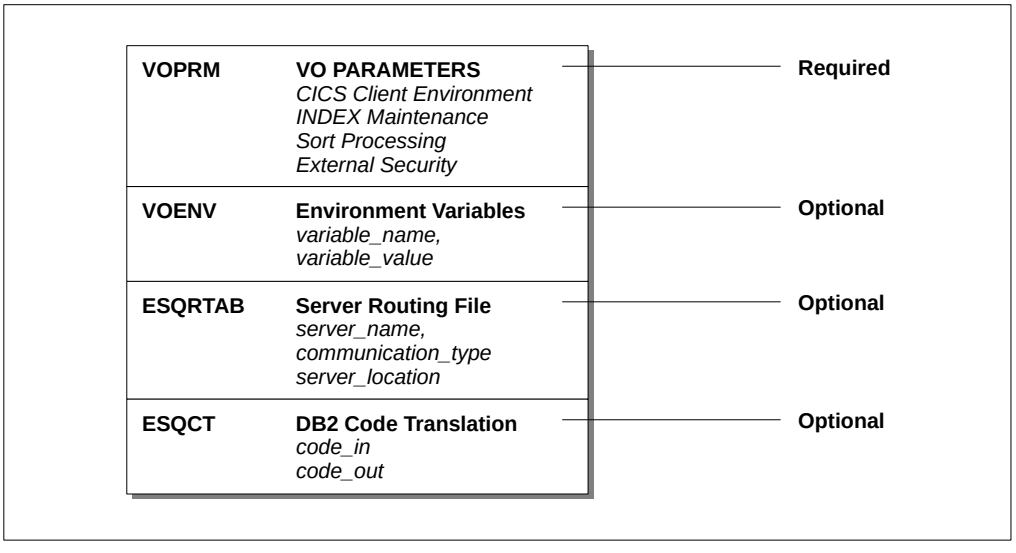


Figure 2-5: VOPRM Module Layout

The order in which the macros appear within the source is not of importance, with the one exception that the module must begin with the VOPRM macro.

A detailed description of the macros and their parameters is provided in a later section of this chapter.

Code Example

Included below is an example of a VOPRM module:

```
*--------------------------------------------------------------------*
* FILE NAME:      VOPRM
*
* DESCRIPTION:    ADABAS SQL VO-PARAMETER MODULE
*                 CONTAINS ...
*                 ... VO PARAMETERS              (Required)
*                 ... ENVIRONMENT VARIABLE       (Optional)
*                 ... SERVER ROUTER TABLE        (Optional)
*                 ... CODE TRANSLATION TABLE     (Optional)
*--------------------------------------------------------------------*
*    VO PARAMETERS                               (Required)
*--------------------------------------------------------------------*
    VOPRM    SORTBSZ=40,       Internal SORT Buffer Size              +
             EXTSEC=NO         External Security Interface
*--------------------------------------------------------------------*
*    ENVIRONMENT VARIABLES                       (Optional)
*--------------------------------------------------------------------*
    VOENV    NAME=ESQSRV,                                             +
             VALUE=ESQ9000     <Server Name>
    VOENV    NAME=ESQLOG,      ***********  Diagnostics  *********     +
             VALUE='CMD,CSC'   Command Logging, C/S Communication      +
                               **********************************
*--------------------------------------------------------------------*
*    SERVER ROUTER TABLE                         (Optional)
*--------------------------------------------------------------------*
    ESQRTAB  SERVER=ESQ9000,   <Server Name>                          +
             ID=IBM1           <Net-Work ID>                          +
             NODE=09000        <ACCESS ID>
*--------------------------------------------------------------------*
*    CODE TRANSLATION                            (Optional)
*--------------------------------------------------------------------*
    ESQCT    CODEIN=-8006,     ESQ Unique Constraint Violation        +
             CODEOUT=-803,     DB2
    END
```

VO Parameters

The following VO Parameters can be defined using the VOPRM macro. These parameters are the only ones required for Adabas SQL Server under VSE; all other parameters are optional.

Parameter	Description
ADALINK	Mechanism used to call Adabas LINK Module (CICS)
GLOBLSZ	Size of Global Memory when executing under TP Monitors (CICS)
SORTBSZ	Size of the Internal SORT Buffer
EXTSEC	Switch to activate the External Authentication Interface

A detailed description of these parameters is provided in the section **VOPRM Macro** later in this chapter.

Environment Variables

Adabas SQL Server's Precompiler and Runtime System use environment variables to assign the (default) server name, or to enable special features and options, such as logging and tracing.

Environment variables are mostly used to activate/deactivate non-standard features. The presently available environment variables are documented below:

Variable Name	Description
ESQSRV	Identifies the name of the active Adabas SQL Server in the server environment. Identifies the default target server, where SQL requests will be routed to, in the client environment. The client environment uses routing information stored in the server routing file (see below) when communicating with the target server.
ESQUSER	Defines the (default) user ID and password, which is used when implicit CONNECT statements are generated by the BASIC and ADVANCED Interactive Facilities.
ESQLOG	Activates the Adabas SQL logging facility, and can be used to log SQL commands and the client/server communication.
ESQMITRC	Activates the Adabas SQL mini-trace facility, which is only available within the TRACE version of Adabas SQL Server. This facility should *only* be used under the guidance of a Software AG service representative.

Values can be assigned to an environment variable using the VOENV macro in the VOPRM module. For further information, refer to the sections discussing the VOENV macro and the code example for the VOPRM module.

```
VOENV    NAME=ESQSRV,VALUE=server_name
```

Environment variables can also be assigned using the PARM keyword of the EXEC JCL-Statement. However, the usage of this feature should be restricted to the Adabas SQL Server Utilities, in particular the BASIC Interactive Facility.

```
// EXEC  ESQBIF,PARM='ESQSRV=server_name'
```

Note:
There should always be an entry in the server routing file for the server name defined by the environment variable ESQSRV.

Server Routing File

A server routing file is required for client/server communication. It identifies the servers with which a client application can communicate. It contains the server's name, location and communication type; e.g. LINKED-IN or CSCI.

The contents and function of the server routing file is the same across all platforms. The mainframe implementation of this feature differs slightly from that on other platforms. Here it is not a file, but rather a table defined within the VOPRM module.

Entries in the server routing file are made using the ESQRTAB macro:

```
ESQRTAB  SERVER=server_name,
             ID=net-work_id,
             NUMBER=node-number,
             TYPE=communication_type
```

Macro Parameter	Description
SERVER=	The name of the server, as defined in the SERVER parameter of the Entire Service Manager SYSPARM file. The length is 8 characters.
ID=	The network ID of the server to be accessed. The length is 8 characters.
NUMBER=	*(Optional)* The node number of the server, as specified in the ACCESS-ID parameter of the Entire Server Manager SYSPARM file. This parameter must only be specified, when the server is local and communication is to be handled by the router (SVC).
TYPE=	The communication type to be used with this server. The value of this parameter defines whether the server is to be accessed in client/server mode (CSCI) or in linked-in mode (LINKED-IN).

Further information about the macro parameter and their possible values is provided later in this chapter.

It is important, that the server routing file be the same on both the client and server side. The server uses the information stored within the server routing file to register itself, and the client uses it to locate the server.

Note:
There should always be an entry in the server routing file for the server name defined by the environment variable ESQSRV.

DB2 Code Translation

Some DB2 applications cannot process Adabas SQL response codes. Adabas SQL Server provides functionality to translate Adabas SQL Server response codes to DB2 response codes.

The Code Translation Table is a part of the VOPRM module. The ESQCT macro entries defines the SQL codes, which are to translated.

```
ESQCT    CODEIN=adabas_sql_code,
         CODEOUT=db2_code
```

A list of Adabas SQL response codes, which are typically translated, is provided below:

Adabas SQL Response Code	DB2 Response Code	Description
–8001	–811	A single row select statement produced a resultant table which contains more than one row when executed. Such an embedded SELECT statement may only produce one row or the empty set.
–8005	–811	A subquery which was not in an EXISTS nor in a predicate nor was it qualified with the keywords 'ANY', 'SOME', or 'ALL' resulted with more than one row in its resultant table. This is not permitted.
–8110	–502	An attempt was made to open a Cursor which is already open.
–8111	–501	An attempt was made to access a Cursor which is currently not open.
–8115	–802	During an internal arithmetic calculation a division by zero error condition was encountered.
–8118	–305	A target host variable has been specified without an accompanying indicator variable and the corresponding field has returned null.
–8134	–922	The current statement was terminated because the user has not been granted the necessary access rights to one or more data structures referenced in the statement.
–9009	–911	An Adabas response code 9 was received while accessing the Adabas SQL catalog. The user transaction has been rolled back.
–3198	–803	Duplicate key error has occurred processing a user table; e.g. an attempt was made to duplicate a descriptor value for a unique descriptor

Adabas SQL Response Code	DB2 Response Code	Description
–9198	–803	Duplicate key error has occurred processing The Adabas SQL catalog; e.g. an attempt was made to duplicate a descriptor value for a unique descriptor
–8006	–803	An attempt was made either using INSERT or an UPDATE statement to amend the contents of a target table in such a way that a unique constraint was violated.
–9861	–818	Metaprogram in the Adabas SQL catalog is not consistent with the application program.

The VOPRM Macro

The VOPRM macro creates the module header and thus, must be the first macro in the source module VOPRM. All parameters are preset with default values.

VOPRM Macro Syntax

The syntax is:

```
VOPRM    [ADALINK=(name,type,location),]
         [EXTSEC=Y/N,]
         [GLOBLSZ=global_memory,]
         [HANDLSZ=handle_table_size,]
         [SORTBSZ=int_sort_buf_size,]
         [SORTNAM=external_sort_name,]
         [SORTTYP=external_sort_type]
```

Example:

Assign the handle size of 8192 bytes and set the internal sort buffer size of 40:

```
VOPRM HANDLSZ=8192,TEMPBSZ=40
```

VOPRM Macro Parameters

The individual parameters which can be specified in the VOPRM macro are described below:

ADALINK

Possible values:	*(name,type,location)*
Default value:	(ADABAS,B,R1)

This parameter describes the mechanism used to call Adabas CICS Link module (ADALNC) within a CICS environment. It consists of 3 sub-parameters:

- The *name* of the CICS Adabas link module
- The *type* of module linkage
  B using BALR
  L using CICS LINK
- The *location* of the Adabas parameter list
  R1 using register 1
  TWA using CICS TWA
  TWA+*nnn* using CICS TWA offset (*nnn*=bytes)

Note:
For further information concerning the location of the Adabas parameter list refer to chapter ***Installing Adabas with CICS*** *in the* ***Adabas Implementation and Maintenance Manual****.*

Note:
For better performance it is recommended to use the default value.
Option B (call using BALR) and R1 (Adabas parameter list address in R1).

Dependent upon the site requirements under CICS an offset in the TWA can be optionally specified by suffixing the TWA parameter as follows: TWA+256, in which case the decimal offset will be 256 bytes. The default offset value is +0.

Warning:
Any changes in the TWA offset must be reflected in the relevant ADALNC module. The ADALNC supplied by Software AG expects the parameter list to start of offset +0 in the TWA.

EXTSEC

Possible values:	Y/N
Default value:	N

This parameter specifies whether or not the Adabas SQL Server External Authentication Interface should be used.

GLOBLSZ

Possible values:	*n*
Default/Minimum value:	4096 (Bytes)

This parameter determines the size of the global memory for a VO application running under TP monitors such as CICS.

HANDLSZ

Possible values:	*n*
Default/Minimum value:	8192 (Bytes)

This parameter determines the size of the handle table for file and shared memory access.

SORTBSZ

Possible values:	*n*
Default value:	40 (KB)

This parameter determines the size of the internal sort buffer and will be rounded up to the next 4 KB.

SORTNAM

Possible values:	*name*
Default value:	SORT

This parameter determines the name of the external sort module.

SORTTYP

Possible values:	C, D, S C=CA-SORT D=DFSORT S=SYNCSORT
Default value:	*

This parameter determines the type of the external sort module.

The VOENV Macro

The macro VOENV is used to assign values to environment variables.

VOENV Macro Syntax

Multiple VOENV entries are allowed and the syntax is:

```
VOENV    NAME=environment_variable_name,
         VALUE=variable_value
```

Example:

Assign the value ESQ9000 to the environment variable ESQSRV:

```
VOENV    NAME=ESQSRV,VALUE=ESQ9000
VOENV    NAME=ESQLOG,VALUE='CMD'
```

In the example above, the server ESQ9000 has been chosen as the target server and the SQL command logging facility has been activated.

Note:
This example is continued below by the routing table entries.

VOENV Macro Parameters

The individual parameters which must be specified in the VOENV macro are described below:

NAME

Expected value: *name*

The *name* of the environment variable to which the value is to be assigned. The maximum length of *name* is 8 characters.

VALUE

Expected value: *value*

The *value* to be assigned the environment variable. The maximum length of *value* is 255 characters.

The ESQRTAB Macro

The ESQRTAB macro defines a table of target services available and the necessary communication information.

ESQRTAB Macro Syntax

```
ESQRTAB  ID=network_id,
         NUMBER=node_number,
         SERVER=server_name,
         TYPE=CSCI
```

Example:

Define a routing table entry for the server ESQ9000 with node number (ACCESS-ID) 9000 at the network node IBM1 (using the communication protocol CSCI).

```
ESQRTAB  SERVER=ESQ9000,ID=IBM1,NUMBER=9000
```

ESQRTAB Macro Parameters

The individual parameters which must be specified in macro ESQRTAB are described below.

ID

Expected value:	*network-id*

This parameter determines the network ID of the server to be accessed. The maximum length of *name* is 8 characters. The network ID depends on the type of communication used as the transport layer, for example:

- TCP/IP: *tcp_host_name*
- SNA*:* *vtam_application_name*

NUMBER

Expected value:	*node-number*

This parameter is specified if the server is local and represents the node number of that server which the router (SVC) will handle.

Optional. This parameter is only required for local client/server communication. The value must be the same as the value specified for the ACCESS-ID parameter in the Entire Service Manager SYSPARM file. client/server communication is performed using the Adabas router server (SVC) specified in the ACCESS-SVC parameter.

SERVER

Expected value:	*server-name*

The *name* identifies the server to which SQL requests will be routed. The value assigned should be the same as the server name, which has been, for example on the mainframe servers, specified in the SERVER parameter in the Entire Service Manager SYSPARM file.

The maximum length of *name* is 8 characters.

TYPE

Possible value:	LINKED-IN/CSCI
Default value:	CSCI

This parameter determines the type of communication protocol which is to be used.

The value CSCI is valid for servers running in all environments.

The ESQCT Macro

The ESQCT macro defines a table of error codes to be translated. A list of Adabas SQL response codes, which are typically translated, is provided earlier in this chapter.

ESQCT Macro Syntax

Multiple ESQCT entries are allowed and the syntax is:

```
ESQCT    CODEIN=code_in,CODEOUT=code_out
```

Example:

```
ESQCT    CODEIN=-8134,CODEOUT=-922
ESQCT    CODEIN=-8118,CODEOUT=-305
```

ESQCT Macro Parameters

The individual parameters which must be specified in the ESQCT macro are described below:

CODEIN

Expected value: *code_in*

code_in is the Adabas SQL Server code to be translated.

CODEOUT

Expected value: *code_out*

code_out is the error code to be issued instead of the Adabas SQL code (code_in).

Tuple Manager Interface

The Tuple Manager is a component of Adabas SQL Server, and expands the capabilities of the Adabas nucleus by providing temporary storage, sort, and retrieval functionality for intermediate resultant tables. The implementation of the Tuple Manager utilizes both main memory and temporary work files for the storage and external sorting of intermediate results.

Some SQL constructs require sorting, for example the language constructs ORDER BY and GROUP BY:

```
SELECT * FROM SAGTOURS.CRUISE
ORDER BY DESTINATION_HARBOR

SELECT START_HARBOR, COUNT(*) FROM SAGTOURS.CRUISE
GROUP BY START_HARBOR
```

For more information, please refer to the chapter **Understanding SQL Query Translation and Optimization,** section: **Reasons for External Sorting** in the *Adabas SQL Server Programmer's Guide.*

The Tuple Manager optimizes the sort process by using an internal sort for small amounts of data and by using the external (system) sort for larger amounts of data. The threshold limit, for switching from internal sort to external sort, is determined by the size of the internal sort buffer (see SORTBSZ Parameter of the VOPRM macro).

Should an error condition arise during the Tuple Manager processing, messages are issued to the system console and/or job protocol. These messages are described below and in the *Adabas SQL Server Messages and Codes Manual.*

Temporary Work Files

Temporary work files on the mainframe platforms have been implemented using a subset of the SAG Editor (EDT V1.5 SM2) functionality:

- multi-user capabilities
- temporary file management

The SAG Editor consists of a buffer pool and a work file. The buffer is local to the server and is created during the initialization. The work file is assigned using the DLBL name ESQWORK**.**

The lifetime of the ESQWORK dataset is limited to that of the server, as the dataset contents are transient. ESQWORK ***MUST*** be assigned in VSAM and preformated using the sample JCL in the Adabas SQL Server sublibrary.

The ESQWORK dataset is dedicated to a single server. It is therefore recommended to assign the correct VSAM ‘SHAREOPTIONS’ to ensure data consistency.

The space requirement for ESQWORK is application-dependent; they depend upon the amount of data which is to be concurrently sorted or processed. A minimum of 20 cylinders is recommended.

Formatting the ESQWORK Dataset

As described above, the ESQWORK dataset must be formatted.

The ESQWORK dataset can be pre-formatted using the temporary file formatting utility **ESQFMT**, which is delivered in the Adabas SQL Server sublibrary.

Sample jobs to define and preformat the ESQWORK file are supplied in the Adabas SQL Server sublibrary in members **J#FORMAT.J** and **J#FORMAS.J**.

Below is an example of the output generated by the temporary file formatting utility ESQFMT:

```
+ESQFMT    : FORMATTING OF ESQWORK STARTED
+ESQFMT    : LEN 04096 TOTAL 01490 WORK 01489 RECV 00000
+ESQFMT    : FORMATTING OF ESQWORK FINISHED SUCCESSFULLY
```

Calculating the Space Requirements

The space requirements of the ESQWORK dataset are dependent upon the requirements of the application:

- the size and number of the records retrieved from the database, and
- the number of client processes, which are to be concurrently supported.

It should be obvious, that these factors are not constant for all applications accessing the server and must therefore be estimated. The values, which are used in the calculation, should be based on either average or maximum values.

The following formula can be used to calculate the space requirements:

```
space_requirements := (nr_records x record_length) x nr_concurrent_users
```

The result of the calculation is in bytes and must be converted into the number of cylinders. This conversion is dependent upon the device-type and is not explained here.

Be aware, that the size of ESQWORK and the size of the Sort work files are interrelated. The Sort work files should be at least as large as, if not larger than, the ESQWORK dataset.

Note:
The space requirements are application-dependent, and may need to be increased based on data volume or the number of concurrent users.

Error Conditions

Error conditions, which are encountered during the Tuple Manager processing, result in a SQL code and/or an system console message and/or job protocol.

The Tuple Manager uses the range of SQL codes: ESQ8502 through ESQ8517**.** The Tuple Manager messages and SQL codes are documented in the *Adabas SQL Server Messages and Codes Manual*.

The Tuple Manager messages issued to the system console have the following format:

```
VO_BM<function> EBP ERROR – sub–code
```

Below are sample SQL code and/or console messages, which would be issued when the size of the ESQWORK dataset is insufficient (too small):

```
ESQ8515, TUPLE MANAGER BUFFER WRITE: VO_WRITE_FILE RETURNED - 2.
+VO_BMWR EBP Allocate Error 4
```

Please notice the subcodes, which indicate a lack of resources:

```
Subcode:    2 – block not allocated
Subcode:    4 – no more blocks to allocate
```

The subcodes are documented in the chapter Adabas SQL Server Mainframe Codes of the *Adabas SQL Server Messages and Codes Manual.*

Sort Processing

The following SQL language constructs invoke sorting:

- SELECT DISTINCT, it may be necessary to sort a result in order to perform duplicate elimination
- DISTINCT functions, for example, COUNT (DISTINCT column), sorting is required for duplicate elimination.
- UNION (without an ALL specification) requires sorting for duplicate elimination.
- GROUP BY, sorting is required to identify groups and to perform aggregation.
- ORDER BY, sorting is required to deliver a sorted result.

One SQL statement may require several of these constructs and thus it may be necessary to sort several times for different purposes.

Adabas SQL Server optimizes the sort processing by performing an internal sort. The runtime system switches from internal to external sort, when the size of the internal sort buffer is exceeded.

The following items should be considered when using the internal sort buffer:

- When executing in LINKED-IN mode, a large internal sort buffer reduces the possibility of switching from internal to external sort buffer.
- When executing in client/server mode, a large internal results in larger client threads, which can increase system paging and the time required for thread roll-in/roll-out.

Internal Sort

The following diagram provides an overview of the internal sort processing.

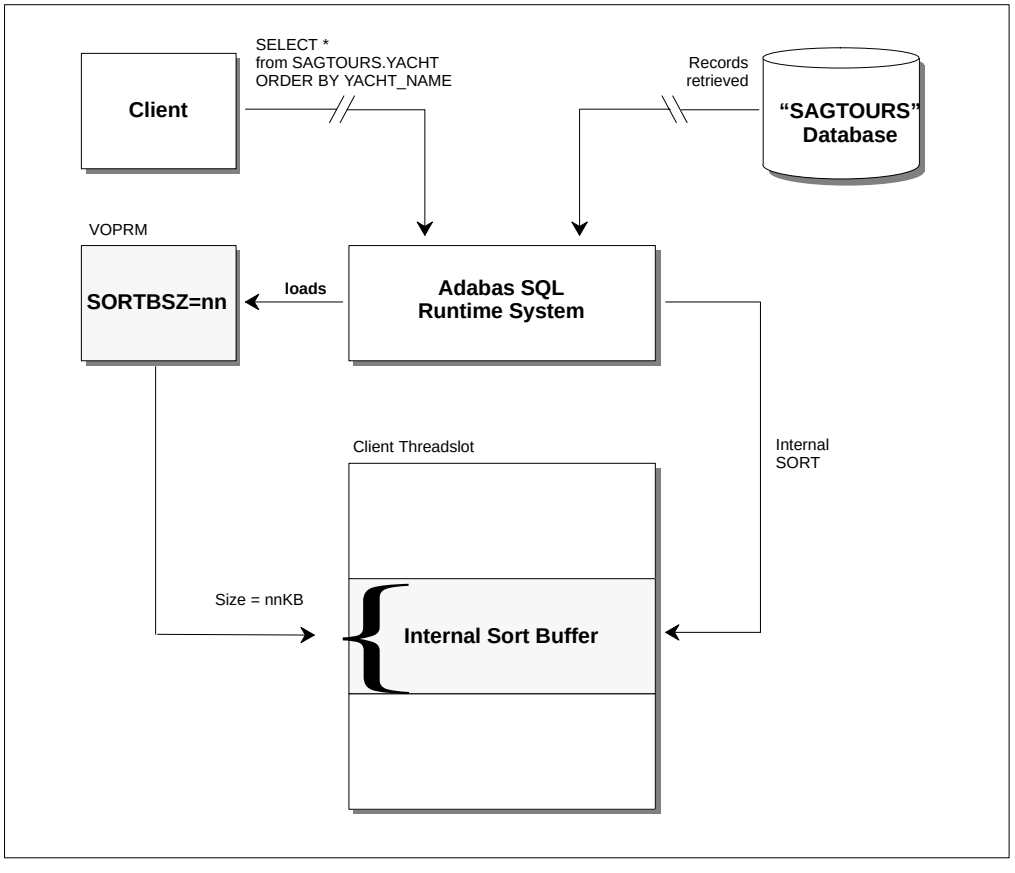


Figure 2-6: Internal SORT

External Sort

The following diagram provides an overview of the external sort processing.

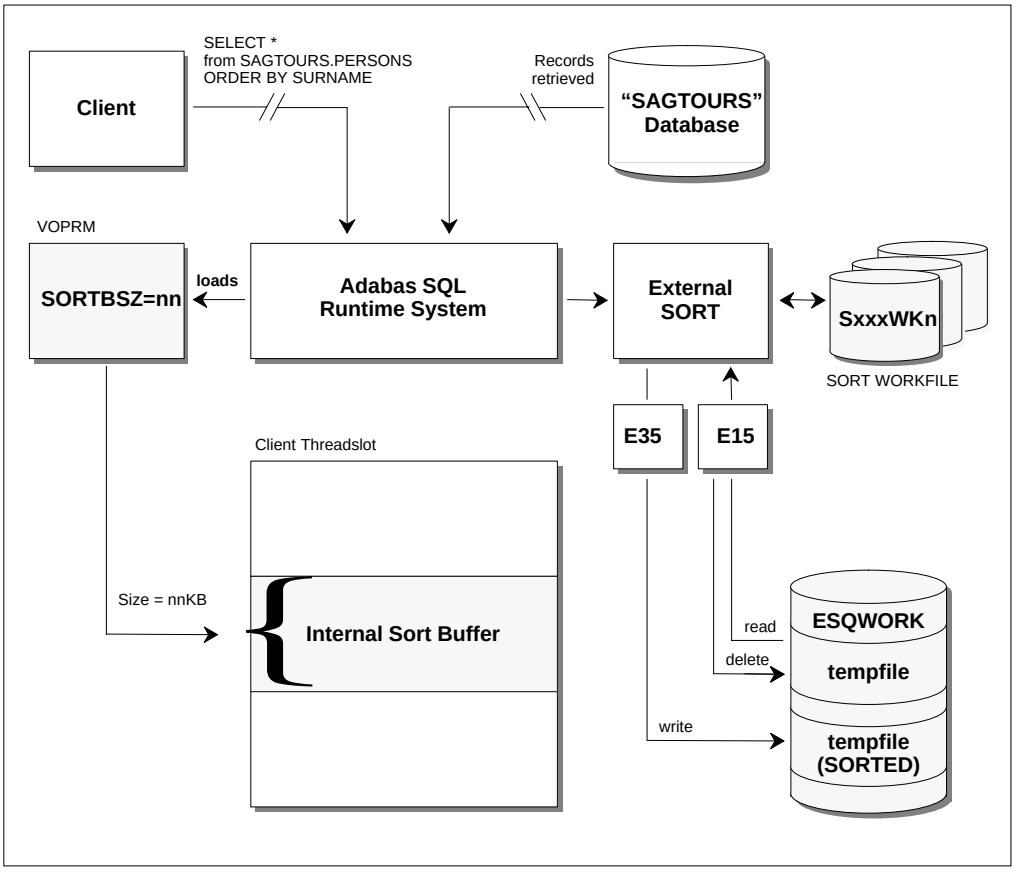


Figure 2-7: External Sort

Sort Buffer Size Setting

The size of the internal sort buffer is set in the module VOPRM. The size is controlled by the SORTBSZ parameter of the VOPRM macro. Values assigned this parameter are in kilobytes and should be a multiple of four (4).

The default size of the internal sort buffer is 40 KB.

For further information about setting this parameter, please refer to the chapter **The VOPRM Module** in this manual.

External Sort Packages

Three external sort packages are supported by Adabas SQL Server in a VSE environment:

- CA-SORT (TM) Version 8.1 and above
- DFSORT (TM)
- SYNCSORT (TM) Version 2.3D and above

Notes:

1. *CA-SORT requires APAR #GS88965 to be applied. Please contact your nearest COMPUTER ASSOCIATES office for this APAR.*
2. *DFSORT requires that space be reserved in the Partition Program Area for its sort modules and work areas. ALL applications executing in LINKED-IN mode and Entire Service Manager (ESG) should therefore have the "// EXEC" statement "SIZE" parameter modified to support this, as follows:*

```
// EXEC ...., SIZE=(AUTO, 128K)
```

3. *Indicate the external sort name and type in VOPRM, as described previously.*

JCL Requirements

The JCL requirements of the external sort packages are product specific, but can be classified as follows:

- SORT Sublibrary
  This sublibrary can either be added to the link list or to the job using JCL definitions.
- SORT Protocol Files
  Some sort packages require additional JCL definitions for the sort protocol and/or diagnostic information.
- SORT Work Files
  The JCL definitions for the sort workfiles are required. Please consult the following section for further information about file assignment, naming conventions and size of sort work files.

SORT Work Files

The sort work files must be assigned in triplets and are identified using the following DD-Names:

```
SxxxWK1
SxxxWK2
SxxxWK3
```

Where the prefix *Sxxx* of the DLBL name, represents the triplet number. The triplets are to be numbered in ascending order, starting with “S001”.

A server executing in LINKED-IN mode requires only a single sort work file triplet.

The number of triplets to be assigned a server executing in client/server mode depends upon the number of concurrent sorting tasks to be supported. It is suggested, that servers executing in client/server mode be assigned more than one triplet of sort work files; doing so influences the overall server throughput. The maximum number of triplets assigned should be less than the number of Entire Service Manager (ESG) threads.

Note:
The maximum number of triplets which can be assigned in client/server mode is ”the number of threads minus one”. The minimum number of triplets which must be assigned in client/server mode is one.

The sort work files should be (approximately) the same size as the temporary work file ESQWORK.

JCL Example

A JCL example is supplied in the member **J#VERSRS.J** in the Adabas SQL Server sublibrary or create a job based upon the example below:

```
* $$ JOB  JNM=ESQSRS,CLASS=0,DISP=D
* $$ LST  CLASS=A,DISP=D
// JOB ESQSRS --- VERIFY EXTERNAL SORT (SINGLE-USER MODE) ---
/*
/* ------------------------------------------------------------- *
/* Execute ESQ installation verification of external sorting
/* (Linked-In)
/* ------------------------------------------------------------- *
/* Issue a complex select to verify external sorting
/* ------------------------------------------------------------- *
```

```
/*
// LIBDEF  PHASE,SEARCH=(SAGLIB.ESQ142S,                              *
               SAGLIB.ESQ142,                                       *
               SAGLIB.ADA621,                                       *
               SORTLIB.SORT),TEMP
// DLBL    ESQPARM,'SAGLIB.ESQ142(P#ESQRUN.D)'
// DLBL    ESQPRIN,'SYSLST'
// DLBL    ESQWORK,'ESQ.SINGLE.ESQWORK',,VSAM,CAT=VSESPUC
// DLBL    SYSIN,'SYSRDR'
// DLBL    SYSOUT,'SYSLST'
// DLBL    SYSPRIN,'SYSLST'
/*
/* ---- ESQ Sortfile datasets ------------------------------------ *
/*
// ASSGN   SYS021,DISK,VOL=ESQ001,SHR
// ASSGN   SYS022,DISK,VOL=ESQ001,SHR
// ASSGN   SYS023,DISK,VOL=ESQ001,SHR
// DLBL    S001WK1,'ESQ.SINGLE.SORTWK1',0,SD
// EXTENT  SYS021,ESQ001,1,0,150,150
// DLBL    S001WK2,'ESQ.SINGLE.SORTWK2',0,SD
// EXTENT  SYS022,ESQ001,1,0,300,150
// DLBL    S001WK3,'ESQ.SINGLE.SORTWK3',0,SD
// EXTENT  SYS023,ESQ001,1,0,450,150
/*
// EXEC    ESQBIF,SIZE=AUTO,PARM='ESQUSER=VR'
ADARUN     DATABASE=36
ADARUN     DEVICE=8381
ADARUN     MODE=MULTI
ADARUN     PROGRAM=USER
ADARUN     SVC=45
/*
SELECT SAILOR_NAME, START_HARBOR, DESTINATION_HARBOR, YACHT_NAME,
       FIRST_NAME_1, FIRST_NAME_2, SURNAME
  FROM SAGTOURS.SAILOR,
       SAGTOURS.CRUISE,
       SAGTOURS.YACHT,
       SAGTOURS.PERSON
 WHERE ID_CRUISE=CRUISE_ID
   AND ID_YACHT=YACHT_ID
   AND ID_SKIPPER=PERSON_ID
 ORDER BY SURNAME, FIRST_NAME_1;
/*
// EXEC    LISTLOG
/&
* $$ EOJ
```

External Security Interface

This section describes the mainframe aspects of the Adabas SQL Server Security topics, in particular the External Security Interface.

Adabas SQL Server Security Concept

The security features implemented in Adabas SQL Server are platform-independent and are in addition to any security mechanisms provided by Adabas itself. The main security features are described below.

A detailed description of the security features provided by Adabas SQL Server are described in the *Adabas SQL Server Programmer's Guide*.

User Concept

In SQL, the concept of the user is captured by an *authorization identifier*. Every operation in an SQL system is validated according to the authorization identifier that attempts to invoke the operation. The implementation of authorization identifiers are left up to the implementation. In Adabas SQL Server, the authorization identifier has been implemented as a *user identifier*, which must be defined to the server using the CREATE USER statement. Adabas SQL Server requires that ALL users be defined to the server.

User identifiers are defined by the administration user DBA and is the only user, which can execute the DDL statements: CREATE and DROP USER.

The administration user DBA is also the only user able to create schema, which inhibits an uncontrolled execution of DDL statements. The user DBA is established as part of the installation, and should be password protected. The user identification DBA cannot be dropped.

A user identifier may be assigned a password using the ALTER USER statement.

Authentication

Users are required to connect to Adabas SQL Server for validation. This can be performed explicitly using the CONNECT statement. If the first SQL statement executed is not a CONNECT statement, an implicit CONNECT is performed using the Adabas SQL client identification as *user identifier without a password.*

Validation is performed on both the user identifier and password identifier, which are passed using the CONNECT statement. Unknown user identifiers or invalid password identifiers will result in the following server response:

```
ESQ6701, CONNECT REJECTED DUE TO AUTHORIZATION FAILURE.
```

- User validation is ALWAYS performed by Adabas SQL Server and cannot be de-activated.
- User validation can also be performed by an external security product. This feature must be activated during the installation and is described below.
- Password validation is only performed, when a password is defined in the catalog.

When the External Security Interface has been activated, the password maintenance should be delegated to the external security package.

Authorization

In the SQL model, the creator of an object is always the owner of the object. The owner has every possible privilege on the object, and can grant some (or all) privileges on that object to other users by using the Data Control Language (DCL) statements, for example using the GRANT and REVOKE. This functionality controls the access operations which are performed using Data Manipulation Language (DML) statements; for example access operations using the SELECT, INSERT, DELETE and UPDATE statements.

In Adabas SQL Server, these security features are not automatically available. These features are selectable and must be activated during the installation of the catalog files (Phase F in the Installation Steps). Once activated, these security features cannot be de-activated without re-installing the catalog.

An Adabas SQL Server which has been installed with the security features described above, is referred to as a Adabas SQL Security Server.

The following special rules apply to access privileges on Data Definition Language (DDL) operations:

- Both schema and user are created by the administration user DBA.
- A schema is owned by a user
- A user can perform any DDL operations in a schema, when the user is the owner.
- A user cannot perform any DDL operations in a schema, when the user is not owner.
- And finally, these rules *cannot* be overridden.

Installing the Server with/without Security Features

This section discusses the consequences of installing Adabas SQL Server both with and without the server security features, which mainly consist of authorization checking.

The GLOBAL SECURITY parameter setting (PPL) during the catalog initialization determines whether the security features are activated or not. This is performed in the installation steps **A.6** and **A.7**.

A word of caution though, the security features once activated cannot be de-activated without re-installing the Adabas SQL Server catalog files.

The following security features are valid for both Adabas SQL Servers generated **with** and **without** security features:

- the USER must be defined using the CREATE USER statement,
- a USER may be assigned a PASSWORD,
- the CONNECT statement with user and password specification is available for connecting to a server for validation.
- the administration user DBA is the only user authorized to create a SCHEMA and to define users,
- the owner of a SCHEMA is the only user authorized to create tables, views, etc.

The consequences of generating Adabas SQL Server **without** the security features are:

- no authorization checking is performed at runtime. Access rights to a specific table, etc. will not be checked,
- the GRANT and REVOKE statements are not available,
- the tables, which hold privilege-related data in the catalog, are created but are empty (for example, the table: TABLE_PRIVILEGES),

The consequences of generating Adabas SQL Server **with** the security features are:

- authorization checking for all SQL requests is performed at runtime,
- the administration user DBA has the same privileges as any other user, with the exception of schema and user definition.

It should be noted that the access privileges, which are maintained using the GRANT and REVOKE statements, are stored in the Adabas SQL Server catalog files. It should be clear, that activating the security features will result in an increased number of calls issued to the database. This extra load may influence the overall server and database performance.

For further information, please refer to the *Adabas SQL Server Programmer's Guide*.

Usage with External Security Products

The security features implemented in Adabas SQL Server are in addition to any security mechanisms provided using external security products.

Authentication

An External Security Interface has been implemented in Adabas SQL Server to perform user and password validation. This interface enables the server to delegate authentication and password maintenance to an external security package.

When activated the External Security Interface is called by Adabas SQL Server during the CONNECT processing to perform user and password validation.

Authorization

An External Security Interface to perform SQL object level security is not available in this version. This functionality is currently only available using Adabas SQL (Security) Server as described above.

Activating the External Security Interface

The External Security Interface is activated during the installation (Phase F) using the VO parameter EXTSEC.

For further detail, please refer to the section VO Parameter Module and Phase F of the installation instructions.

Special Considerations

The following topics require special consideration, when activating the External Security Interface:

- **User ID Maintenance**: The usage of the External Security Interface requires that users be defined to BOTH the external security package and Adabas SQL Server.
- **Password Maintenance**: To simplify the system administration, the password maintenance should be delegated to the external security package. This can be accomplished by removing (or not defining) passwords for Adabas SQL user identifiers. The length of the password identifiers defined in Adabas SQL Server may not exceed the maximum length allowed in the external security package. Passwords are usually limited to a maximum of 8 characters. Passwords, which are longer, will result in an validation error.
- **Administration User DBA**: The administration user DBA is never validated using the External Security Interface. The user identifier DBA is hard-coded and has limited access privileges. The validation is performed by the ADABS SQL Server. It is recommended to protect the user DBA with a password.

- **Explicit CONNECT**: Usage of the external interface *requires* that an CONNECT statement be explicitly coded and executed. An implicit CONNECT will result in a validation error (ESQ6701), and is caused by the fact, that the password is unknown to the Adabas SQL link module and is not (or rather cannot) be supplied.
- **LINKED-IN Mode**: When a client application is executed in LINKED-IN mode, Adabas SQL Server is not authorized to change the login user identification. Therefore, the CONNECT user identifier *must* be the same as the login user identification.

Use with Adabas Security Products

The security features implemented in Adabas SQL Server are in addition to any security mechanisms provided by Adabas itself.

Adabas Data Encryption (Ciphering)

For Adabas SQL Server to access database files, which are protected by Adabas Data Encryption (Ciphering), the cipher-code must be inserted into the ADDITIONS4 field during each call. This can be done by invoking either Adabas SQL User Exit 5 or an Adabas user exit.

Adabas Security (ADASCR)

For Adabas SQL Server to access database files which are protected by Adabas Security (ADASCR), the password must be inserted into the ADDITIONS3 field during each call. This can be done by invoking either Adabas SQL User Exit 5 or an Adabas user exit.

User Exits

Overview

A user exit is a user-written routine that enables the user to participate in the processing performed by 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 following user exits are available for Adabas SQL Server:

User Exit	Use
Client User Exit1	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.

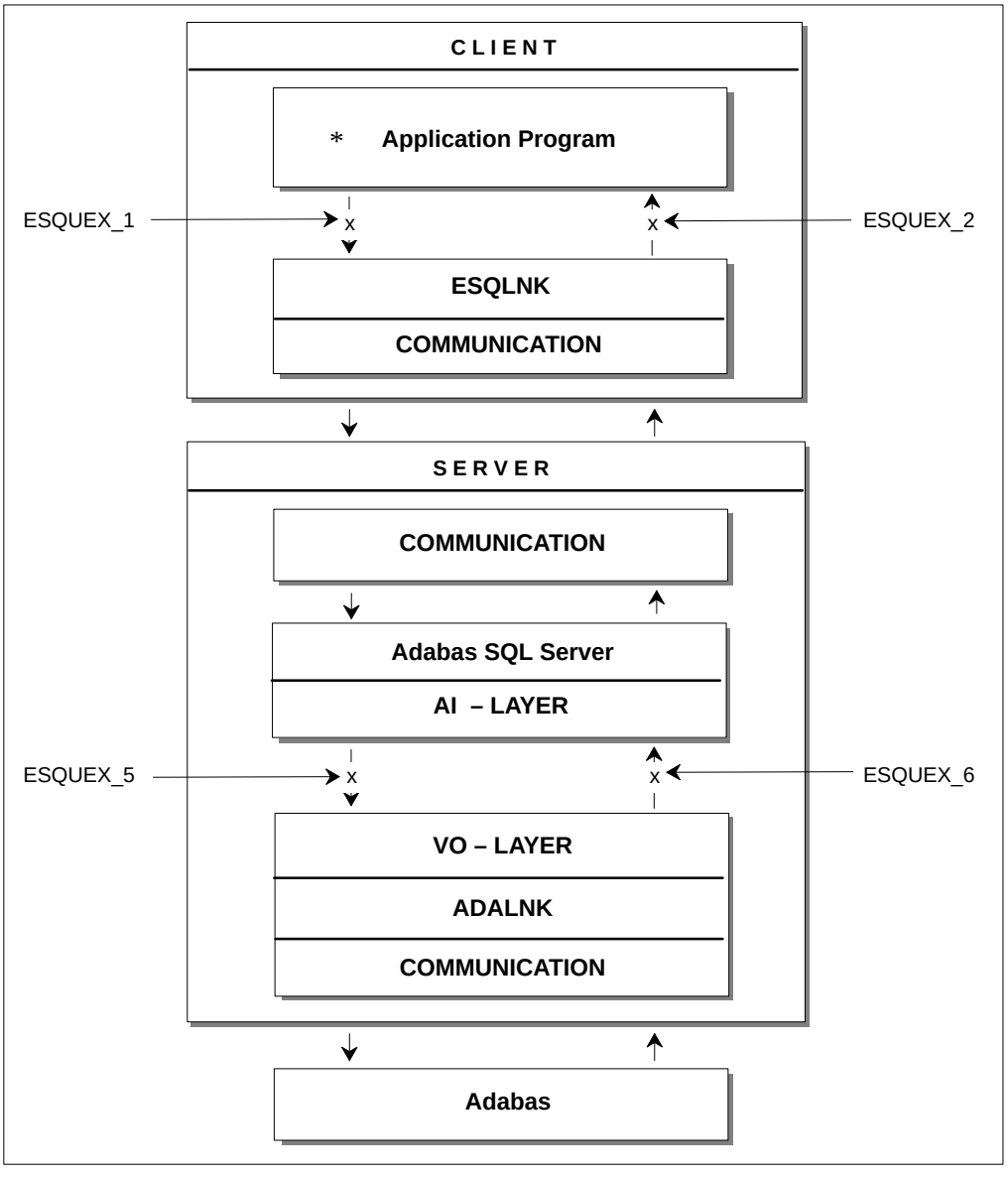


Figure 2-8: Locations of user exits within Adabas SQL Server

Creation and Definition

Make the user exit available to Adabas SQL Server by setting an environment variable to point to the name of the user exit. The environment variable is ”ESQUEX_” followed by the number of the user exit:

```
VOENV     NAME=ESQUEX_1,
          VALUE=my_user_exit_1
```

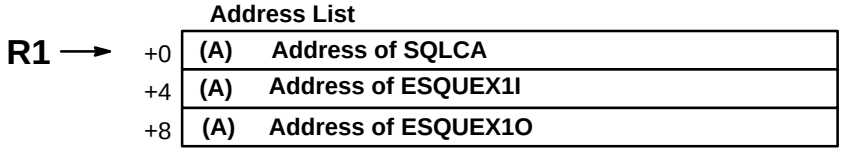
With User Exit Logging, as explained in chapter **Logging Facilities** later in this manual, you are able to log the user exit call and some parameters.

If the Server User Exits 5 and 6 are reentrant they should be defined as RESIDENTPAGE in the Entire Service Manager parameter file SYSPARM as described in the installation step **B.12**.

Note:
All user exits must be able to run AMODE 31 and they are called with standard linkage conventions.

Client User Exit 1 - Description

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. The below mentioned structures are found in the Adabas SQL Server sublibrary.



This file provides the layout for the SQL communication area:

```
.*
SQLCA    DSECT ,
SQLCAID  DC    CL8'SQLCA'              /* eye catcher                */
SQLCABC  DC    A(136)                  /* size of SQLCA in bytes     */
SQLCODE  DS    F                       /* ESQ return code            */
SQLERRML DS    H                       /* length of error message    */
SQLERRMC DS    CL70                    /* error message              */
SQLERRP  DS    CL8                     /* internal error information */
SQLERRD  DS    6F                      /* internal error information */
SQLWARN  DS    CL8                     /* warning flags              */
SQLEXT   DS    CL8                     /* reserved                   */
.*
         MEND  ,
```

This file provides the layout for the user exit 1 input data:

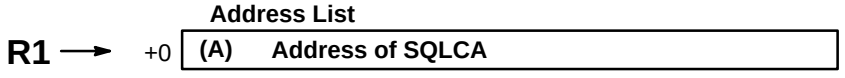
```
.*
ESQUEX1I DSECT ,
SQL_COMMAND DS A
ESQ_CONNECT EQU 3
ESQ_DISCONNECT EQU 8
ESQ_DISCONNECT_ALL EQU 21
ESQ_DISCONNECT_DEFAULT EQU 22
L_NAME   DS    A
L_PWD    DS    A
C_NAME   DS    CL(L_CLIENTS_NAME)      /* EQU defined in ESQACC */
C_PWD    DS    CL(L_CLIENTS_PWD)       /* EQU defined in ESQACC */
         MEND  ,
```

This file provides the layout for the User Exit 1 output data.

```
.*
ESQUEX1O DSECT ,
L_NAME_O DS    A
L_PWD_O  DS    A
C_NAME_O DS    CL(L_CLIENTS_NAME)      /* EQU defined in ESQACC */
C_PWD_O  DS    CL(L_CLIENTS_PWD)       /* EQU defined in ESQACC */
         MEND  ,
```

Client User Exit 2 - Description

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.



The layout for the SQL communication area is shown under User Exit 1.

Client User Exits 1 and 2 - Example

The following example may be used ’as is’ under Batch and Com-plete. If used under Com-plete, it should be defined as RESIDENTPAGE for performance reasons. If it should be used for ESQ-clients running under CICS, it needs to be adapted to CICS conventions.

Assemble/link information: The assembly requires macros from the Adabas SQL Server sublibrary. This routine must be linked with the attribute RENT.

Example 1:

This example of a user exit changes the User ID to ’JOHN’ for a CONNECT statement.

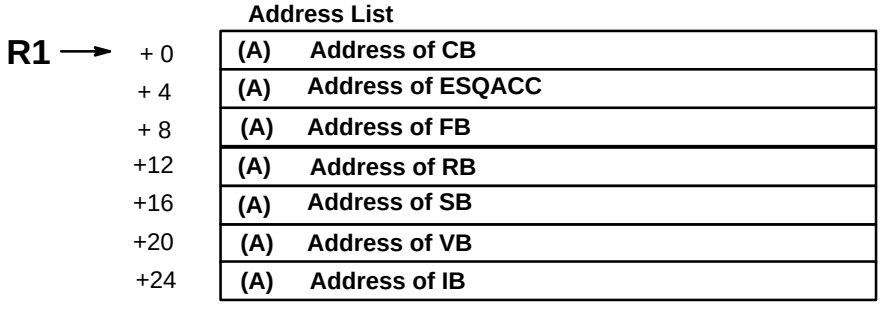
```
*
ESQUEX1  CSECT ,
ESQUEX1  AMODE 31
ESQUEX1  RMODE ANY
         USING ESQUEX1,R15
         B     START
         DC    CL8’ESQUEX1’
         DC    C’&SYSDATE’
         DC    C’&SYSTIME’
START    DS    0H
*
* Entry Housekeeping
*
         STM   R14,R12,12(R13)         save registers
         DROP  R15                     establish ...
         LR    R11,R15
```

```
         USING ESQUEX1,R11             ... base
         L     R9,4(,R1)               R9 := A(input data)
         USING ESQUEX1I,R9
         L     R10,8(,R1)              R10 := A(output data)
         USING ESQUEX1O,R10
*
* Check current statement
*
         CLC   SQL_COMMAND,=A(ESQ_CONNECT)  CONNECT statement?
         BNE   RETURN                  no, go home
*
* Update Output data
*
         MVC   L_NAME_O,=A(L'JOHN)     set length new User ID
         MVC   C_NAME_O(L'JOHN),JOHN   set new User ID
         MVI   C_NAME_O+L'JOHN,X'00'   set end of string
*
* Exit Housekeeping
*
RETURN   DS    0H
         L     R14,12(,R13)            restore ...
         LM    R0,R12,20(R13)          ...registers and ...
         BR    R14                     ......... B Y E
*
* Constants
*
JOHN     DC    C'JOHN'                 everybody's new User ID
         LTORG ,
*
* DSECTs
*
         ESQUEX1I ,                    UEX 1 input data
         ESQUEX1O ,                    UEX 1 output data
*
* Equates
*
         ESQEQU ,                      register EQUates
         ESQACC ,                      ESQ EQUates
*
         END ,
```

Server User Exit 5 - Description

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.

ESQACC can be found in the Adabas SQL Server sublibrary. For information on all other input parameters refer to the *Adabas Command Reference Manual*.



Note:
The NULL value (0) is a valid parameter value and must be handled by the user exit.

This file provides the layout for the Adabas SQL Server Adabas Client Context (ESQACC):

```
L_CLIENTS_NAME EQU 32
L_CLIENTS_NODE EQU 8
L_CLIENTS_PWD  EQU 32
.*
ESQACC   DSECT ,
C#SID    DS    A                      /* session id        */
C#CON    DS    A                      /* current context   */
ESQRTS_CONTEXT EQU -1
C#NODE   DS    CL(L_CLIENTS_NODE)     /* client's node     */
C#NAME   DS    CL(L_CLIENTS_NAME)     /* client's name     */
C#PWD    DS    CL(L_CLIENTS_PWD)      /* hidden: password  */
.*
         MEND  ,
```

Server User Exit 6 - Description

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.

ESQACC can be found in the Adabas SQL Server sublibrary. For information on all other input parameters refer to the *Adabas Command Reference Manual*.

		Address List
R1 →	+ 0	(A) Address of CB
	+ 4	(A) Address of ESQACC
	+ 8	(A) Address of FB
	+12	(A) Address of RB
	+16	(A) Address of SB
	+20	(A) Address of VB
	+24	(A) Address of IB

Note:
The NULL value (0) is a valid parameter value and must be handled by the user exit.

Server User Exits 5 and 6 - Rules

Adabas SQL Server Session Contexts

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). To enable distinction 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 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 Adabas SQL Server, for example, to store a meta program.

If the Server User Exits 5 and 6 are re–entrant they should be defined as RESIDENTPAGE in the Entire Service Manager parameter file SYSPARM as described in the installation step **B.12**.

The Adabas SQL Server session context is provided in the macro ESQACC, and is defined shown in section **Descriptions** earlier in this chapter:

Embedding Server User Exits 5 and 6

The user exit processing has been realized within 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 must return an integer value. The returned value is interpreted within 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 response 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.

User Exit Parameter Values

The NULL value (0) is a valid parameter value and must be handled by the user exit. The processing of a NULL value will result in an address exception.

3

INSTALLING THE ADABAS SQL SERVER SYSTEM

This chapter describes preparing for and installing Adabas SQL Server, Entire Service Manager (ESG) and the Adabas SQL Utilities. The step-by-step description of the installation procedure is presented in five parts:

Phase A:	Installing and verifying the LINKED-IN single-user server.
Phase B:	Installing Entire Service Manager (ESG).
Phase C:	Verifying the client/server (multi-user) installation (and index creation preparation).
Phase D:	Installing Clients (Com-plete, TSO, CICS).
Phase E:	Installing the Adabas SQL Server Utilities.
Phase F:	Activating the Adabas SQL Server Security Features.

Software Configuration

The installation of Adabas SQL Server requires the following products:

- Adabas 5.3 SM2 or higher
- Entire Service Manager Version 1.1 SM 3 or higher
- Entire Net-Work Version 5.4 SM 1 or higher *(Optional)*

The installation of the Adabas SQL Server ADVANCED Interactive Facility requires the following products and product components:

- Natural Version 2.2 SM 7 or higher;
- Natural Editor

Adabas SQL Clients which communicate using ODBC require the following products:

- Adabas ODBC 2.1 SM 1 or higher
- Entire Net-Work Version 2.2 SM 1 or higher (Windows 3.1)
- Entire Net-Work Version 2.3 SM 1 or higher (Windows 95/NT)

Note:
Entire Net-Work *is required for remote client/server communication only.*

Important Last Minute Documentation

The Adabas SQL Server sublibrary contains a file called **$README.$** which holds important additional information.

Upgrading From A Previous Version

Customers upgrading from Adabas SQL Server Version 1.3 must perform a complete installation of Adabas SQL Server Version 1.4.

The installation Phases A, B, C and D are mandatory and must be performed prior to migrating a Version 1.3 Directory to a Version 1.4 Catalog.

The creation of a *new* Version 1.4 Catalog in installation **step A.3** is mandatory.

The installation Phase E is optional and need only be performed, when the Adabas SQL Advanced Interactive Facilities are required.

The installation Phase F is optional and need only be performed, when the External Security Interface is to be activated.

The Migration Utility (ESQMIG) is provided to assist in the conversion of the Version 1.3 SQL Directory to a Version 1.4 Catalog. This utility requires that a COMPLETE installation of Adabas SQL Server Version 1.4 has been previously performed.

Existing User-Exits (ESQUEX*n*) and VO-Parameter modules (VOPRM) are not compatible with the current Adabas SQL Server version and MUST be re-assembled using the macros delivered in the Adabas SQL Server sublibrary.

Existing applications must be relinked to include the newest version of the ESQO*xx* module.

Phase E can be performed whenever the ADVANCED Interactive Facilities are needed.

Note:
Ensure that the latest version of Entire Service Manager is in use.

Partition Storage Requirements

All Adabas SQL Server components and utilities require a **minimum** partition size of 8 megabytes.

Installing a Trace Version

Following the installation steps as described in the following, a non-trace version of Adabas SQL Server will be installed in the ESQvrs sublibrary. A trace version is also available on the installation tape in the ESQvrsT sublibrary. This version may be needed to diagnose software problems in cooperation with Software AG support personnel.

General Conventions for Installation Examples

Example Naming Conventions

In all examples provided in the Installation chapter the following names/values have been assigned to the following parameters. At your site they should be replaced by appropriate, unique, and valid names/values:

SERVER-NAME = AQS9000
VTAM APPLID = AAQV9000
VTAM ACB NAME = AQV9000
ACCESS-ID = 9000

Replacement Variable Symbols

The example job members provided in the Adabas SQL Server sublibrary must be customized to fit the site requirements. These members contain replacement variable symbols (for example: $ESQLIB). These symbols are used to indicate the places where modifications are required.

A list of the replacement variables has been included in **Appendix B** to this manual.

Installation Aid Utility

The installation aid utility (ESQAID) can be used to customize the example members. The use of the utility is optional.

VO Interface to Files using the DLBL Statement

The VO Interface enables Adabas SQL Server to choose between file types that it wishes to access. These are determined by the DLBL File Name Field syntax. Here are some examples:

Librarian File

Format:

```
// DLBL <dlbl>,'<lib>.<sublib>(<member>.<type>)'
```

Example:

```
// DLBL ESQSYS,'ESQ.SYSTEM.LIBRARY',TEMP
     .
     .
// DLBL ESQPARM,'ESQSYS.ESQ142(ESQPARMS.D)'
```

This assigns to the label card ESQPARM the member ESQPARMS of type D from library ESQSYS (which is assigned to ESQ.SYSTEM.LIBRARY by another DLBL card) and sublibrary ESQ142.

Sequential File

Format:

```
// DLBL <dlbl>,'<dlbl1>{,BLKSIZE=<blksize>{,LRECL=<lrecl>{,RECFM=<recfm>}}}'
```

where:
BLKSIZE, LRECL and RECFM are optional and have the defaults 80, 80 and fixed format respectively,

Example:

```
// DLBL ESQTRAC,'SYS008,BLKSIZE=3509,LRECL=121, RECFM=F'
.
.
// DLBL SYS008,'ESQ.WORK',0,SD
// EXTENT SYS008,ESQPU5,1,0,5000,50
```

The first DLBL statement is the VO DLBL which points to another DLBL labelled sequential file on SYS008. It will write fixed format records to that file with a block size of 3509 and a record length of 121 bytes. The target sequential file on the second DLBL is labelled SYS008, has the name ESQ.WORK and is on volume ESAPU5, at start track 5000 of size 50 tracks.

SYSLST or SYSRDR files

Format:

```
// DLBL  <dlbl1>,'<SYSLST/SYSRDR>,{BLKSIZE=<blksize>,{LRECL=<lrecl>,
         {RECFM=<recfm>}}}'
```

where:
BLKSIZE, LRECL and RECFM are optional,

Example:

```
// DLBL ESQPARM,'SYSRDR'
```

will read 80 byte records (default) from SYSRDR until EOF is encountered.

DUMMY Files

Format:

```
// DLBL ESQDMPG,'DUMMY'
```

Files opened for output will be suppressed.
Files opened for input will return EOF (end of file).

Concatenation of Input Files

The VO interface supports the concatenation of up to 8 files using the following syntax:

```
// DLBL ESQSYS,'ESQ.SYSTEM.LIBRARY',TEMP
.
.
// DLBL ESQPARM,'ESQSYS.ESQ142(ESQPARMS.D)'
// DLBL ESQPARM,'ESQSYS.ESQ142S(USER.D)'
// DLBL ESQPARM,'SYSRDR'
.
.
// EXEC ......
.
COMPILER
BEGIN
MODE (DDL ALLOWED , DML ALLOWED)
END
/*
```

Concatenation takes place by repeating the DLBL label name (here ESQPARM). This concatenates 2 librarian files from the library pointed to by ESQSYS (library ESQ.SYSTEM.LIBRARY). These files are sublibrary ESQ142, member ESQPARMS, type D and sublibrary ESQ142S, member USER, type D. The last data concatenated comes from the SYSRDR beginning ”COMPILER” and finishing ”END”.

Installation Steps

Phase A: Installing and Verifying the LINKED-IN Single-User Server

Step A.1: Install Adabas 5.3 or higher (SMA Job P060, Step 5400)

Adabas 5.3 is a prerequisite for Adabas SQL Server.

- If you do not have Adabas 5.3 or higher installed, contact your DBA and proceed with the installation of Adabas as described in the *Adabas Implementation and Maintenance Manual*.
- Carefully read the following and apply corrections:

  The following corrections are prerequisites and must be applied to the Adabas sublibrary:

ZAP Number	*Adabas Version*
AN33105, AN33148,AN34098	5.3.
AN12012, AN12120	6.1.2
AN13060, AN13107, AN13140	6.1.3

  Adabas 6.1 only:

  If compatibility of the L3/L6 command with DESCENDING option between the Adabas 6.1 mainframe implementation and the Adabas open systems implementation is required, apply Adabas ZAP AN13107.

  If using AOSASM version 6.1 the following ZAPs are prerequisites: AN13162 and AN13174.

  If using AOSASM version 6.2 the following ZAP is a prerequisite: AN21008.

- Set the ADARUN Nucleus Parameters:

  Before (re)starting your Adabas database, ensure that the **ADARUN** nucleus parameters are set as required for Adabas SQL Server. The recommended MINIMUM values are:

LBP	=	1000000
LDEUQP	=	150000
LFP	=	30000
LI	=	150000
LP	=	3000
LS	=	70000
LWP	=	220000
NAB	=	64
NH	=	4000
NI	=	1000
NQCID	=	100

  Once you have verified the installation of Adabas 5.3 or higher, continue with the next installation step.

Step A.2: Unload Adabas SQL Server Libraries (SMA Job I008, Step 5402)

The following JCL restores the Adabas SQL Server libraries:

```
* $$ JOB   JNM=ESQREST,.....
* $$ LST   DISP=D,CLASS=A
// JOB     ESQREST
/*
/* This job downloads the datasets, as supplied on the
/* ESQ installation tape, to disk during the
/* installation procedure.
/*
// PAUSE
// ASSGN   SYS006,cuu
// DLBL    SAGLIB,'.......LIBRARY'
// EXTENT  ,vvvvvv,1,0,tttttt,nnnn
/* ============================================================
*  RESTORE ESQvrs.LIBRARY   DISTRIBUTION LIBRARY
*                           SOURCE, OBJ, JCL and PHASE
/* ============================================================
// MTC     REW,SYS006
// MTC     FSF,SYS006,nn
// EXEC    LIBR
   RESTOR  SUB=SAGLIB.ESQvrs : SAGLIB.ESQvrs –
           R=Y TAPE=SYS006
   RESTOR  SUB=SAGLIB.ESQvrsS : SAGLIB.ESQvrsS –
           R=Y TAPE=SYS006
   RESTOR  SUB=SAGLIB.ESQvrsT : SAGLIB.ESQvrsT –
           R=Y TAPE=SYS006
   RESTOR  SUB=SAGLIB.ESQvrsZ : SAGLIB.ESQvrsZ –
           R=Y TAPE=SYS006
/*
/&
* $$ EOJ
```

Where:

vrs	Adabas SQL Server version, release and SM level.
volser	Volume serial number of the disk used for Adabas SQL Server.
tttttt	Start extent of the Adabas SQL Server library to be allocated.
nnnn	Number of tracks to be allocated to the Adabas SQL Server library.
cuu	Unit address of the allocated tape device.
nn	Position of the dataset on the Adabas SQL Server tape.

Also, refer to the Report of Tape Creation in your distribution package for the correct information for the above values. They are subject to change between product releases.

- (Optional). An installation aid (ESQAID) is provided in the Adabas SQL Server sublibrary. This utility can be used to customize the example members in the Adabas SQL Server sublibrary to fit the site requirements.

  A list of the replacement variables used in example members in the Adabas SQL Server sublibrary has been included in **Appendix B**.

Step A.3: Load the Adabas SQL Server Catalog Files into Adabas (SMA Job I050, Steps 5400, 5401, 5402)

- Edit the member **J#LODCAT.J** in the Adabas SQL Server sublibrary. This job uses the datasets ESQvrs.SYS1, ESQvrs.SYS2 and ESQvrs.SYS3 from the Adabas SQL Server installation tape to load the Adabas SQL Server catalog files.

  Note:
  The 3 file numbers chosen for the catalog files MUST be contiguous and within the range of 1 and 255. Adabas SQL Server will not run correctly if this requirement is not met.

- Contact the DBA at your site to find available Adabas file numbers to load.
- Submit **J#LODCAT.J** after making the relevant modifications. The DBA may prefer to use an Adabas load utility job that already exists to load the files instead of updating the **J#LODCAT.J** job.

Step A.4: Load the Adabas SQL Server Error-Messages File into Adabas (SMA Job I050, Step 5403)

- Edit the member **J#LODMSG.J** in the Adabas SQL Server sublibrary. This job uses the dataset ESQvrs.SYS4 from the Adabas SQL Server installation tape to load the Adabas SQL Server error-messages files.
- Contact the DBA at your site to find an available Adabas file number between 1 and 255 to load.
- Submit **J#LODMSG.J** after making the relevant modifications. The DBA may prefer to use an Adabas load utility job that already exists to load the file instead of updating the **J#LODMSG.J** job.

Step A.5: Allocate and Initialize the Adabas SQL Server LINKED–IN Single–User VSAM Work File (ESQWORK)
(SMA Job I008, Step 5407)
(SMA Job I081, Step 5401)

Every Adabas SQL Server requires its own dedicated VSAM work file. During this installation step we will allocate one work file for LINKED–IN single–user mode. It may be necessary dependent on your requirements to allocate multiple work files. The VSAM files MUST be defined with the correct VSAM SHAREOPTIONS to ensure data consistency & unique resources.

Please refer to chapter **Adabas SQL Server Interfaces**, section **The Tuple Manager** for detailed information regarding temporary work files.

- Edit the member **J#FORMAS.J** in the Adabas SQL Server sublibrary to reflect the correct file names and parameter values for your environment.
- Submit **J#FORMAS.J** after making the relevant modifications.

Step A.6: Update the Adabas SQL Server ESQPARMS
(SMA Job I055, Step 5410)
(SMA Job I055, Step 5411)

Example **ESQPARM** parameter files for LINKED-IN single-user and client/server mode are supplied in the Adabas SQL Server sublibrary in members **P#ESQRUN.D** and **P#ESQSRV.D**.

- Tailor the following variables to suit your environment:

$ESQDBID	Database ID of the Adabas database into which the Catalog and Error Messages files were loaded in steps **A.3** and **A.4**.
$ESQCATF	File number of the first Catalog file loaded in step **A.3**.
$ESQERRF	File number of the Error Messages file loaded in step **A.4**.

$ESQFREE1 to $ESQFREE2	Defines the range of file numbers that will be checked when creating a table or cluster. The Adabas system file numbers must not lie within this range, especially not those of the Adabas checkpoint or security files. *Note:* *This statement is optional. The default range is 1 –> 255.*
$ESQSEC	If the server is to be generated with security features, substitute with ON, otherwise OFF. ***Warning**:* *The security features are activated during server generation and cannot be de-activated. For details refer to the previous chapter, section **Installing the Server with/without the Security Features**.*

Step A.7: Load the Catalog Table Descriptions (Metadata)
(SMA Job I055, Step 5412)
(SMA Job I200, Step 5400)

- Edit the member **P#ESQCAT.D** in the Adabas SQL Server sublibrary. This member contains the catalog table description metadata which will be used to prime the Adabas SQL Server catalog. Tailor the following variables to suit your environment. The mentioned files were all loaded during step **A.3**.

$ESQDBID	Database ID of the Adabas database into which the Catalog files were loaded.
$ESQCATF1	File number of the first Catalog file loaded.
$ESQCATF2	File number of the second Catalog file loaded.
$ESQDBNAME	Logical name of the Adabas database to be accessed by Adabas SQL Server. The name can be chosen freely.

- Edit the member **J#CAT.J** in the Adabas SQL Server sublibrary and update it to match the correct dataset names and parameter values for your environment. Also update the ADARUN parameter values to reflect the environment where the Adabas SQL Server catalog and error-messages files were allocated.
- Submit **J#CAT.J** after making the relevant modifications.

Note:
The successful completion of this step is a prerequisite for all following installation steps.

Step A.8: Install the Demo Application SAGTOURS / Verify the LINKED-IN Single-user Mode Server (SMA Job I200, Step 5401)

The purpose of this step is to verify the Adabas SQL Server installation and database communication.

- Edit the member **J#VERESQ.J** in the Adabas SQL Server sublibrary and update it to match the correct dataset names and parameter values for your site.

  The purpose of this job is to verify the Adabas SQL Server installation by creating database objects, and subsequently updating and retrieving data from the tables as follows :
  - Create an example application (SAGTOURS)
  - Populate the tables
  - Select data from the tables.

  The Adabas SQL BASIC Interactive Facility is used to communicate with the server.
- Submit **J#VERESQ.J** after making the relevant modifications.

Application Definitions

The Data Definition Language (DDL) statements required to define the example application SAGTOURS are provided as text members in the Adabas SQL Server sublibrary:

CRUSRESQ.D	Create User ESQ
CRSCTOUR.D	Create Schema SAGTOURS
CRTAB01–05.D	Create Tables

The Data Manipulation Language (DML) statements required to populate and query the application SAGTOURS are provided as text members in the Adabas SQL Server sublibrary:

INSERT01–05.D	Populate Tables
SELECT.D	Table query

Note:
Ensure that the sublibrary containing the VOPRM module is in your LIBDEF search chain.

Note:
*If you received the message **SAGESQ-E-ESQ6701** when running this job (**J#VERESQ.J**), the most likely reason is that step **A.7** has not completed successfully. Should this have been the cause, reinstall the Adabas SQL Server catalog files (step **A.3**). Upon completion, continue with installation step **A.7**.*

Step A.9: Verify the External Sort Interface in LINKED-IN Mode (SMA Job I200, Step 5402)

- Edit the member **J#VERSRS.J** in the Adabas SQL Server sublibrary and update it to match the correct dataset names and parameter values for your environment.
- Submit **J#VERSRS.J** after making the relevant modifications.

  The Data Manipulation Language (DML) statement used to query the application SAGTOURS is provided in the member **SORT.D** in the Adabas SQL Server sublibrary.

Phase B: Installing Entire Service Manager (ESG)

Step
B.1: Unload Entire Service Manager Libraries

The following JCL restores the Entire Service Manager libraries:

```
* $$ JOB  JNM=JCLINST1,
* $$ LST  DISP=D,CLASS=A
// JOB    JCLINST1
/* This job downloads the datasets, as supplied on the
/* ESG installation tape, to disk for use during the
/* installation procedure.
// PAUSE                ..... WAIT FOR TAPE UNIT
// ASSGN   SYS006,cuu
// DLBL    SAGLIB,'.......LIBRARY'
// EXTENT  ,vvvvvv,1,0,tttttt,nnnn
/* ========================================================
*  RESTORE ESGvrs.LIBRARY   DISTRIBUTION LIBRARY
*                           SOURCE, OBJ and PHASE
/* ========================================================
// MTC     REW,SYS006
// MTC     FSF,SYS006,nn
// EXEC    LIBR
   RESTOR  SUB=SAGLIB.ESGvrs : SAGLIB.ESGvrs –
           R=Y TAPE=SYS006
/*
/&
* $$ EOJ
```

Where:

vrs	Entire Service Manager version, release and SM level.
volser	Volume serial number of the disk used for Entire Service Manager.
tttttt	Start extent of the Entire Service Manager library to be allocated.
nnnn	Number of tracks to be allocated to the Entire Service Manager library.
cuu	Unit address of the allocated tape device.
nn	Position of the dataset on the Entire Service Manager tape.

Also, refer to the Report of Tape Creation in your distribution package for the correct information for the above values. They are subject to change between product releases.

Step B.2: Create the Entire Service Manager User Data Library (SMA Job I008, Step 6601)

This library contains installation-dependent modules and user programs. This sublibrary is recommended, since future maintenance to the system may replace the private distribution libraries completely, thereby destroying any user–specific modules in that library.

- Edit the member **JCLINST2.J** in the Entire Service Manager (ESG) sublibrary and update it to match the correct file names and parameter values for your environment.

```
* $$ JOB   JNM=JCLINST2,
* $$ LST   DISP=D,CLASS=A
// JOB     JCLINST2
/*
/* THIS IS THE ESG INSTALLATION JOB2
/*
/* THIS JOB CREATES THE ESG USER DATA LIBRARY
/* THE FOLLOWING CHANGES HAVE TO BE PERFORMED BEFORE RUNNING THIS
/* JOB.
/*
/* 1. INSERT A VALID JECL AND JOB CARD
/* 2. CHANGE THE OUT DSN TO THE REQUIRED LIBRARY.
/*
/* ============================================================
*  CREATE the ESQ User Sublibrary
/* ============================================================
/*
// EXEC    LIBR
   DEFINE  SUB=esg_user_sublib  –
               REPLACE=YES
/*
/&
* $$ EOJ
```

- Submit **JCLINST2.J** after making the relevant modifications.

Step B.3: Install Entire Net-Work for Remote Client/Server Environments/ODBC Access (SMA Job P060, Step 6600)

Note:
If you want to work in a local client/server environment only, Net-Work is not required.

Entire Net-Work 5.4 or higher is a prerequisite for remote client/server use of Adabas SQL Server. If it is not present during server registration, Entire Service Manager will issue messages.

- If you do not have Entire Net-Work 5.4 (or higher) installed contact your System Programmer or DBA and proceed with the installation as described in the *Entire Net-Work 5 Installation and Operations Manual*.

If Entire Net-Work 5.4 (or higher) is installed, proceed with the next installation step.

Step B.4: Assemble and Link the Adabas Link Module (SMA Job I025, Step 6634)

This module is delivered in source format as ADALCO on the Adabas distribution libraries. It must be assembled and linked in the Entire Service Manager (ESG) user sublibrary.

If you are running a version of Adabas lower than 6.2, please make the following source updates:

- Change the statement in the WAIT subroutine,

```
BP       GOTWAIT to
BNZ      GOTWAIT
```

- Add the following lines after the statements in the WAIT subroutine,

```
BNZ      GOTWAIT
SPACE:
*                        new statements
*    SET OS-WAIT FLAG IN ECB
*
      IPK    ,             GET CALLERS KEY
      SPKA   0             SET KEY ZERO
      OI     0(R1),X'80'   SET OS-WAIT FLAG
      SPKA   0(R2)         BACK TO USER'S KEY
*                        end of new statements
```

- If the Adabas SVC is running in AMODE 31, link ADALCO in AMODE 31:

```
// EXEC LNKEDT,PARM='AMODE=31'
```

- If the Adabas SVC is running in AMODE 24, then the ADALCO module must also be linked in AMODE 24 and the following source updates must be applied (in this case, the thread extension cannot be used):

- Change the return at Label EXIT:

```
BR     RE    to   DC  XL2'0B0E'   =  BSM  R0,RE
```

- Change the branch to the WAIT subroutine in the GOTWAIT subroutine,

```
BALR   RE,RF to   DC  XL2'0CEF'   =  BASSM RE,RF
```

Step B.5: Allocate and Initialize the Entire Service Manager Roll File (SMA Job I008, Steps 6615, 6616)

- Edit the member **JCLINST3.J** in the Entire Service Manager (ESG) sublibrary to match the correct file name, location and size of the Entire Service Manager (ESG) roll file. For performance reasons it is recommended to allocate this file as a Non–VSAM file.
- Submit **JCLINST3.J** after making the relevant modifications.

Step B.6: Allocate and Initialize the Entire Service Manager VSAM System Data Containers (SMA Job I008, Step 6620, 6625, 6630, 6635) (SMA Job I009, Step 6610, 6620, 6630, 6640, 6650, 6660, 6670, 6680)

- Edit the member **JCLINST4.J** in the Entire Service Manager (ESG) sublibrary to reflect the appropriate file name, location and size of the ESG system data containers. The initialization step of the job will use IDCAMS REPRO.
- Submit **JCLINST4.J** after making the relevant modifications.
- Edit the member **JCLINSTD.J** in the Entire Service Manager (ESG) sublibrary to upgrade the ESG system data containers previously created and initialized in job **JCLINST.4.J**.
- Submit **JCLINSTD.J** after making the relevant modifications.

Step B.7: Allocate and Initialize the Entire Service Manager SD File (SMA Job I008, Step 6650) (SMA Job I025, Step 6605)

- Edit the member **JCLINST6.J** in the Entire Service Manager (ESG) sublibrary to reflect the correct file name, location and size of the ESG SD file.

  This file MUST be allocated in VSAM.

- Submit **JCLINST6.J** after making the relevant modifications.

Step B.8: Allocate and Initialize the Entire Service Manager Spool File (SMA Job I008, Step 6655) (SMA Job I025, Step 6610)

- Edit the member **JCLINST7.J** in the Entire Service Manager (ESG) sublibrary to reflect the correct file name, location and size of the ESG spool file.

  This file MUST be allocated in VSAM.
- Submit **JCLINST7.J** after making the relevant modifications.

Step B.9: Allocate and Initialize the Entire Service Manager Dump File (SMA Job I008, Step 6660) (SMA Job I025, Step 6615)

- Edit the member **JCLINST8.J** in the Entire Service Manager (ESG) sublibrary to reflect the correct file name, location and size of the ESG dump file.

  This file MUST be allocated in VSAM.
- Submit **JCLINST8.J** after making the relevant modifications.

Step B.10: Define Entire Service Manager to System History (MSHP)

Entire Service Manager MUST be defined to system history. The Maintain System History Program (MSHP) is used to maintain corrections which may have to be applied to Entire Service Manager (ESG).

- Edit the member **JCLINSTA.J** in the Entire Service Manager sublibrary to match the correct file names and values for your environment.
- Submit **JCLINSTA.J** after making the relevant modifications.

Step B.11: Set the Adabas SQL Server Parameters (VOPRM) (SMA Job I055, Steps 5400, 5403)

In this step, you will establish parameter values that are critical to a successful use of Adabas SQL Server in a client/server environment. The following example parameter settings are possible:

```
VOENV    NAME=ESQSRV,VALUE=AQS9000
ESQRTAB SERVER=AQS9000,ID=IBM1,NUMBER=9000
```

Note:
The VOENV macro parameter NAME=ESQSRV is mandatory and indicates the environment variable name which must under no circumstances be changed.

Interacting Effects of Parameter Settings

VALUE=	in the VOPRM/VOENV macro parameter must be equal to the SERVER= value in the VOPRM ESQRTAB parameter. This same value must also be supplied as the first parameter in the SYSPARM SERVER parameter, as shown above.
NUMBER=	in the VOPRM/ESQRTAB macro parameter must be equal to the SYSPARM ACCESS-ID parameter.
ID=	in the VOPRM/ESQRTAB macro parameter must be set to the node name specified in the parameter dataset in your Entire Net-Work startup job.

- Create the VOPRM module for Adabas SQL Server by using the example job **J#VOPRM.J** in the Adabas SQL Server sublibrary. The associated **VOPRM** parameters used as input to this job can be found in member **P#VOPRM.A** in the Adabas SQL Server sublibrary.
- Ensure that your **VOENV** and **ESQRTAB** parameters are set based on the example above.
- Submit **J#VOPRM.J** after making the relevant modifications.

Note:
Ensure that the sublibrary into which the VOPRM.PHASE is link–edited is included in the LIBDEF search chain of your Entire Service Manager startup job.

Step B.12: Set the Entire Service Manager Parameters (SYSPARM) (SMA Job I025, Step 6650)

In this step, you will be establishing parameter values that are critical to your successful use of Adabas SQL Server in a client/server environment. The following example parameter settings are possible:

```
ACCESS–ID=9000
SERVER=(AQS9000,TSVRADMN,0000,C=69,ESQINI,ESQDRV,ESQSRV)
```

- Edit the member **ESG#PROC.J** in the Adabas SQL Server sublibrary. This member will be modified again in Step **B.17** prior to starting Entire Service Manager. In the SYSPARM section make the following changes to reflect your environment:

VTAMAPPL	Must be changed to the ACB name which you will be defining in the PRD2.CONFIG dataset in step **B.13**.
SECSYS	If your installation runs with a security system such as ACF2**,** TOP SECRET or RACF, you may check the SECSYS sysparm, as well as the definitions for the security system required for Entire Service Manager (ESG). You should set this value to NO until you are ready to activate the External Security Interface as described in step **F.3**.
ACCESS-SVC	Must be set to the SVC number of your router SVC.
ACCESS-ID	Set to the Access node id you have chosen for your site.
ADASVC5	Used to define the default SVC with which to access the database.
APPLYMOD	Sets Entire Service Manager runtime modifications. APPLYMOD=19 APPLYMOD=29 APPLYMOD=52 APPLYMOD=57
CPUTIME	Should be at least 60 cpu seconds for each thread in which the Adabas SQL Server server can run.
RESIDENTPAGE=	Loads the resident page modules required RESIDENTPAGE=AOSASM RESIDENTPAGE=CSCI RESIDENTPAGE=ESQSRV
THREADS	The ESQ threads must be 1024K. A single small thread (256K) should be defined for utilities.

SERVER=(*server-name*,TSVRADMN,0000,C=5,ESQINI,ESQDRV,ESQSRV)
Defines an Adabas SQL server with the name *server-name*.

Note:
It is important that you use the Adabas SQL Server SYSPARM for this step and NOT the example SYSPARM supplied with Entire Service Manager or an existing Com-plete SYSPARM.

Step B.13: Define the Entire Service Manager VTAM ACB (SMA Job I025, Step 6630)

- Define a unique ACB to VTAM in the PRD2.CONFIG dataset to enable the Entire Service Manager (ESG) VTAM interface. The following example definition generates an ACB called ”AAQV9000” with the necessary privileges for ESG:

```
*
AAQV9000 VBUILD TYPE=APPL
*
* APPL DEFINITION STATEMENTS FOR ADABAS SQL SERVER
*
AQV9000  APPL  EAS=5,             EST NO CONCURRENT SESSIONS
         ACBNAME=AQV9000,         APPLID FOR ACB
         AUTH=(ACQ,PASS),         COMPLETE ACQUIRE & PASS TMLS
         PARSESS=YES              PARALLEL SESSIONS FOR NETPASS
                                   AND REQUEST BLOCKED INPUT
*
```

- Enter the following example command on your system console to make the application known to your VTAM**:**

```
V NET,ID=____,INACT,FORCE
V NET,ID=____,ACT
```

- Refer to the appropriate VTAM documentation for more information.

Step B.14: Install the Entire Service Manager Batch Interface (SMA Job I025, Step 6632)

Batch applications that use Entire Service Manager (ESG) services communicate with the target ESG using ACCESS. This communication is based on an Adabas SVC (ACCESS SVC) and a DBID (ACCESS NODE ID).

The Entire Service Manager (ESG) batch interface module loads a module with name ACSTAB and searches this module for an entry name BATCH.

- Modify the example ACSTAB module in the Entire Service Manager (ESG) sublibrary to reflect the system requirements for ACCESS SVC and ACCESS NODE ID.
- Edit the member **JCLINSTF.J** in the Entire Service Manager (ESG) sublibrary to assemble and linkedit the module ACSTAB into the ESG user sublibrary.
- Submit **JCLINSTF.J** after making the relevant modifications.

Note:
The DBID chosen may be greater than 255 and that value must match the SYSPARM value for ACCESS NODE ID. The SVC value must match the SYSPARM value for ACCESS SVC.

Note:
Ignore any warning messages issued by the linker in regard to AMODE and RMODE conflicts.

Step B.15: Initialize the Entire Service Manager System Intercept (SMA Job I025, Step 6655)

In VSE environments, communication between the Entire Service Manager nucleus and the user program is handled by the Entire Service Manager SVC (Supervisor Call — normally 200).

The program ESGSIP is used to dynamically install the SVC without an IPL. ESGSIP requires a prior SET SDL for the SVC, and therefore must run in the BG partition.

You only need one system intercept module ESGSIP for all Entire Service Managers (ESGs) running in your system.

Executing the ESGSIP Program

```
* $$ JOB   JNM=ESGSIP,DISP=D,CLASS=?
* $$ LST   DISP=D,CLASS=A
// JOB     ESGSIP   initialize ESG system adapter
// ASSGN   SYSLST,IGN
// DLBL    SAGLIB,'.....LIBRARY'
// EXTENT  ,vvvvvv
// LIBDEF  PHASE,SEARCH=SAGLIB.ESGvrs,TEMP
SET     SDL
CRSVATBL, SVA
/*
// UPSI    00000000
// EXEC    COMSIP
/*
/&
* $$ EOJ
```

Insert the following (or the equivalent) in the ASI BG JCL procedure immediately before the START of the POWER partition so that the Entire Service Manager SVC will be installed automatically during each IPL.

```
// DLBL    SAGLIB,'.....LIBRARY'
// EXTENT  SYS010,vvvvvv
// ASSGN   SYS010,DISK, VOL=vvvvvv,SHR
// LIBDEF  PHASE,SEARCH=SAGLIB.ESGvrs,TEMP
SET SDL
CRSVATBL,SVA
/*
// UPSI    00000000
// EXEC    COMSIP
```

An example JCL to execute the ESGSIP program can be found in the Entire Service Manager (ESG) sublibrary in member **ESGSIP.J**.

Step B.16: Allocate and Initialize the Adabas SQL Server Client/Server Multi–User VSAM Work File (ESQWORK)
(SMA Job I008, Step 5406)
(SMA Job I081, Step 5400)

Every Adabas SQL Server requires its own dedicated VSAM work file. During this installation step we will allocate one work file for client/server multi–user mode. It may be necessary dependent on your requirements to allocate multiple work files. The VSAM files MUST be defined with the correct VSAM SHAREOPTIONS to ensure data consistency & unique resources.

Please refer to chapter **Adabas SQL Server Interfaces**, section: **Tuple Manager Interface** for detailed information regarding temporary work files.

- Edit the member **J#FORMAT.J** in the Adabas SQL Server sublibrary to reflect the correct file names and parameter values for your environment.
- Submit **J#FORMAT.J** after making the relevant modifications.

Step B.17: Prepare the Entire Service Manager Startup Procedure
(SMA Job I025, Step 6650)

Entire Service Manager (ESG) can be run in both STATIC and DYNAMIC partitions. To run Entire Service Manager in a DYNAMIC partition you will require Adabas with ESA features for ACCESS support.

- Edit the procedure member **ESG#PROC.J** in the Adabas SQL Server sublibrary to prepare the startup procedure for the server.

List of items to consider when updating this procedure:

- The typical Adabas SQL Server requires at least 12 megabytes to operate.
- The LIBDEF search chain should include the following sublibraries:

Entire Service Manager User sublibrary	(Step **B.2**)
Entire Service Manager sublibrary	(Step **B.1**)
Adabas SQL Server sublibrary	(Step **A.2**)
Adabas sublibrary	
Sort sublibrary	(site–dependent)

- The ROLL1 DLBL card should point to the Entire Service Manager (ESG) ROLL file created in Step **B.5**.

- The COMSYS1–4 DLBL cards should point to the Entire Service Manager (ESG) VSAM system data container created in Step **B.6**.
- The COMSD DLBL card should point to the Entire Service Manager (ESG) SD file created in Step **B.7**.
- The COMSPL DLBL card should point to the Entire Service Manager (ESG) SPOOL file created in Step **B.8**.
- The COMDMP DLBL card should point to the Entire Service Manager (ESG) DUMP file created in Step **B.9**.
- The ESQPARM DLBL card should point to the Adabas SQL Server parameters created in Step **A.6**. (**P#ESQSRV.D**).
- The ESQWORK DLBL card should point to the Adabas SQL Server multi-user work file (ESQWORK) created in Step **B.16**.
- The sort work files are assigned in triplets using the DLBL names S*nnn*WK1, S*nnn*WK2 and S*nnn*WK3,
  where:
  nnn represents the triplet's number.

  Refer to the chapter **Adabas SQL Server Interfaces**, section **Tuple Manager Interface** for detailed information regarding external sort requirements.
- The SYSPARM section should contain the Entire Service Manager (ESG) system parameters modified in Step **B.12**.

Step B.18: Start Entire Service Manager (SMA Job P060, Step 6600)

- Before starting Entire Service Manager, the CA-DYNAM adapter must be disabled for the Entire Service Manager (ESG) partition. After an Entire Service Manager shutdown, this adapter must be re-enabled.
- Make sure that your Entire Net-Work is active before starting Entire Service Manager (ESG). ESG will attempt to register with the network. If the network is not active this will fail and client server access will not be possible. This is indicated by a register failed message in the Entire Service Manager (ESG) SYSLOG during initialization. If Entire Net-Work is to be activated after Entire Service Manager (ESG) server initialization, enter the following console command:

  ```
  nnn SERV server-name,REG
  ```

Where:
nnn is the "AWAITING REPLY" message number for the Entire Service Manager (ESG) server partition,
server-name is the ESQ server name given as the SERVER=(parameter of the ESG SYSPARM cards.

- Start the ESG server by submitting the job created in Step **B.17**.

Monitor the startup messages closely for any signs of problems. When initialization is complete, a message to that effect is written to the operator console. Messages will also indicate that Adabas SQL Server, VTAM, and ACCESS interfaces have been initialized.

Step B.19: Stopping Entire Service Manager

Warning:
Do not perform this step at this point in the installation. It has been included here for informational purposes only and will be referenced later.

Entire Service Manager may be terminated using the command EOJ, for example:

```
nnn EOJ
```

Where:
nnn is the "AWAITING REPLY" message number for the Entire Service Manager (ESG) server partition,

The command immediately terminates outstanding terminal I/O requests and performs a logical shutdown of Entire Service Manager (ESG).

For further information refer to the *Entire Service Manager User's Guide and Utilities*, chapter: **Operator Commands**.

Phase C: Verifying the Client/Server (Multi-User) Installation

Verification of the client/server environment is performed in two steps:

- In the first step the basic verification is performed by communicating with Adabas SQL Server in client/server mode and retrieving from tables created in step **A.8**.
- In the second step the verification of the Adabas SQL Precompiler and the Adabas SQL client/server environment in Entire Service Manager (ESG) is performed. Verification routines are provided for COBOL**,** C and PL/1**.**

Step C.1: Verify the Client/Server Installation (SMA Job I200 , Step 5403)

Note:
*The following steps require the client/server environment to be active. If the server is not active at this time, start the Entire Service Manager (ESG) server as described in step **B.18** .*

- Edit the member **J#VERESG.J** job in the Adabas SQL Server sublibrary and update it to match the correct dataset names and parameter values for your site.
- Submit **J#VERESG.J** after making the relevant modifications.

This job will verify the multi-user installation by retrieving data from the tables created in step **A.8**.

Note:
*Ensure that the sublibrary containing the Client/Server Multi-User **VOPRM** module is in your LIBDEF search chain.*

Step C.2: Verify the Adabas SQL Server Client Installation (SMA Job I055, Step 5413) (SMA Job I200, Step 5404)

- Edit the member **J#VERCOB.J** in the Adabas SQL Server sublibrary to reflect the correct dataset names and parameter values for your environment.
- Ensure that the following conditions apply:
  - The COBOL verification program **COB2PGM.B** in the Adabas SQL Server sublibrary compiled in this step is tailored for a COBOL II environment. If you want to compile the program in a COBOL I environment use the verification program **COB1PGM.B**. Additionally, the PRECOMPILER COBOL LANGUAGE SETTINGS parameter in member **P#ESQPC.D** in the Adabas SQL Server sublibrary must be changed from COBOL II = ON to COBOL II = OFF.
  - The ADARUN statements in the first step (PRE) point to the DBID and SVC of the database where the Adabas SQL Server catalog files were loaded in step **A.3**.
  - Ensure that COBOL parameters QUOTE/APOST match the Adabas SQL parameter PRECOMPILER COBOL LANGUAGE SETTINGS DOUBLE/SINGLE QUOTES. Double quotes is the default setting and is used by the verification program.
  - The ADARUN statements in the last step (RUN) point to the SVC defined by the ACCESS-SVC parameter in step **B.12**.
  - In the LIBDEF search chain of the first and last step of the job, ensure that you have included the sublibrary where your **VOPRM** module from step **B.11** resides.

  The files pointed to by the ESQPARM DLBL statements in the first and last step need to be updated with the database number and file numbers of the Adabas SQL catalog and error-messages files that were loaded in steps **A.3** and **A.4**. See **Appendix A** for additional information regarding GLOBAL**,** COMPILER and PRECOMPILER syntax.
- Submit **J#VERCOB.J** after making the relevant modifications. The verification consists of the following:
  - Precompiling a COBOL program
  - Compiling the precompiled source
  - Linking and executing the compiled program

  The compiled program will display the results of a selection of rows from the table **SAGTOURS.CRUISE** where the column **START_HARBOR** has the value "**PANAMA**".

Step C.3: Prepare for CREATE INDEX in Client/Server Mode (SMA Job P060, Step 5400)

The general information about the CREATE INDEX statement is to be found in the *Adabas SQL Server Reference Manual*. In addition to that information, it is important to know that in an VSE environment this statement cannot be executed online but must be submitted using a batch job in Client-Server Multi-User mode, which performs the index creation on the specified column(s).

Note:
*The following procedure MUST be performed for **ALL** databases that are defined by the CREATE DATABASE SQL statement as described in the **Adabas SQL Server Reference Manual**.*

- Edit the member **INV00000.J** in the Adabas SQL Server sublibrary to reflect the correct dataset names and parameter values for your environment.

Warning:
*Do NOT remove or change the order of the Adabas ADAINV statements in the JCL. The **#ADAINV#** statement MUST be positioned as the first parameter in the ADAINV input statements and MUST start in column 1. Adabas SQL Server will dynamically replace this card at execution time.*

Note:
*Ensure that the sublibrary containing the ACSTAB module created in Step **B.14** is included in the LIBDEF search chain.*

- Save the modified member as **INVddddd.J** into the Adabas SQL Server sublibrary, where *ddddd* is the 5–digit DBID.
- Modify the Entire Service Manager startup JCL (**ESG#PROC.J**) to include a DLBL entry for the member created above as follows:

```
//  DLBL ESddddd, 'ESQLIB(INVddddd.J)'
```

Where:
ddddd is the 5–digit DBID

Step C.4: Verify the CREATE INDEX and DROP INDEX Functionality (SMA Job I200, Step 5405, 5406)

- Edit the member **J#CRINX.J** in the Adabas SQL Server sublibrary to reflect your environment.
- Submit **J#CRINX.J** after making the relevant modifications. Upon submission of the job, the ESG server will display the following system message:

```
Submitting ADABAS Utility Job for User <userid>
```

- Check the status of the submitted job upon completion.
- After successful index creation, drop the index and table previously created by **J#CRINX.J**.
- Edit the member **J#DRINX.J** in the Adabas SQL Server sublibrary to reflect your environment.
- Submit **J#DRINX.J** after making the relevant modifications.

Step C.5: Verify the External Sort Interface in Client/Server Mode (SMA JobI200 , Step 5407)

- Edit the member **J#VERSRT.J** in the Adabas SQL Server sublibrary and update it to match the correct dataset names and parameter values for your environment.
- Submit **J#VERSRT.J** after making the relevant modifications.

  The Data Manipulation Language (DML) statement used to query the application SAGTOURS is provided in the member **SORT.D** in the Adabas SQL Server sublibrary.

Phase D: Installing Clients

Step D.1: Installation of Clients under Com-plete (SMA Job P060, Step 5400)

This section discusses the installation of Adabas SQL clients in a Com-plete environment.

The following diagram provides an overview of the Adabas SQL client modules in a Com-plete environment. The individual modules and system requirements are described below.

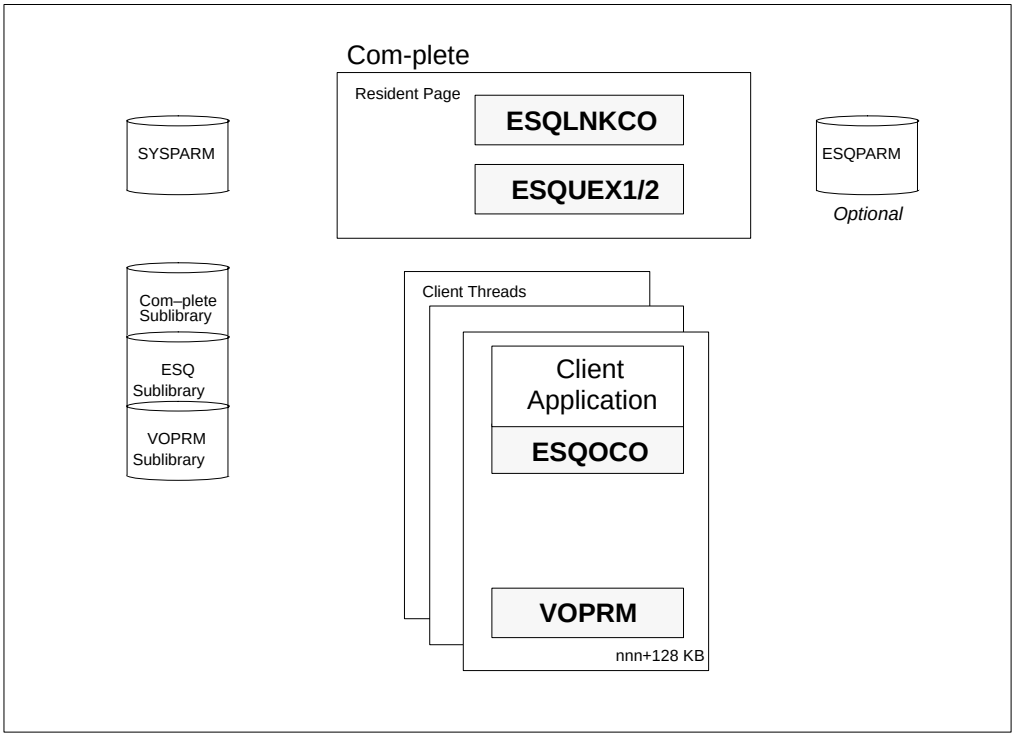


Figure 3-1: Adabas SQL Client under Com-plete

Adabas SQL requires that the module ESQOCO be linked to a client application. This module provides an interface to the Adabas SQL link module ESQLNKCO, which must be defined as a RESIDENTPAGE in the Com-plete parameter file (SYSPARM).

For those installing the Adabas SQL Utilities, please refer to Phase E of the installation steps for further information on linking Natural.

The Adabas SQL link module requires approximately 128 KB in the user thread. This area contains the client-context and MUST be located below 16MB.

The Adabas SQL link module loads the VO parameter module (VOPRM), which contains the server routing file/table (ESQRTAB). This module is required and is used to determine the location of the (default) server. Please refer to the section discussing the VO parameter module for further information.

Installation Steps

Perform the following installation steps:

- Link the module ESQOCO to your application program.
- Enlarge the thread size (below 16MB) to handle the additional space requirements of the Adabas SQL link module (ca. 128KB).
- Edit your Com-plete startup job. Modify the LIBDEF PHASE search chain to include the Adabas SQL Server sublibrary and the sublibrary containing the **VOPRM** module created in step **B.11**.
- Add the following line to your Com-plete parameter file (SYSPARM):

```
RESIDENTPAGE=ESQLNKCO
```

Step D.2: Installation of Clients under CICS (SMA JobP060 , Step 5400)

This section discusses the installation of Adabas SQL clients in a CICS environment.

The following diagram provides an overview of the Adabas SQL client components in a CICS environment. The individual modules and system requirements are described below.

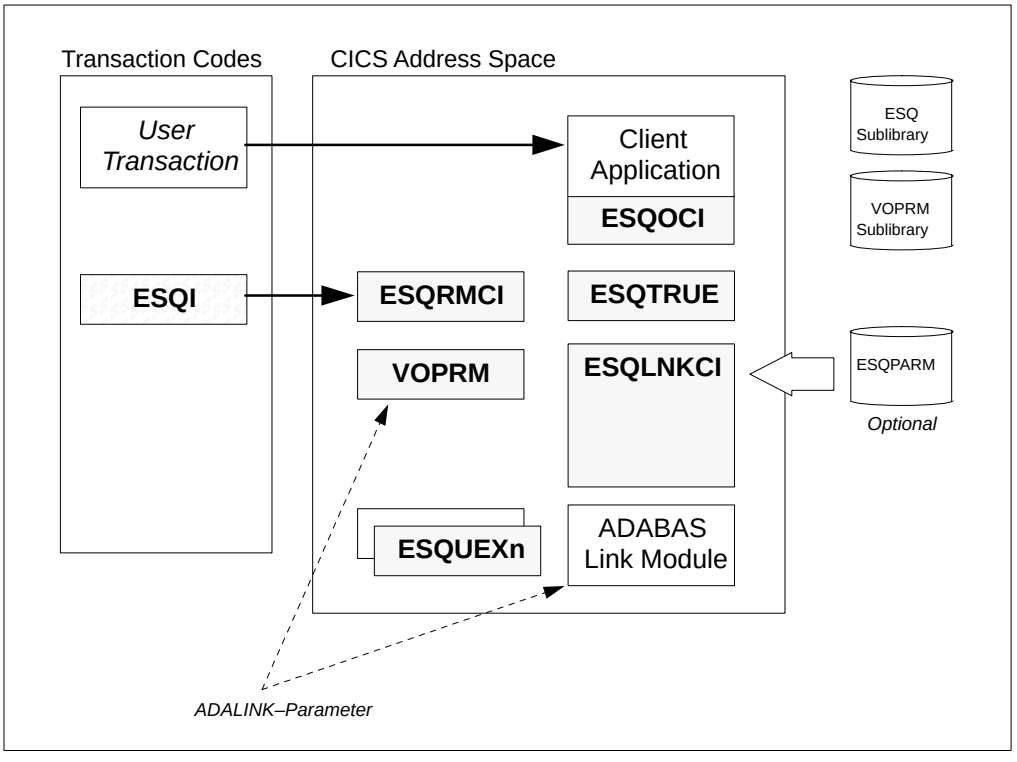


Figure 3-2: Adabas SQL Client under CICS

CICS Interface Modules

The following members from the Adabas SQL Server sublibrary should be made available to the CICS startup job:

ESQLNKCI
ESQRMCI
ESQTRUE
VOPRM

CICS Definitions

- Update the **CICS resource definitions** using either RDO (CEDA) or the sample table updates provided in the Adabas SQL Server sublibrary in the following members:

  ESQDCT.D
  ESQPCT.D
  ESQPLT.D
  ESQPPT.D

  Note:
  User-exits must be installed as a CICS program. This includes creating PPT entries and link-editing the user-exits with the CICS EXEC interface stub.

- Updates for the **DFHPPT** List:

```
DFHPPT TYPE=ENTRY,PROGRAM=ESQRMCI
DFHPPT TYPE=ENTRY,PROGRAM=ESQLNKCI
DFHPPT TYPE=ENTRY,PROGRAM=ESQOCI
DFHPPT TYPE=ENTRY,PROGRAM=ESQTRUE
DFHPPT TYPE=ENTRY,PROGRAM=VOPRM
```

  When using CICS Storage Protection, the modules ESQLNKCI, ESQRMCI and ESQTRUE must be defined with: EXECKEY(CICS).

- Updates for the **DFHDCT** List:

```
DFHDCT TYPE=SDSCI,DSCNAME=ESQPARM,RECSIZE=80,BLKSIZE=80,
               RECFORM=FIXBLK,TYPEFLE=INPUT
DFHDCT TYPE=SDSCI,DSCNAME=ESQPRIN,RECSIZE=250,BLKSIZE=4008,
               RECFORM=FIXBLK,TYPEFLE=OUTPUT
DFHDCT TYPE=SDSCI,DSCNAME=ESQTRAC,RECSIZE=132,BLKSIZE=3968,
               RECFORM=FIXBLK,TYPEFLE=OUTPUT
DFHDCT TYPE=EXTRA,DESTID=ESQP,DSCNAME=ESQPARM,
               OPEN=DEFERRED
DFHDCT TYPE=EXTRA,DESTID=ESQN,DSCNAME=ESQPRIN,
               OPEN=DEFERRED
DFHDCT TYPE=EXTRA,DESTID=ESQT,DSCNAME=ESQTRAC,
               OPEN=DEFERRED
```

- Updates for the **DFHPLT** List:

  Before the Adabas SQL Server interface to the ESG Server can run, it must be initialized. This will be done automatically by CICS if the following entry is added to the DFHPLT:

  ```
  DFHPLT PROGRAM=ESQRMCI
  ```

  If you choose not to add this entry to the **PLT**, the **ESQI** transaction must be executed before starting an application which calls Adabas SQL Server.

- Updates for the **DFHPCT** List:

  The following entry allows you to enable the task-related user exit and initialize the Adabas SQL Server environment under CICS.

  ```
  DFHPCT TYPE=ENTRY,TRANSID=ESQI,PROGRAM=ESQRMCI,TWASIZE=32
  ```

  The following message will be issued on execution of the ESQI transaction:

  ```
  ESQ001I ESQ CICS ENVIRONMENT ESTABLISHED
  ```

- It may be necessary to refresh modules without taking the CICS system down. An example of this would be the need to apply a zap to one of the Adabas SQL Server interface modules. Task-Related user exits used by Adabas SQL Server may still be active and must be deactivated prior to refreshing modules as follows:
  - Execute the “ESQI –D” transaction to deactivate the Task-Related User Exit
  - The following message will be issued if the exit has been successfully deactivated:

    ```
    ESQ001D ESQ CICS ENVIRONMENT DISABLED
    ```
  - Perform a CEDA “NEWCOPY” on the Adabas SQL Server module that was updated.
  - Run the ESQI transaction to re-initialize the SQL Server

    ```
    ESQ001I ESQ CICS ENVIRONMENT ESTABLISHED
    ```
  - Re-run the Adabas SQL Server application program

VO Parameter

- Create a Client VOPRM module for Adabas SQL Server in your CICS environment by using the example job **J#VOPRM.J** in the Adabas SQL Server sublibrary. The associated **VOPRM** parameters used as input to this job can be found in member **P#VOPRM.D** in the Adabas SQL Server sublibrary. The **VOPRM** that you create MUST be placed into a sublibrary that is in the LIBDEF search chain of CICS. It is recommended to use the same **VOPRM** for clients that is used by Adabas SQL Server in Entire System Manager, as created in step **B.11.**
- Please add the following parameters to the **VOPRM macro** call in your VOPRM as previously described in the section **VO Parameter Module**:

  ```
  ADALINK=(adabas_link_module,B,R1)
  ```

Dataset Definitions

- Modify the LIBDEF search chain in your CICS startup job to include both the Adabas SQL Server sublibrary **and** the sublibrary into which the VOPRM for CICS created above was linked.
- Copy the **P#ESQRUN.D** member created in step **A.6** from the Adabas SQL Server sublibrary into a VSE SD or VSAM file. This file is processed using CICS Transient Data functionality which does not support Librarian files.
- Add a // DLBL ESQPARM, '...' statement to your CICS startup job pointing to the file created above.

TWASIZE Requirements

- Verify and if necessary modify the TWASIZE in CICS. All CICS transactions using Adabas SQL Server require a minimum TWASIZE of 32 bytes.

Step D.3: Installation of Adabas ODBC Clients (SMA Job P060, Step 5400)

Refer to the product documentation for a detailed description of the individual installation steps.

In order to avoid problems during Adabas ODBC installation verification, please ensure that the Net-Work/CSCI interface has been properly installed. This is described in the *Entire Net-Work Installation Manual* in the chapter “Interfaces/Setting Up The Environment”.

The software pre-requisites for Adabas ODBC are described on page 2 of the *Adabas ODBC Client Release Notes*.

IMPORTANT – ensure that the location of the Net-Work/CSCI dll’s has been included in the DOS path. Not doing so may result in the following error condition:

```
ESQ8681, Load of CPI:cpi() failed.
```

Note:
Entire Net-Work must have been installed PRIOR to installing Adabas ODBC.

Phase E: Installing the Adabas SQL Server Utilities

This section discusses the installation of the Adabas SQL Server Utilities.

Utilities Requiring Additional Installation Steps

ADVANCED Interactive Facilities (ADVANCED)

Natural based application loaded into Natural library SYSESQ. This application requires additional installation steps described in detail below.

Utilities not Requiring Additional Installation Steps

BASIC Interactive Facilities (BASIC)

Utility program (ESQBIF) provided in the Adabas SQL Server sublibrary. No additional installation steps are required for the use of this utility.

Adabas SQL Server Migration Utility

Utility program (ESQMIG) provided in the Adabas SQL Server load library. No additional installation steps are required for the use of this utility.

Generate Table Description Utility

Utility program (ESQGTD) provided in the Adabas SQL Server sublibrary. No additional installation steps are required for the use of this utility.

Installation Steps for the Adabas SQL Server Utilities

Note:
Please contact the Natural System Administrator at your site for assistance in completing the following steps.

Step E.1: Modify the Natural Parameter Module (NATPARM)

- Add the following parameters to your NATPARM:

```
NTFILE   ID=239,DBID=ddd,FNR=nnn
NTFILE   ID=137,DBID=ddd,FNR=nnn+1
```

The *ddd*, *nnn* and *nnn+1* should point to the database and file numbers of the first two Adabas SQL Server Catalog files that were loaded in Step **A.3**:

```
NTFILE   ID=239,DBID=198,FNR=170
NTFILE   ID=137,DBID=198,FNR=171
```

In this example, the Catalog files were loaded into database 198 as file numbers 170, 171 and 172.

- Add the following parameter either to your NATPARM or to your dynamic parameters:

```
DELETE=OFF
```

If this parameter is not defined, the application will terminate abnormally.

- If you are running Natural under CICS, you must also add the following parameters to your NATPARM:

```
NTPRM PSEUDO=OFF
```

- After you have made the above changes, **NATPARM** will have to be reassembled and relinked using the appropriate job from your existing Natural installation.

Step E.2: Relink the Natural NUCLEUS

- Relink your Natural nucleus using the appropriate job from your Natural installation. Ensure that the modified NATPARM module created above is available to the linker. Add the following "includes" to the Natural linkedit job:

```
INCLUDE ESQHD21
```

Ensure that entries for both the Adabas SQL Server sublibrary and the Adabas sublibrary are included in the LIBDEF search chain for the linker.

Com-plete Users:

- If you are running Natural (Version 2.2 or below) under Com-plete, ensure that the current MEMRES value in the NATCOMPT source (thread resident portion of Natural) is increased by a minimum of 164,000.
- If you are running Natural (Version 2.3 or above), ensure that the value of the NTHSIZE parameter in the NATPARM is at least 164,000 bytes smaller than the COM–PLETE thread size.
- Reassemble this module using the appropriate job from your existing Natural installation.

CICS Users:

- If you are running a Natural (Version 2.2 or below) under CICS your Natural sublibrary contains a member NCIPARM which contains a variable ADANAME. This ADANAME variable must be set to the name of the Adabas link routine that was specified in your VOPRM created in the **Installation of Clients under CICS** steps above. In our example this name was **Adabas**.
- If you are running a Natural (Version 2.3 or above) the variable ADANAME in the Natural CICS parameter module must be of the same value as specified in VOPRM.
- Relink Natural.

  If the Adabas link routine is linked in 24 bit mode you will receive a S0C4 abend when attempting to run the ADVANCED utility. Adabas must be linked with AMODE=31.

  If the DFHDCT entries are not added properly or the updated DFHDCTPR is not used in your CICS a **–8600** error will occur when you attempt to use the ADVANCED utility.

Step E.3: Load the Natural Programs for the Application SYSESQ (SMA Job I061, Step 5400)

- Use the batch Natural INPL job from your existing Natural installation to load the Natural programs. The CMWKF01 DLBL should point to the ESQvrs.INPL dataset on the installation tape, where *vrs* equals the version of Adabas SQL Server you are installing.

Step E.4: Load the Natural Error Messages for the Application SYSESQ (SMA Job I061, Step 5401)

- Use the batch Natural Error-Message Load job from your existing Natural installation to load the error messages for the ADVANCED utility. The CMWKF02 DLBL should point to the ESQvrs.ERRN dataset on the installation tape, where *vrs* equals the version of Adabas SQL Server you are installing.

Step E.5: Natural Security Definitions (SMA Job I100, Step 5400)

- If you are going to use ADVANCED in a Natural Security environment you will need to define the application **SYSESQ** to Natural Security with **MENU** as the start-up program.

Phase F: Activating the Adabas SQL Server Security Features

Step F.1: Alter the Password of the Administration User DBA (SMA Job I200, Step 5408)

The purpose of this step is to assign the administration user DBA a password, as described in the chapter **The Adabas SQL Server Security Concept/User Administration** of the *Adabas SQL Server Programmer's Guide*.

- Edit the member **J#ALTER.J** in the Adabas SQL Server sublibrary and update it to match the correct dataset names and parameter values for your site. The following steps are performed in this job:
  - Alter the password of the administration user "DBA", and
  - Verify the password change by executing a SELECT statement.
- Submit the member **J#ALTER.J** after making the relevant modifications.

The Adabas SQL BASIC Interactive Facility is used to communicate with the server.

For further information on the ALTER statement, please refer to the *Adabas SQL Server Reference Manual*.

The Data Manipulation Language (DML) statement used to query the CONNECT user identifier is provided in the member **SELUSER.D** in the Adabas SQL Server sublibrary.

Step F.2: Create a User in the Adabas SQL Server Catalog (SMA Job I200, Step 5409)

The purpose of this step is to create a user in the Adabas SQL Server catalog files. This user ID will be used in the following step to verify the External Security Interface.

The user identifier *(login user_id)*, which is to be created, should already be known to the external security product.

The user identifier should be created in the Adabas SQL Server catalog ***without*** password protection, even when this user identifier is password protected by the external security product. It is recommended, that password maintenance be performed using the external security product.

- Edit the member **J#USER.J** in the Adabas SQL Server sublibrary and update it to match the correct dataset names and parameter values for your site. The purpose of this job is to:
  - Create the *login_user_id* in the Adabas SQL Server catalog, and
  - Verify the user definition by executing a SELECT statement.
- Submit the member **J#USER.J** after making the relevant modifications.

  The Adabas SQL BASIC Interactive Facility is used to communicate with the server and query the CONNECT user identification.

  For further information on the CREATE USER statement, please refer to the *Adabas SQL Server Reference Manual*.

  Note:
  *Steps **F.3** and **F.4** should not be performed when external security products are in use.*

Step F.3: Activate and Verify the External Security Interface in Client/ServerMode (SMA Job I055, Step 5400, 5403) (SMA Job I200, Step 5410, 5411)

In this step, you will be establishing the parameter values, which are critical to your successful use of the Adabas SQL External Security Interface.

VOPRM Parameters

- Edit the member **P#VOPRM.A** which was used in step **B.11** to create the VO Parameter module **VOPRM**. Activate the External Security Interface by setting the value of the VOPRM Macro EXTSEC parameter to YES.
- Create the VOPRM module for Adabas SQL Server, as was done in step **B.11**, using the example job **J#VOPRM.J**, which is provided in the Adabas SQL Server sublibrary.

SYSPARM Parameters

- Modify the Entire Service Manager SYSPARMS, which were created in step **B.12**. Activate the External Security Interface by setting the SECSYS parameter to one of the following values: ACF2, RACF or TOPSECRET. The value to be assigned is dependent upon the external security product, which is installed at your site.

Activate the External Security Interface

In order to pick-up the parameter changes and activate the external security interface, the client/server environment must be terminated and re-started..

- Terminate Entire Service Manager as described in step **B.19**. This can be done by executing the following operator command:

```
MSG pp   (where pp is the partition in which the ESG is executing)
rrr EOJ  (where rrr is the reply ID)
```

- Start Entire Service Manager as described in step **B.18**. This can be done by submitting the job created in **B.17**.

Verify the External Security Interface

The *login_user_id* must be defined to both Adabas SQL Server and the external security product. This user identification should have been defined in step **F.2**, if not please do so now.

This *login_user_id* is used to connect to the server, and is passed to the external security interface for authentication checking.

- Submit the member **J#VERSEC.J** after making the relevant modifications. An *invalid user password* should be used to verify the authentication checking. The result should be an authorization error – an Adabas SQL response code: ESQ6701.
- Submit the member **J#VERSEC.J** again, using this time a *valid user password*. The job should now process successfully.

  The Adabas SQL BASIC Interactive Facility is used to communicate with the server and query the CONNECT user identification.

Step F.4: Activate and Verify the Adabas SQL External Security Interface in LINKED-IN Mode (SMA Job I055, Step 5401, 5404) (SMA Job I200, Step 5412, 5413)

In this step, you will be establishing the parameter values, which are critical to your successful use of the Adabas SQL External Security Interface.

VOPRM Parameters

- Create the VOPRM module for Adabas SQL Server by using the example job **J#VOPRMS.J** in the Adabas SQL Server sublibrary. The associated VOPRM parameters used as input to this job can be found in member **P#VOPRMS.A** in the Adabas SQL Server sublibrary. Activate the External Security Interface by setting the value of the VOPRM Macro EXTSEC parameter to YES.
- Submit the member **J#VOPRMS.J** after making the relevant modifications. The VOPRM module as supplied in the ESQvrsS sublibrary will be overwritten..

Verify the External Security Interface

The *login_user_id* must be defined to both Adabas SQL Server and the external security product. This user identification should have been defined in step **F.2**, if not please do so now.

This *login_user_id* is used to connect to the server, and is passed to the external security interface for authentication checking.

- Submit the member **J#VERSES.J** after making the relevant modifications. An *invalid user password* should be used to verify the authentication checking. The result should be an authorization error – an Adabas SQL response code: ESQ6701.
- Submit the member **J#VERSES.J** again, using this time a *valid user password*. The job should now process successfully.

The Adabas SQL BASIC Interactive Facility is used to communicate with the server and query the CONNECT user identification.

Warning:
The user identifier used in the CONNECT statement should be the same as the login_user_id*, when using the external security interface in LINKED-IN mode. An SQL response (ESQ6701) will be returned, in the case that the user identifiers differ and the server process is not authorized to change the* login_user_id*.*

4

OPERATING ADABAS SQL SERVER

The chapter contains the information necessary to successfully operate Adabas SQL Server system with all its components and interfaces.

General Information

This section contains the Adabas SQL Server utilities and user programs which communicate with the Adabas DBMS using ANSI/ISO SQL (Structured Query Language).

The execution of the 3 major components of Adabas SQL Server (Precompiler, Compiler and Runtime System) can be influenced by parameter settings using PPL (Parameter Processing Language) and the SQL parameter module (VOPRM) which must be present in a mainframe environment. For additional information regarding PPL refer to the respective appendix later in this manual.

Although the normal mode of operation is multi-user mode, it is also possible to execute an application program together with an Adabas SQL LINKED-IN Server in the same address space. This can be performed by appending the sublibrary containing the LINKED–IN single–user **VOPRM** to the beginning of the LIBDEF search chain.

Dataset Overview and Description

The following is a list of datasets (first the required, then the optional ones), their purpose and status as well as the component(s) which depend on them. This list is followed by a detailed description of each dataset. The Generate Table Description Utility is named GTD in the following table.

Dataset Name	Description	Used by Component	File Assignment
ESQWORK	Temporary work File	Runtime System, Server	Required
ESQPCIN	Precompiler Input File	Precompiler	Required
ESQPCOU	Precompiler Output File	Precompiler	Required
ESQERRL	Precompiler Error Listing File	Precompiler	Required
ESQGTDO	GTD Utility Output File	Generate Table Description Utility	Required
ESQMIGE	Migrate Utility Error File	Migration Utility	Required
ESQMIGO	Migrate Utility Output File	Migration Utility	Required
ESQPARI	Include Facility Parameter (PPL) Input File	Include Facility	Required
ESQPARM	Parameter (PPL) Input File	Precompiler, Runtime System	Required
ESQPCII	Include Facility Input File	Include Facility	Required
ESQPCLB	Include Facility Copycode Input File	Include Facility	Required
ESQPRIN	Utility Output File	BASIC Interactive Facility	Required
SYSIN	Utility Input File	BASIC Interactive Facility	Required
SYSOUT	Utility Output File	BASIC Interactive Facility, Migration Utility	Required
SYSPRIN	Utility Output File	BASIC Interactive Facility	Required
SxxxWKn	External Sort Files	Runtime System,Server	Required
ESQCLOG	Adabas Command Log File	Client, Precompiler, Server, Utilities	Optional
ESQDMPG	Meta Program Dump File	Precompiler, Runtime System	Optional
ESQGTDS	GTD Utility Input SYSTRANS File	Generate Table Description Utility	Optional

Dataset Name	Description	Used by Component	File Assignment
ESQMIGT	Migrate Utility Trace Output File	Migration Utility	Optional
ESQTRAC	Trace Output File	Precompiler, Runtime System	Optional

Required Parameters

ESQWORK – Temporary Work File

The dataset ESQWORK is used for allocating temporary work files and for sorting by the Adabas SQL Server runtime and should be assigned to all jobs making use of the runtime components.

ESQPCIN – Precompiler Input File

The dataset ESQPCIN is the Adabas SQL Server Precompiler input file and contains the source program to be precompiled.

Note:
When you are executing the Adabas SQL Server Include Facility, the COBOL precompiler is automatically executed after all copycode has been included. The file ESQPCIN is therefore the output file from the Include Facility.

ESQPCOU – Precompiler Output File

The dataset ESQPCOU is the Adabas SQL Server Precompiler output file and contains the precompiled source program, i.e. the source input intended for compilation. In the case of precompilation errors, this file may be empty.

ESQERRL – Precompiler Error Listing File

The dataset ESQERRL is used by the Adabas SQL Server Precompiler to protocol the execution (e.g. source listing, error messages and codes, and warnings).

ESQGTDO – Generate Table Description Utility Output File

The dataset ESQGTDO is used by the Adabas SQL Server Generate Table Description Utility and contains the output DDL statements generated by this utility. This file may in turn be used as input to the Adabas SQL Server BASIC Interactive Facility.

ESQMIGE – Migration Utility Error File

The dataset ESQMIGE is used by the Adabas SQL Server Migration Utility and contains error messages and warnings generated by this utility.

ESQMIGO – Migration Utility Output File

The dataset ESQMIGO is used by the Adabas SQL Server Migration Utility and contains the output DDL/DML statements generated by this utility. This file may in turn be used as input to the Adabas SQL Server BASIC Interactive Facility.

ESQPARI – Include Facility Parameter (PPL) Input File

The dataset ESQPARI contains the Adabas SQL Server Parameters (PPL). This dataset is used only by the Adabas SQL Server Include Facility.

The contents of this dataset are described in detail in **Appendix A**, **The Parameter Processing Language (PPL)**.

ESQPARM – Parameter (PPL) Input File

The dataset ESQPARM contains the Adabas SQL Server Parameters (PPL). This dataset is used by almost all components of Adabas SQL Server.

- Adabas SQL Server Client
- Adabas SQL Server (Server)
- Adabas SQL Server Precompiler
- Adabas SQL Server Utilities

The contents of this dataset are described in detail in **Appendix A**, **The Parameter Processing Language (PPL)**.

ESQPCII – Include Facility Input File

The dataset ESQPCII is the Adabas SQL Server Include Facility input file and contains the COBOL source program to be precompiled.

ESQPCLB – Include Facility Copybook Sublibrary

The dataset ESQPCLB is the Adabas SQL Server Include Facility Copybook sublibrary and contains copycode which will be included into the COBOL source program to be precompiled.

ESQPRIN – Utility Output File

ESQPRIN is used as an output file by almost all components of Adabas SQL Server.

- Adabas SQL Server Client
- Adabas SQL Server (Server)
- Adabas SQL Server Precompiler
- Adabas SQL Server Utilities

SYSIN – Utility Input File

SYSIN is the source input file for Adabas SQL Server Utilities:

- Adabas SQL Server BASIC Interactive Facility
- Adabas SQL Server Generate Table Description Utility

SYSOUT – Utility Output File

SYSOUT is used as an output file by almost all components of Adabas SQL Server.

- Adabas SQL Server Client
- Adabas SQL Server (Server)
- Adabas SQL Server Precompiler
- Adabas SQL Server Utilities

SYSPRIN – Utility Output File

SYSPRIN is used as an output file by almost all components of Adabas SQL Server.

- Adabas SQL Server Client
- Adabas SQL Server (Server)
- Adabas SQL Server Precompiler
- Adabas SQL Server Utilities

SxxxWKn – External Sort Files

The system sort routine requires workfiles when sorting larger amounts of data. These workfiles are assigned to the DLBL SxxxWKn. These should be assigned in triplets and should be allocated at least as large as the ESQWORK dataset.

Optional Parameters

ESQCLOG – Command Log File

The dataset ESQCLOG is used by Adabas SQL Server for the logging of commands issued to the Adabas call interface and is activated by the Adabas LOG (PPL) statement.

Depending on the log options selected, extremely large amounts of output may be written to this dataset.

ESQDMPG – Meta Program Dump File

The dataset ESQDMPG is used by Adabas SQL Server for the dumping/listing of a created meta-program. This dataset is used by all components of Adabas SQL Server and is only required when using the GLOBAL META-PROGRAM parameter (PPL) described in **Appendix A**.

ESQGTDS – Generate Table Description Utility Input SYSTRANS File

The dataset ESQGTDS is used by the Adabas SQL Server Generate Table Description Utility and contains an Natural DDM SYSTRANS file. This is used primarily for the purpose of supplying long names for columns and tables.

ESQMIGT – Migration Utility Trace Output File

The dataset ESQMIGT is used by the Adabas SQL Server Migration Utility and contains the trace output generated by this utility.

ESQTRAC – Adabas SQL Server Trace File

The dataset ESQTRAC is for internal use only and should not be used without consulting Software AG technical support. The file is used by the TRACE version of Adabas SQL Server and is activated by TRACE (PPL) or environment parameter settings.

Depending on the trace options selected, extremely large amounts of output may be written to this dataset.

Note:
The Adabas SQL Server trace facilities should only be used when attempting to solve problems related to Adabas SQL Server, and under the direction of Software AG technical support.

Operating the Adabas SQL Server Precompiler

Prerequisites

Before precompiling an application program, ensure that the Adabas Nucleus has been started and is active.

The following points should be taken into consideration prior to precompiling an application program and are explained below:

- Execution Parameter
- Default Schema Identifier
- Dataset Requirements
- Starting the Precompiler

Execution Parameter

The precompiler parameters are derived from the parameters (PPL) found in the dataset ESQPARM and from the parameter settings in the module VOPRM**.**

ESQPARM should contain all parameters relevant to precompiling the application program. An example precompile parameter file is supplied in member **P#ESQPC.D** in the Adabas SQL Server sublibrary.

Default Schema Identifier

The value of the default SQL Schema Identifier for the application module is determined by the GLOBAL DEFAULT SCHEMA IDENTIFIER parameter setting (PPL). When this is not provided, then the user ID at execution-time is used as default SQL Schema Identifier.

The value of the user ID is runtime environment dependent. The user ID has a maximum length of 8 characters.

Dataset Requirements

The datasets used by the Adabas SQL Server Precompiler are as follows :

- Input (C/COBOL/PL1) for the Precompiler is read from the dataset ESQPCIN.
- Output (C/COBOL/PL1) from the Precompiler is written to the dataset ESQPCOU.
- Output listing from the Precompiler is written to ESQERRL**.**

Dataset Name	Description	File Assignment
SYSRDR	Adabas Parameter file	Required
SYSLST	Adabas Listing file	Required
ESQERRL	Precompiler Listing	Required
ESQPARM	Parameter File	Required
ESQPCIN	Precompiler Input File	Required
ESQPCOU	Precompiler Output File	Required
ESQPRIN	Listing file	Required
SYSOUT	Listing file	Required
SYSPRIN	Listing file	Required
ESQDMPG	Meta-Program Dump File	Optional *(1)*
ESQTRAC	Trace File	Optional *(2)*

(1) required when using GLOBAL META PROGRAM Parameter (PPL)
(2) required when using TRACE Parameter (PPL or VOPRM)

Starting the Precompiler

Example JCL's for executing the various Adabas SQL Server Precompilers are supplied in the Adabas SQL Server sublibrary in members **J#VERC.J** (C), **J#VERCOB.J** (COBOL) and **J#VERPL1.J** (PL/1).

Operating the Adabas SQL Server Include Facility

The Adabas SQL Server Include Facility provides the functionality to expand EXEC SQL INCLUDE statements in COBOL programs before the programs are processed by the Adabas SQL Server Precompiler. The Include Facility (ESQPCINC) automatically executes the COBOL precompiler after all copycode has been included, therefore the prerequisites and dataset requirements for the Include Facility are basically the same as for the Precompiler above.

Prerequisites

Before precompiling an application program, ensure that the Adabas Nucleus has been started and is active.

The following points should be taken into consideration prior to precompiling an application program and are explained below:

- Execution Parameter
- Default Schema Identifier
- Dataset Requirements
- Starting the Include Facility and Precompiler

Execution Parameter

The precompiler parameters are derived from the parameters (PPL) found in the dataset ESQPARM and from the parameter settings in the module VOPRM.

ESQPARM should contain all parameters relevant to precompiling the application program. A example precompile parameter file is supplied in member **P#ESQPC.D** in the Adabas SQL Server sublibrary.

Default Schema Identifier

The value of the default SQL Schema Identifier for the application module is determined by the GLOBAL DEFAULT SCHEMA IDENTIFIER parameter setting (PPL). When this is not provided, then the user ID at execution-time is used as default SQL Schema Identifier.

The value of the user ID is runtime environment dependent. The user ID has a maximum length of 8 characters.

Dataset Requirements

The datasets used by the Adabas SQL Server Include Facility and Precompiler are as follows:

- Input COBOL source for the Include Facility is read from the dataset ESQPCII.
- Input copycode for the Include Facility is read from the sublibrary ESQPCLB.
- Output (COBOL) from the Include Facility is written to the dataset ESQPCIN.
- Input (COBOL) for the Precompiler is read from the dataset ESQPCIN.
- Output (COBOL) from the Precompiler is written to the dataset ESQPCOU.
- Output listing from the Precompiler is written to ESQERRL.

Dataset Name	Description	File Assignment
SYSRDR	Adabas Parameter file	Required
SYSLST	Adabas Listing file	Required
ESQERRL	Precompiler Listing	Required
ESQPARI	Include Facility Parameter File (Input)	Required
ESQPARM	Parameter File (Output)	Required
ESQPCII	COBOL Input Source File	Required
ESQPCIN	Precompiler Input File	Required
ESQPCLB	Include Facility Copybook Sublibrary	Required
ESQPCOU	Precompiler Output File	Required
ESQPRIN	Listing file	Required
SYSOUT	Listing file	Required
SYSPRIN	Listing file	Required
ESQDMPG	Meta-Program Dump File	Optional *(1)*
ESQTRAC	Trace File	Optional *(2)*

(1) required when using GLOBAL META PROGRAM Parameter (PPL)
(2) required when using TRACE Parameter (PPL or VOPRM)

Starting the Include Facility

An example JCL for executing the Adabas SQL Server Include Facility and Precompiler for COBOL is supplied in the Adabas SQL Server jobs dataset in member **J#PCINC.J**.

Example

```
* $$ JOB JNM=ESQPCINC,CLASS=0,DISP=D
* $$ LST CLASS=A,DISP=H
// JOB ESQPCINC --- EXECUTE INCLUDE FACILITY/PRECOMPILE PROGRAM ---
/*
/* ------------------------------------------------------------- *
/* EXECUTE ESQ COBOL INCLUDE FACILITY AND PRECOMPILE COBOL PROGRAM
/* (Linked-In)
/* ------------------------------------------------------------- *
/*
// LIBDEF  PHASE,SEARCH=(SAGLIB.ESQ142S,                          *
              SAGLIB.ESQ142,                                     *
              SAGLIB.ADA621),TEMP
// DLBL    ESQERRL,'SYSLST'                  <- ESQPCCOB ERROR  LOG
// DLBL    ESQPARI,'SAGLIB.ESQ142(P#ESQPC.D)' <- ESQPCINC PPCARDS
// DLBL    ESQPARM,'SAGLIB.ESQ142(P#ESQPC.$)' <- ESQPCINC PPCARDS
// DLBL    ESQPCII,'SAGLIB.ESQ142(COB2PGM.B)' <- ESQPCINC INPUT  COBOL
// DLBL    ESQPCIN,'SAGLIB.ESQ142(COB2PGM.$)' <- ESQPCINC OUTPUT COBOL
// DLBL    ESQPCLB,'SAGLIB.COPYLIB(#.B)'     <- ESQPCINC INCLUDE LIB
// DLBL    ESQPCOU,'SAGLIB.ESQ142(COB2PGM.C)' <- ESQPCCOB OUTPUT COBOL
// DLBL    ESQPRIN,'SYSLST'                  <- ESQPC*** OUTPUT LIST
// DLBL    ESQWORK,'ESQ.SINGLE.ESQWORK',,VSAM,CAT=VSESPUC
// DLBL    SYSOUT,'SYSLST'
// DLBL    SYSPRIN,'SYSLST'
// EXEC    ESQPCINC,SIZE=AUTO,                                    *
              PARM='COB2PGM,ESQ'             <- PROGRAM,SCHEMA-ID
ADARUN     DATABASE=36
ADARUN     DEVICE=8381
ADARUN     MODE=MULTI
ADARUN     PROGRAM=USER
ADARUN     SVC=45
/*
// EXEC    LISTLOG
/&
* $$ EOJ
```

In the above example the file pointed to be the ESQPARI DLBL statement contains the following statements (in addition to other parameters):

```
GLOBAL
BEGIN
  .
  .
  DEFAULT SCHEMA IDENTIFIER = '$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$'
END
PRECOMPILER
BEGIN
  COBOL LANGUAGE SETTINGS ( COBOL II = ON  )
  COMPILATION UNIT IDENTIFIER ( LIBRARY = 'DEMO    ',
                                PROGRAM =  '########' )
END
```

The first phase of the Include Facility substitutes the correct values for DEFAULT SCHEMA IDENTIFIER and PROGRAM in the GLOBAL and PRECOMPILER sections and writes the resulting parameters to the file pointed to by the ESQPARM DLBL statement which is then used as input to COBOL Precompiler.

```
GLOBAL
BEGIN
  .
  .
  DEFAULT SCHEMA IDENTIFIER = 'ESQ'
END
PRECOMPILER
BEGIN
  COBOL LANGUAGE SETTINGS ( COBOL II = ON  )
  COMPILATION UNIT IDENTIFIER ( LIBRARY = 'DEMO    ',
                                PROGRAM =  'COB2PGM ' )
END
```

The second phase of the Include Facility includes copycode from the sublibrary pointed to by the ESQPCLB DLBL statement. This statement MUST contain both Member Name and Member Type as in the example JCL above, where Member Name is '#' and Member Type the type of data to be included (B,C etc.). The generated COBOL program including all copycode is written to the file pointed to by the ESQPCIN DLBL statement.

The third (last) phase of the Include Facility loads and executes the COBOL Precompiler (ESQPCCOB) using the files generated in the previous phases. The precompiled COBOL program is written to the file pointed to by the ESQPCOU DLBL statement can be used as input to COBOL compiler.

Operating the Adabas SQL Server Zap Table Display Facility

The Adabas SQL Server Zap Table Display Facility displays a list of applied ZAPs.

Prerequisites

The following points should also be taken into consideration prior to starting the facility and are explained below:

- Dataset requirements
- Starting the facility

Dataset Requirements

The dataset requirements of the facility are as follows :

Dataset Name	Description	File Assignment
ESQPRIN	Listing file	Required

Starting The Zap Table Display Facility

An example JCL for executing the Adabas SQL Server Zap Table Display facility is supplied in the Adabas SQL Server jobs dataset in member **J#ZAPT.J**.

The following is an example of the Zap Table Display facility output:

```
Adabas SQL Server (ESQ) Zap Utility
Version     :  1.4.2.7
Patch level :  01
Corrections applied
AQ42001 P123456
Processing completed successfully
```

Creating an SQL Application Program

An application program is prepared for execution in a 3-phase procedure (precompile, compile and linkedit).

Precompile

The Adabas SQL Server Precompiler supports several 3rd generation host languages. Currently supported are C, COBOL, and PL/I. This language information can be passed using an OS parameter card with following syntax :

```
// EXEC ESQPC<language identifier>
```

where the <*language identifier*> can have the values below :

```
C        for C
COB      for COBOL
PLI      for PL/1
```

When your application program design and coding are complete, you are ready to prepare the source statements for execution. The SQL statements embedded in the 3rd generation host language must be prepared by the Adabas SQL Server Precompiler for compilation as host language statements. Before a program can be executed, it must be compiled by a compiler of the appropriate host language (C, COBOL, etc.).

The Adabas SQL Server Precompiler scans every statement of the program source and produces a modified program in which every SQL statement has been replaced by host language statements like variable definitions and a call to the Adabas SQL Server Language Interface Module “ESQO*tt”*, where *tt*=BT for batch, CI for CICS and CO for Com-plete.

Ensure that the Adabas nucleus containing the Adabas SQL catalog and error-messages files is active.

Note:
Refer to the precompiler and compiler parameter settings as described in Appendix A, which influence the precompiler processing. The COMPILATION UNIT IDENTIFIER must be unique for each precompiled program. This parameter value specifies the name of the metaprogram which is generated at runtime. Duplicate names will lead to a performance loss due to unnecessary generation and storage of metaprograms.

Compile

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

Linkedit

Linkediting is performed after compilation. The Adabas SQL Server link interface (ESQO*tt*) must be included during this phase.

The module names for the individual environments are:

ESQO*tt* Module	**Environment**
ESQOBT	Batch
ESQOCI	CICS
ESQOCO	Com-plete

Retrieving Error Message Texts

To retrieve error message texts in the application program please refer to the parameter structure and example call to ESQERR in the Adabas SQL Server sublibrary in member **ESQERR.A.**

COBOL Precompiler – Include Facility

The Adabas SQL Server Precompilers will generate code when confronted with the following statements:

```
EXEC SQL
INCLUDE SQLCA
END–EXEC
```

In order to process other INCLUDE statements in COBOL, it is necessary to use the Include Facility**.** This facility physically includes the contents of the file to be included and removes the file from the “EXEC SQL” statement. The resultant pre-source is passed on to the precompiler.

An example JCL for executing the Include Facility is supplied in the Adabas SQL Server sublibrary in member **J#PCINC.J**.

Note:
The INCLUDE SQLCA statement is optional. The Adabas SQL Precompilers will automatically generate an SQLCA if no INCLUDE statement is detected.

C Precompiler – Digraphs and Trigraphs

The following digraphs and trigraphs for host variables are now supported:

Normal Char	Digraphs	Trigraph
[	(| or (!	??(
]	|) or ¦)	??)

Examples for host variable array definitions in C programs are:

```
char host_var [10];
```

The following digraphs may be used:

```
char host_var (!10!);
char host_var (|10|);
```

The following trigraph may be used:

```
char host_var ??(10??)
```

5

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 enables the user to submit SQL statements which are then batch processed. Catalog information can be retrieved using predefined views.

2 ADVANCED Interactive Facilities (ADVANCED)

ADVANCED is a menu-driven application based on Software AG's products Natural and the Software AG Editor and can only be used in environments where these products are installed.

– ADVANCED Interactive SQL enables the user to enter SQL statements interactively and see results displayed on the screen.
– ADVANCED Catalog Retrieval enables the user to retrieve information stored in the catalog, which is an Adabas file.

3 Adabas SQL Server Migration Utility

This utility serves as a migration tool especially designed for the upgrade from Adabas SQL Server Version 1.3 to Adabas SQL Server Version 1.4.2.

4 Generate Table Description Utility

If existing Adabas files must be introduced to Adabas SQL Server using 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 using the BASIC Interactive SQL Utility.

BASIC Interactive Facilities

Target Users

BASIC has been designed for users who want to:

- submit ad–hoc SQL statements, for example for database maintenance tasks,
- retrieve catalog information, for example, databases, tables, views.

Overview

The BASIC Interactive Facilities are a batch query tool which supports basic functionality.

Statements containing cursor names, host variables or parameter markers cannot be executed in this context.

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

To invoke the BASIC Interactive Facilities and to submit the SQL statements, use the following example JCL or member **J#BIF.J** in the Adabas SQL Server sublibrary:

```
* $$ JOB  JNM=ESQBIF,CLASS=0,DISP=D
* $$ LST  CLASS=A,DISP=D
// JOB ESQBIF --- EXECUTE ESQBIF ---
/*
```

```
/*  ------------------------------------------------------------- *
/*  ESQ Basic Interactive Facility (ESQBIF)
/*  ------------------------------------------------------------- *
/*
// LIBDEF  PHASE,SEARCH=(SAGLIB.ESQ142S,                          *
              SAGLIB.ESQ142,                                     *
              SAGLIB.ADA621),TEMP
// DLBL    ESQPARM,'SAGLIB.ESQ142(P#ESQRUN.D)'
// DLBL    ESQPRIN,'SYSLST'
// DLBL    ESQWORK,'ESQ.SINGLE.ESQWORK',,VSAM,CAT=VSESPUC
// DLBL    SYSIN,'SYSRDR'
// DLBL    SYSOUT,'SYSLST'
// DLBL    SYSPRIN,'SYSLST'
// EXEC    ESQBIF,SIZE=AUTO,PARM='ESQUSER=PJ'
ADARUN     DATABASE=36
ADARUN     DEVICE=8381
ADARUN     MODE=MULTI
ADARUN     PROGRAM=USER
ADARUN     SVC=45
/*
SELECT * FROM INFORMATION_SCHEMA.TABLES ;
// EXEC    LISTLOG
/&
* $$ EOJ
```

Predefined Catalog Retrieval Views

To enable easy retrieval several views have been predefined. They are listed in the *Adabas SQL Server Programmer's Guide*, **Appendix C**, for example:

To display all tables, the following:

SELECT * FROM INFORMATION_SCHEMA.TABLES;

To display all tablespaces, type the following:

SELECT * FROM INFORMATION_SCHEMA.TABLESPACES;

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

SELECT * FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME = "SAILOR";

To display all indexes, type the following:

SELECT * FROM INFORMATION_SCHEMA.TABLE_INDEXES;

ADVANCED Interactive Facilities

Target Users

The ADVANCED Interactive Facilities, called ADVANCED in the following, depends on Software AG's products Natural/Software AG Editor to be installed first. As a menu-driven application ADVANCED offers the user quick and easy access, but with far more comfort than BASIC, to:

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

Overview

ADVANCED enables the user to execute SQL statements dynamically and to retrieve information from the catalog.

In the case of security being enabled for the current default server the password is required for the ADVANCED function Interactive SQL. The user ID to which the password belongs is defined by the *init-user function in Natural. Refer to the appropriate Natural Reference Manual for your platform. For the ADVANCED function Catalog Retrieval a password is not necessary and will be ignored if specified.

Invoking the ADVANCED Interactive Facilities

ADVANCED must have been installed as described in the installation chapter earlier.

ADVANCED is a Natural application and is located in the Natural library SYSESQ. ADVANCED can be started using the startup program MENU and the following main menu will appear.

```
16:00:46      ***  ADABAS SQL Interactive Facilities  ***          97-12-10
                             Main Menu




                      Code Function
                      ---- ----------------------------
                       D   Disconnect
                       I   Interactive SQL
                       R   Catalog Retrieval
                       ?   Help
                       .   Exit
                      ---- ----------------------------
                Code .. _    Userid .:
                             Password:




Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
     Help        Exit                                                  Canc
```

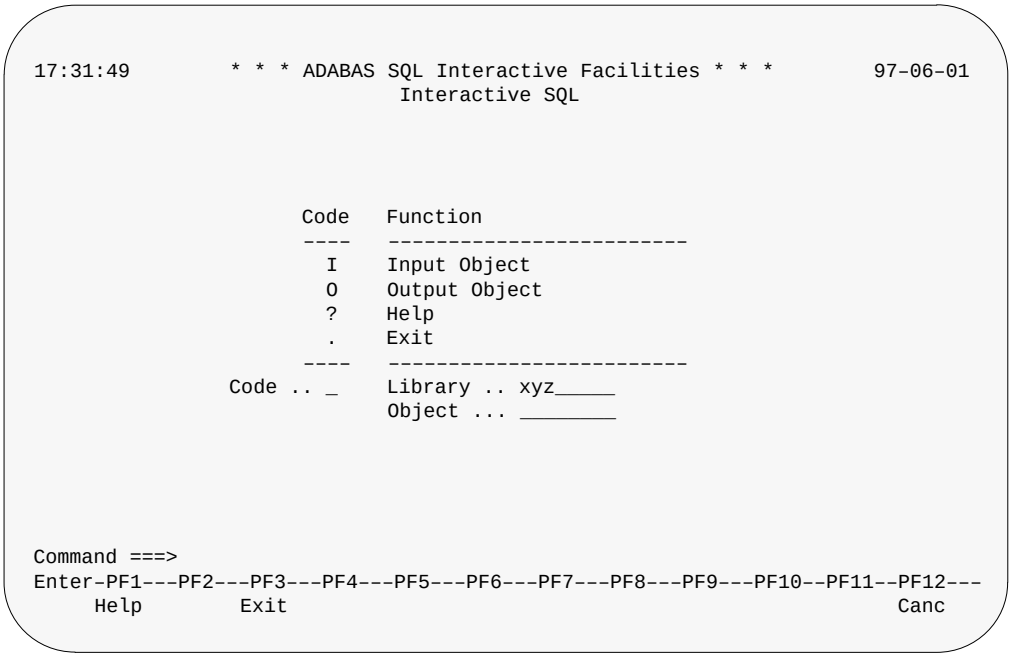
The function Disconnect terminates the session between Adabas SQL Server and ADVANCED.

ADVANCED Interactive SQL

Interactive SQL is a menu-driven Natural application which enables the user to execute SQL statements dynamically. Statements containing cursor names, host variables or parameter markers cannot be executed interactively. As well as 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.

How to Start and End ADVANCED Interactive SQL

Interactive SQL is started by entering function code “I” in the Main Menu. The system then presents the Interactive SQL menu offering the functions: Input Object, Output Object, Help and Exit, as shown below:



The Interactive SQL Menu Options

Code	Description
I	The Input function serves to display, edit and execute input objects like DDL and DML statements in the interactive SQL input screen.
O	The Output function serves to display output object like resultant tables and SQL return codes in the interactive SQL output screen.
?	The Help function equips each screen with one or more help texts regarding the actual field or position within the program. The Help function is invoked by typing '?' or pressing PF1 wherever help is wanted.
.	To exit Interactive SQL press PF3 while in the main menu or enter period in the Code field of the main menu. Pressing PF3 always takes you one step back to the previous screen or function. When PF3 is pressed on an editor screen where modifications have been made, the Exit Function window is displayed.

The following parameters can be specified:

Parameter	Description
Library *library-name*	Specifies the name of the current Natural library which contains the specified input/output objects. Specification of libraries whose name begins with "SYS" is not allowed.
Object *object-name*	If a valid object name is specified, the corresponding object is displayed. If a value is specified followed by an asterisk (*), all input/output objects in the current library whose name begins with that value are listed. If asterisk notation is specified only, a selection list of all input/output objects in the current library is displayed. If the Object field is left blank, the blank SQL input/output screen is displayed.

The Input as well as the Output functions make use of an editor which is very similar to the Software AG Editor. The editor commands are described in detail in the online help facility which can be invoked by pressing PF1. The most commonly used commands are explained below.

Interactive SQL Commands

The commands available in Interactive SQL can be divided into three groups:

- Global Maintenance Commands
- Editor Main Commands
- Editor Line Commands

Global Maintenance Commands

Within Interactive SQL, the following global maintenance commands apply:

Command	Description
COPY *name*	Copies the specified object from the current library into the editor, after (A) or before (B) the current line.
LIBRARY *name*	Specifies the library "*name*" as current library.
LIST *name**	Lists all objects from the current library whose name starts with "*name*". From the list, you can select an object marking it with "S".
PURGE *name*	Purges the specified object from the current library.
READ *name*	Reads the specified object from the current library into the editor. The current name is set to "*name*".
SAVE [*name*]	Saves the actual screen contents under the object *"name"* in the current library. If no name is specified, the current name is taken. Current library and object names are displayed above the command line.

Object and library names must correspond to the Natural naming conventions (see the *Natural Reference Manual* for details). Objects can be input objects or output objects.

Global PF-Key Settings

Within Interactive SQL, the following global PF-key settings apply:

Key	Setting	Description
PF1	Help	Pressing PF1 invokes the Interactive SQL online help system from any Interactive SQL screen.
PF3	Exit	Pressing PF3 always takes you one step back to the previous screen or function. When pressed on an editor screen where modifications have been made, the Exit Function window is displayed (see also the chapter **Editors** of the **Natural Utilities Manual**). When you press PF3 in the main menu, you leave Interactive SQL.
PF12	Canc	Pressing PF12 always takes you back to the menu from where the current screen has been invoked. When you press PF12 in the main menu, you leave Interactive SQL.

Editor Main Commands

The free-form editor available within Interactive SQL requires that the Software AG Editor is installed. The main and line commands available are a subset of those available within this editor.

Main commands are entered in the command line of the editor screen. The most important main commands are:

Command	PF Key	Description
BOTTOM (++)		Positions to the bottom of the data.
CHANGE		Scans for a specified string and replaces each such string found with another specified string.
CLEAR		Clears the editor source area.
DELETE		Deletes the line(s) containing a given string according to the specified selection operands.
DOWN (+)	PF8	Scrolls the specified scroll amount downwards.
FIND		Finds a string specified by command operands at the location(s) specified by selection operands.
FIX *n*		Fixes the first *n* number of columns to display when scrolling lists in editor format right. *+ n* scrolls down by *n* lines. *– n* scrolls up by *n* lines
LEFT	PF10	Scrolls the specified scroll amount to the left.
LIMIT *n*		Sets a limit for the FIND command; *n* lines are processed.
PRINT		Prints the data displayed.
RESET		Resets all pending line commands.
RFIND	PF5	Repeats the last FIND command.
RIGHT	PF11	Scrolls the specified scroll amount to the right.
TOP (—)		Positions to the top of the data.
UP (–)	PF7	Scrolls the specified scroll amount upwards.

The scroll amount for the UP, DOWN, LEFT, and RIGHT commands is specified in the SCROLL field at the top right corner of the list screen. Valid values for the scroll amount are:

Value	Scroll Amounts
CSR	determined by cursor position
DATA	equals the page size less one line
HALF	is half the page size
MAX	equals the amount of data to the bottom/top
PAGE	is equal to the page size
n	is equal to *n* lines

Editor Line Commands

Line commands are entered in the editor prefix area of the corresponding statement line. The most important line commands are:

Command	Description
A	Inserts line(s) to be moved or copied after the current line.
B	Inserts line(s) to be moved or copied before the current line.
C	Copies the current line.
CC	Marks the beginning and end of a block of lines to be copied.
D	Deletes the current line.
DD	Marks the beginning and end of a block of lines to be deleted.
I*nn*	Inserts *nn* new lines after the current line.
M	Moves the current line.
MM	Marks the beginning and end of a block of lines to be moved.

Both main commands and line commands are described in detail as part of the Interactive SQL online help facility, which is invoked by pressing PF1 (Help). For further details, please refer to the relevant Software AG Editor documentation.

The Function: Input Object (Action "I")

To invoke this function enter function code "I" in the Code field of the Interactive SQL screen.

Depending on the object name specified, the particular SQL statement or the statements will be displayed.

If the Object field is left blank, the SQL input screen will appear. The statements can be edited using the Editor Main or Line Commands as described above. It is possible to enter more than one statement. Each statement must end with a semicolon.

The objects can be saved, listed, retrieved, copied and purged by using the Global Maintenance Commands.

The Setup Function – Processing SQL Statements

To set up a processing profile for Interactive SQL, a sub-menu can be displayed by pressing PF2. The following options are provided:

- Execute statements one by one.

  After the execution of each statement, the output screen is shown immediately. The user can decide whether to execute the next statement by pressing PF4 or to return to the input screen by pressing PF3.

- Execute all statements together.

  All statements are executed together and the output screen shows all results.

- Optional COMMIT/ROLLBACK.

  A window is invoked after the execution of each statement offering the option to either COMMIT or ROLLBACK the last database modifications. For details on COMMIT or ROLLBACK refer to the *Adabas SQL Server Reference Manual*.

- Automatic COMMIT/ROLLBACK.

  Each database modification is automatically either committed or rolled back, depending on whether all statements were executed successfully.

- Header length every *n* data lines.

  To allow for easier orientation while browsing through extensive output data, the header with the column names can be inserted after every *nth* line.

- Record length data session: *m.*

  The record length for the output data for one session can be set to avoid truncation. The > (greater than) character indicates whether truncation occurred.

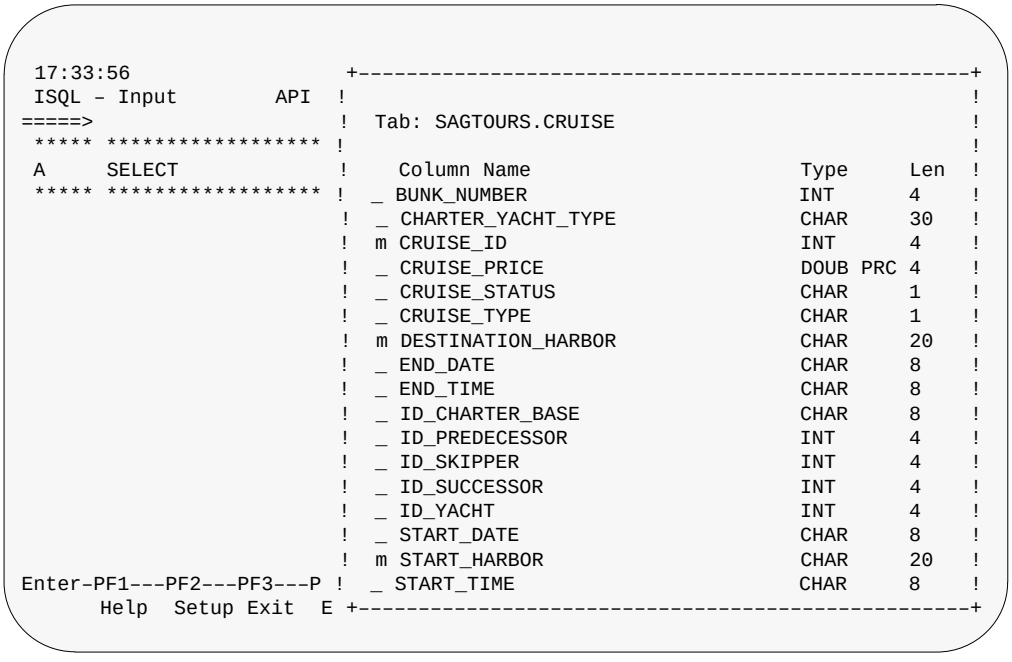
The SELECT Command – Listing Catalog Information

To assist in coding an SQL object, exisiting SQL tables and columns can be listed in pop-up windows using the SELECT command. From the list, you can include table and column names into the editor. The SELECT command is available for table and column selection:

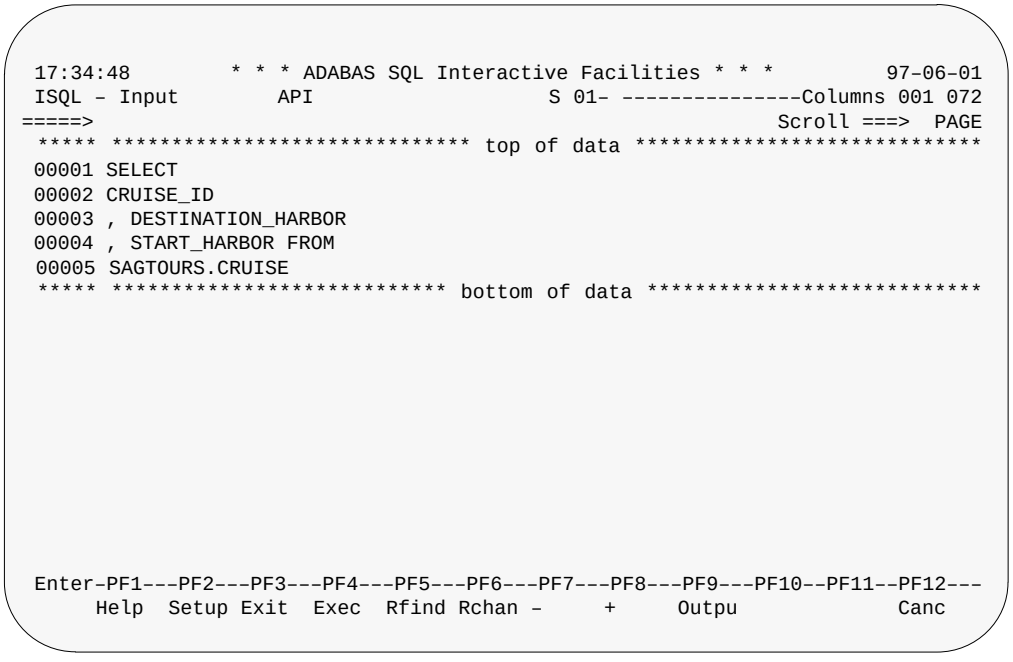
Command	Description
SELECT TABLE *table specification*	Selects all tables matching the specified table specification. You can specify a value followed by an asterisk (*), and all tables whose name begins with this value are selected. If you specify the asterisk notation alone, all existing tables are selected.
SELECT COLUMN *table specification*	Selects all columns of the specified table. Since the table must be uniquely identified, asterisk notation cannot be used.

Note:
If you specify an unqualified table (without a schema identifier) all tables found in the catalog with the specified name are selected, regardless of which schema they belong to.

From the column list, you can select columns to be inserted in the statement which is currently in preparation.



After the selected columns have been inserted, only the FROM has to be added manually to conform to the syntax requirements.



Execution of Input Objects

The SQL statement loaded in the editor can be executed immediately by pressing PF4 (EXEC) or by typing the command EXEC in the command line. The results are automatically shown on the output screen. Pressing PF3 returns the user to the input screen.

The Function: Output Object (Action “O”)

This function is invoked either by entering the function code “O’’ or by executing an SQL statement like described above.

The Global Maintenance Commands List, Save, Retrieve, Copy and Purge can be used within the Output function.

The output screen shows the statement which was executed and the resultant table.

```
17:36:33        * * * ADABAS SQL Interactive Facilities * * *      97-06-01
ISQL - Output       API                   S 02- ------------Columns 001 072
=====>                                                    Scroll ===>  PAGE
***** ***************************** top of data *************************
 00001 SELECT
 00002 CRUISE_ID
 00003 , DESTINATION_HARBOR
 00004 , START_HARBOR FROM
 00005 SAGTOURS.CRUISE
  00006 ------------------------------------------------------------------------
 00007   CRUISE_ID DESTINATION_HARBOR   START_HARBOR
  00008 ------------------------------------------------------------------------
 00009    10000001 LIVERPOOL            WILHELMSHAVEN
 00010    10000002 WILHELMSHAVEN        LIVERPOOL
 00011    10000003 NEAPEL               LISSABON
 00012    10000004 WILHELMSHAVEN        NEAPEL
 00013    20000001 MIAMI                NASSAU
 00014    20000002 HAVANNA              MIAMI
 00015    20000003 SANTO DOMINGO        HAVANNA
 00016    30000001 MIAMI                NASSAU
 00017    30000002 HAVANNA              MIAMI
 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12
      Help        Exit        Rfind Rchan -     +            <     >    Canc
```

The Help Function (Action “?”)

At any point where help is required, entering a “?’’ will present a window with the appropriate information to continue. For example, typing “?’’ in the Code field of the ADVANCED main menu will display the following help screen:

```
12:24:22                                                          97-06-02
                        Help System - Interactive SQL

The Interactive SQL allows to execute SQL statements dynamically.


On the input screen you can enter one ore more SQL statements using the editor.
You can save SQL statements in a NATURAL library. After executing the SQL
statements, the retrieved data (in case of a SELECT) and the ESQ status infor-
mation will be displayed on the output screen. The retrieved data can be saved
for later use.
                         Code Topic
                         ---- ----------------------
                          1   Interactive SQL input screen

                          2   Execute SQL statements
                          3   Interactive SQL output screen

                          4   Editor commands
                          .   Exit Help
                         ---- ----------------------
                  Code .. _

 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit                                                 Canc
```

ADVANCED Catalog Retrieval

How to Invoke and End ADVANCED Catalog Retrieval

The Catalog Retrieval function is invoked by entering function code “R” on the ADVANCED Interactive Facilities main menu. The Catalog Retrieval screen is displayed:

```
16:01:24      ***  ADABAS SQL Interactive Facilities  ***          97-12-10
                        Catalog Retrieval




                    Code Function            Parameter
                    ---- ------------------- --------------------
                     D   List Databases      Database
                     T   List Tables         Schema Name,Table Name
                     ?   Help
                     .   Exit
                    ---- ------------------- --------------------
               Code .. _   Database .... ____________________
                           Schema Name.. ____________________
                           Table Name..  ____________________



Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
     Help        Exit                                                  Canc
```

Code	Description
D	The List Databases function lists databases defined in the catalog.
T	The List Tables function lists tables defined in the catalog.

To end the Catalog Retrieval function and to return to the ADVANCED main menu enter “.” or press PF3.

The following parameters must be specified as selection criteria:

Parameter	Description
Database *database-identifier*	The name of the database to be listed. Asterisk notation (*) for the selection of all databases is possible. The Database identifier is relevant to the List Databases function only.
Schema *schema-identifier*	The schema identifier of the table(s) to be listed. Asterisk notation (*) is possible. The schema identifier is relevant to the List Tables function only.
Table *table-identifier*	The name of the table to be listed. Asterisk notation (*) is possible. The table identifier is relevant to the List Tables function only.

List Databases

To invoke the List Databases function, enter function code “D” on the Catalog Retrieval screen. The name of the database(s) to be listed must be specified. If a value followed by an asterisk is specified, all databases defined in the catalog whose name begins with that value are listed. If the Database field is left blank, or if asterisk notation is specified only, all databases defined in the catalog are listed.

```
16:05:20          ***  ADABAS SQL Interactive Facilities  ***          97-12-10
DATABASE *                                    S 01      Row 0 of 3 Columns 001 028
====>                                                     Scroll ===>  PAGE
__ DATABASE              DBID
** ******** top of data *******
__ ESQ_DEMO_DB           36
__ ESQ_NIST_SADVA        36
__ SAKLM_DB              36
** ****** bottom of data ******
__
__
__
__
__
__
__
__
__
__
__
__
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
     HELP        EXIT        RFIND        -     +                      CANC
```

Commands Allowed on Databases

The following line commands are available on the database listing screen. Line commands are entered in front of the desired database(s):

Command	Description
I	Displays information about a database.
S	Selects a database to be used with main commands (see below).
D	Deselects a database.
TB	Displays all tables defined in a database.
TS	Displays all tablespaces defined in a database.

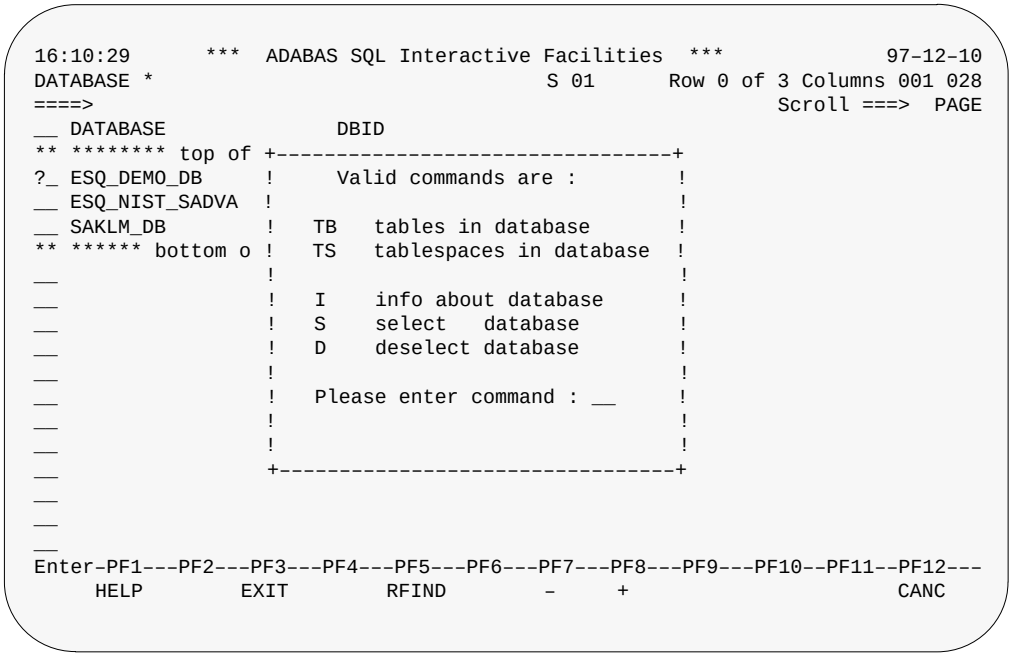
The listings of tables displayed as a result of the “TB” command can be used for further processing, whereas the contents of the screens displayed as a result of the “TS” or “I” command are for information purposes only.

Commands “TB” and “TS” can also be used as main commands. Main commands are entered in the command line of the database list screen and apply to all databases previously selected with the “S” line command.

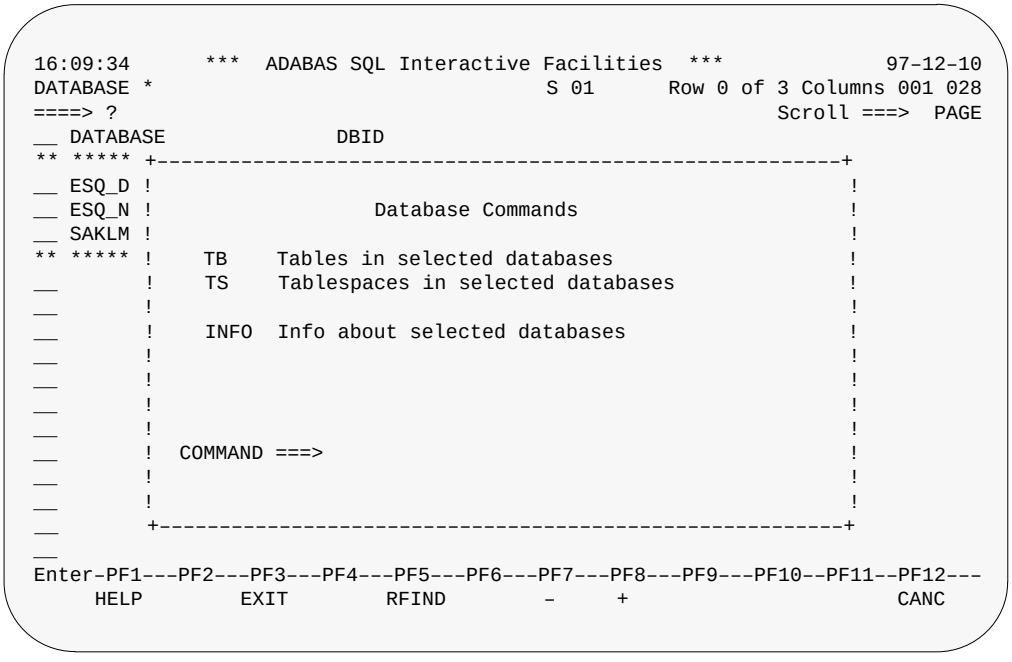
A further main command is the INFO command, which is the equivalent of the “I” line command, but displays information on all previously selected databases. Instead of being displayed, all information resulting from the “I” or INFO commands can also be marked for printing. Even if already displayed, information can be printed by issuing the PRINT command.

```
16:07:57      ***  ADABAS SQL Interactive Facilities  ***           97-12-10
DATABASE *                                  S 01      Row 0 of 3 Columns 001 028
====> info                                                Scroll ===>  PAGE
__ DATABASE              DBID
** **** +-----------------------------------------------------------+
__ ESQ_ !                                                           !
__ ESQ_ !                  Select what to display                   !
__ SAKL !                                                           !
** **** !                                                           !
__       !                                                          !
__       !                   _ tablespaces in database              !
__       !                   _ tables      in database              !
__       !                                                          !
__       !                                                          !
__       !                                                          !
__       !                  Mark  _ to print output                 !
__       !                                                          !
__       +-----------------------------------------------------------+
__
__
__
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
     HELP        EXIT        RFIND        -     +                     CANC
```

A list of all line commands available with the List Database function can be invoked by entering the help character “?” in front of any of the listed databases.



A list of all main commands available with the List Database function can be invoked by entering “?” in the command line of the database list screen.



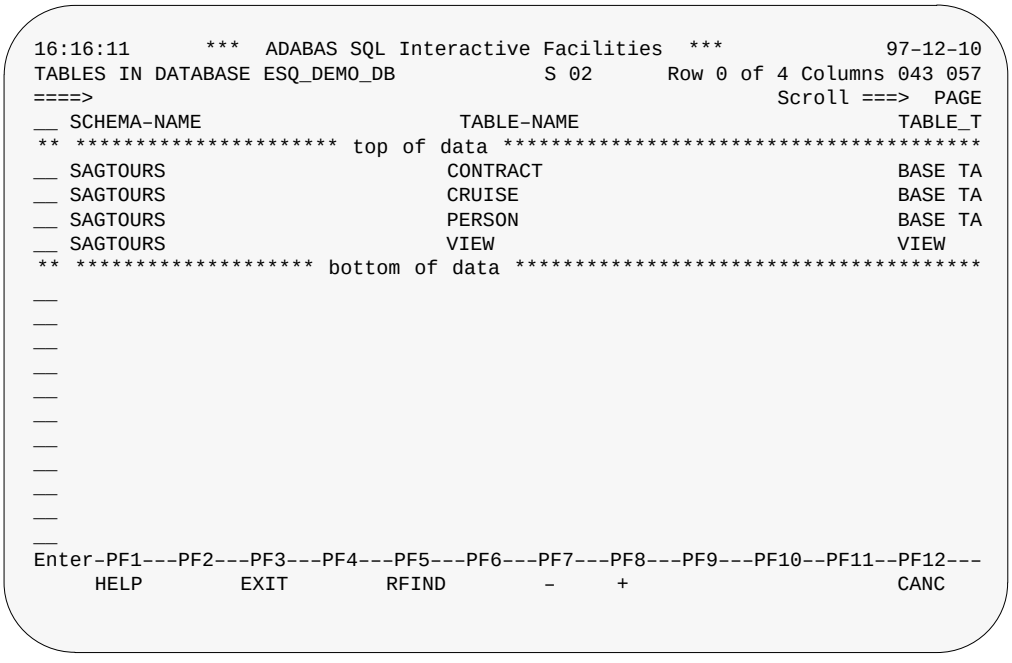
List Tablespaces

Since the function to list tablespaces is not part of the Catalog Retrieval main menu, tablespaces can be listed using the “TS” command issued on the database listing screen only. If you do so, a tablespace listing screen is displayed.

```
15:58:27      ***  ADABAS SQL Interactive Facilities  ***           97-12-11
TABLESPACES IN DATABASE ESQ_DEMO_DB       S 02      Row 0 of 4 Columns 043 078
====>                                                 Scroll ===>  PAGE
TABLE-SPACE-NAME                          FILENAME            FILE    DATABASE-
******************************** top of data ********************************
SAGTOURS.CONTRACT                         CONTRACT            221     ESQ_DEMO_
SAGTOURS.CRUISE                           CRUISE              220     ESQ_DEMO_
SAGTOURS.PERSON                           PERSON              222     ESQ_DEMO_
SAGTOURS.TEST                             TEST                241     ESQ_DEMO_
****************************** bottom of data ******************************









Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
     HELP        EXIT        RFIND        -     +           <     >    CANC
```

List Tables

To invoke the List Tables function, enter function code “T” on the Catalog Retrieval screen. The schema identifier and identifier of the table(s) to be listed can be specified. If a value followed by an asterisk is specified, all tables defined in the catalog whose qualifiers begin with that value are listed. If the qualifier/identifier fields are left blank, or if asterisk notation is specified only, all tables defined in the catalog are listed.



```
16:16:11      ***  ADABAS SQL Interactive Facilities  ***             97-12-10
TABLES IN DATABASE ESQ_DEMO_DB            S 02      Row 0 of 4 Columns 043 057
====>                                                 Scroll ===>  PAGE
__ SCHEMA-NAME                      TABLE-NAME                          TABLE_T
** ********************* top of data ***************************************
__ SAGTOURS                         CONTRACT                            BASE TA
__ SAGTOURS                         CRUISE                              BASE TA
__ SAGTOURS                         PERSON                              BASE TA
__ SAGTOURS                         VIEW                                VIEW
** ******************** bottom of data ***************************************
__
__
__
__
__
__
__
__
__
__
__
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
     HELP        EXIT        RFIND        -     +                      CANC
```

Commands Allowed on Tables

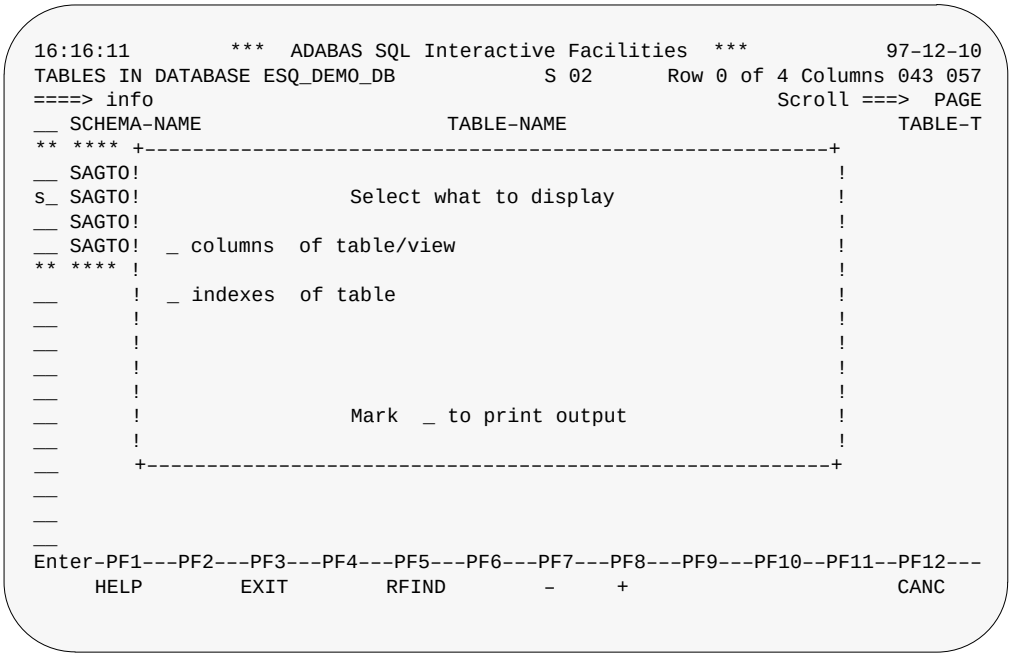
The following line commands are available on the table listing screen. Line commands are entered in front of the desired table(s):

Command	Description
I	Displays information about a table.
S	Selects a table to be used with main commands.
D	Deselects a table.
CO	Displays all columns of a table.
IX	Displays all indexes on a table.

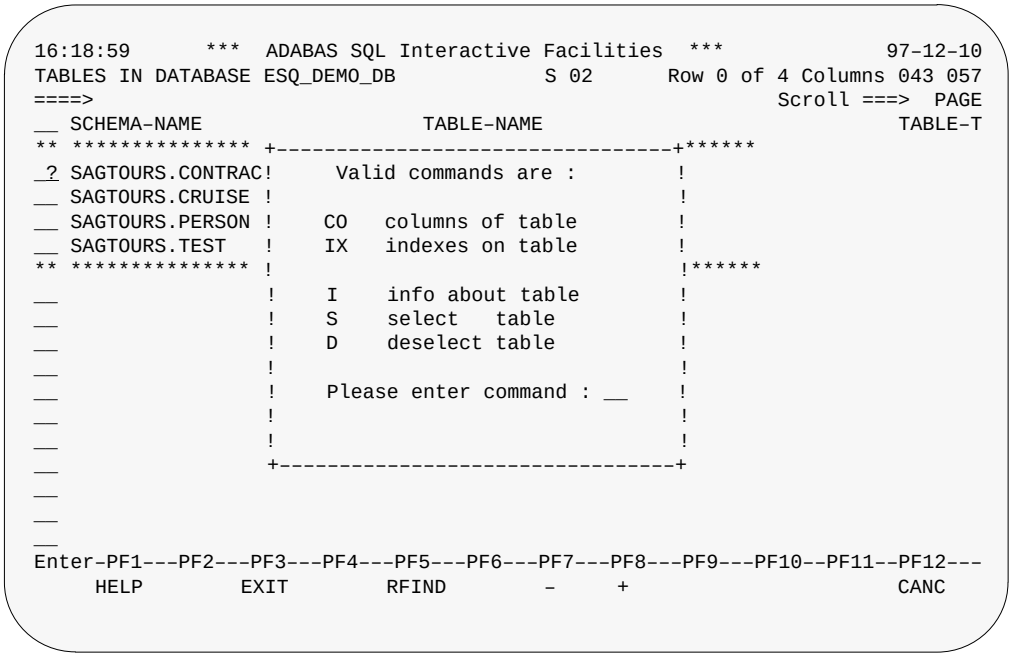
The listings displayed as a result of “IX”, “CO” and “I” are for information purposes only.

Commands “CO”, and “IX” can also be used as main commands, which are entered in the command line of the table listing screen and apply to all tables previously selected with the “S” line command.

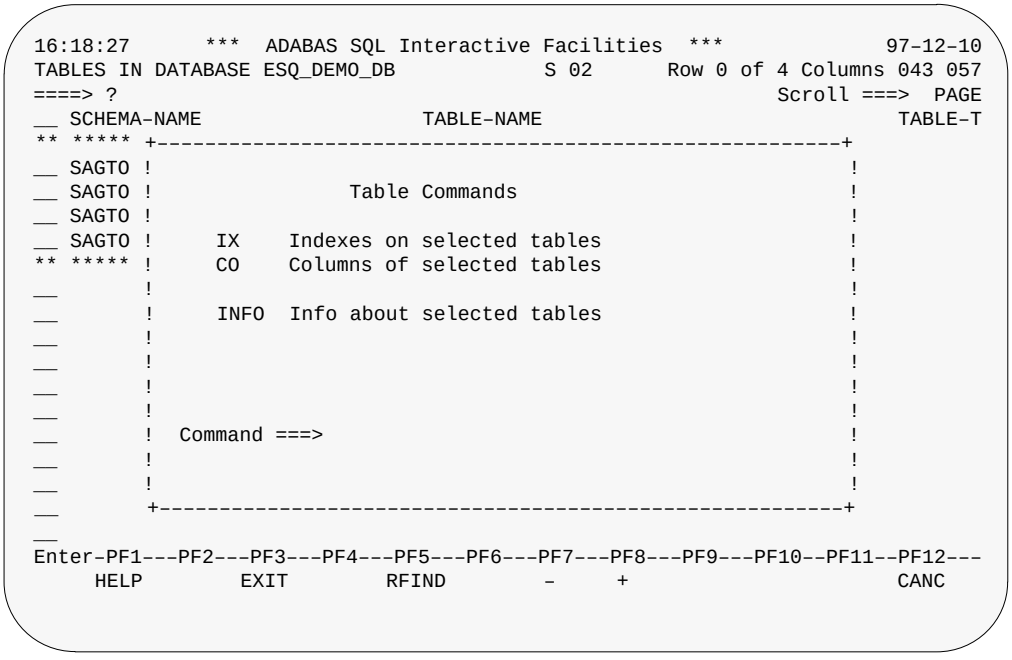
The INFO main command, which is the equivalent of the “I” line command, displays information on all tables previously selected. All information resulting from the “I” or INFO commands can also be printed.



A list of all line commands available with the List Tables function can be invoked by entering the help character "?" in front of any of the listed tables or pressing PF1 when the cursor is positioned in front of any of the listed tables.



A list of all main commands available with the List Tables function can be invoked by entering the help character “?” in the command line or pressing PF1.



The Migration Utility

Target Users

The Migration Utility is to be used as an aid migrating the contents of an Adabas SQL Server Version 1.3 directory to an SQL catalog as supported by Adabas SQL Server Version 1.4.

Overview

Despite the extensive increase in functionality offered by Version 1.4 of Adabas SQL Server, in general, upward compatibility is assured. However, certain steps are necessary in order to ensure a smooth migration. These steps are supported by the provision of a Migration Utility. Migration primarily concerns itself with the problems posed by a radically new catalog structure. The new catalog complies with the 1992 ANSI SQL Standard and is not compatible with the directory structure contained in Adabas SQL Server Version 1.3.

The Migration Utility takes the the 1.3 directory contents as input and produces a file containing the generated DDL statements (in the following referred to as ‘generated file’). The Migration Utility then starts the BASIC Interactive Facilities to process the contents of this generated file.

Although the catalog must be established completely afresh, the underlying user data, contained in the target Adabas files, can remain undisturbed during the process of migration. We recommend that a backup of all data is made prior to migration.

It may, however, not be necessary to use the services of the Migration Utility. If the contents of the existing directory are well documented and all create table description statements and all create view statements have been saved, then this is effectively equivalent to the output of the Migration Utility. However, the three concepts of users, schemas and security must still be considered (see below). In addition, as CREATE TABLE DESCRIPTION statements are now actively checked for accuracy, it may be, in isolated cases, that what was previously permitted is now rejected.

Once migration has been completed, the old 1.3 server must be removed. It is not recommended to run a 1.4 server and a 1.3 server in parallel, serving the same set of Adabas target files.

Prerequisites

- A prerequisite for the use of the Migration Utility is an installed Adabas SQL Server Version 1.4. It is strongly recommended that the new catalog is empty prior to migration to avoid conflicting situations.
- Another prerequisite is that the 1.3 directory file and 1.4 target catalog files reside in the same Adabas database.

New Concepts

Due to the various new concepts, it may be necessary to provide manual input to the generated file. In particular, the following issues must be considered:

- A user concept. It is now necessary to introduce the concept of authenticated users. All users of Adabas SQL Server must be made known using the CREATE USER statement.
- A schema concept. Data structures are “contained” in schemata. A schema implies ownership.
- A security concept. If the server has been installed with the security option, then privileges must be granted to users as required.

Special Considerations

In general, pre-compiled applications will not require modification or indeed re-building in any way. However, non-upwardly compatible features will result in the need for modification:

- Existing identifiers now equivalent to a new keyword. New keywords have been introduced. If an existing identifier is the same as one of these keywords, then the identifier must be modified.
- DDL and DML statements mixed within the same transaction. Such statements must be in separate transactions.
- Certain DDL statements may not be upwardly compatible. These are:
  CREATE DATABASE (if host variable was used)
  CREATE TABLE DESCRIPTION

Invoking the Migration Utility

To invoke the Migration Utility, use the following example JCL or member **J#MIG.J** in the Adabas SQL Server sublibrary:

```
* $$ JOB  JNM=ESQMIG,CLASS=0,DISP=D
* $$ LST  CLASS=A,DISP=D
// JOB ESQMIG --- EXECUTE ESQMIG UTILITY ---
/*
/* ------------------------------------------------------------------ *
/* ESQ Migration Utility (ESQMIG)
/* ------------------------------------------------------------------ *
/* Migrate an ADABAS SQL Server Version 1.3 Directory to a
/* Version 1.4 Catalog.
/* ------------------------------------------------------------------ *
/*
// LIBDEF  PHASE,SEARCH=(SAGLIB.ESQ142S,                              *
              SAGLIB.ESQ142,                                        *
              SAGLIB.ADA621,                                        *
              SORTLIB.SORT),TEMP
// DLBL    ESQMIGE,'SYSLST'
// DLBL    ESQMIGO,'SAGLIB.ESQ142(ESQMIGO.D)'
// DLBL    ESQMIGT,'SYSLST'
// DLBL    ESQPARM,'SAGLIB.ESQ142(P#ESQRUN.D)'
// DLBL    ESQPRIN,'SYSLST'
// DLBL    ESQWORK,'ESQ.SINGLE.ESQWORK',,VSAM,CAT=VSESPUC
// DLBL    SYSIN,'SYSRDR'
// DLBL    SYSOUT,'SYSLST'
// DLBL    SYSPRIN,'SYSLST'
/*
/* ---- ESQ Sortfile datasets ------------------------------------ *
/*
// ASSGN   SYS021,DISK,VOL=ESQ001,SHR
// ASSGN   SYS022,DISK,VOL=ESQ001,SHR
// ASSGN   SYS023,DISK,VOL=ESQ001,SHR
// DLBL    S001WK1,'ESQ.SINGLE.SORTWK1',0,SD
// EXTENT  SYS021,ESQ001,1,0,150,150
// DLBL    S001WK2,'ESQ.SINGLE.SORTWK2',0,SD
// EXTENT  SYS022,ESQ001,1,0,300,150
// DLBL    S001WK3,'ESQ.SINGLE.SORTWK3',0,SD
// EXTENT  SYS023,ESQ001,1,0,450,150
/*
```

```
// EXEC    ESQMIG,SIZE=AUTO,PARM='fnr dbid [-v] [-t] [-n] [-s] [-h]'
ADARUN     DATABASE=36
ADARUN     DEVICE=8381
ADARUN     MODE=MULTI
ADARUN     PROGRAM=USER
ADARUN     SVC=45
/*
// EXEC    LISTLOG
/&
* $$ EOJ
```

The Migration Utility can be invoked using the following syntax:

```
// EXEC ESQMIG,SIZE=AUTO,PARM='fnr dbid [-v] [-t] [-n] [-s] [-h]'
```

where

fnr	File number of the Adabas SQL Server 1.3 directory
dbid	Database ID of the Adabas SQL Server 1.3 directory
–v	no generattion of VIEWs
–t	Generation of CREATE TABLE and CREATE TABLESPACE statements
–n	No generation of CREATE TABLE and CREATE TABLESPACE statements
–s	No generation of CREATE TABLESPACE statements (only valid in combination with the –t option)
–h	Prepare for semi-automatic migration (recommended)

Migration

If the new concepts of Adabas SQL Server Version 1.4 are to be used, it is necessary to postprocess the generated file **ESQMIGO** in a second step before submitting it to the BASIC Interactive Facilities for execution.

For each user, a CREATE USER statement is to be generated and for the schemas to be created, the appropriate authorization is to be specified.

If further databases, other than the default database, are already defined in the catalog, it may also be necessary to postprocess the generated file. This is particularly true, if table descriptions existed in the version 1.3 specifying a database that is already defined in Adabas SQL Server Version 1.4. In this case, the generated CREATE DATABASE statements are to be adapted by hand.

1 Invoke the Migration Utility to establish the generated file.

2 Adapt the file to reflect the required data structures. Adaptations are necessary for the following three cases:

- **Case I**: If you want to use the owner concept for schemas, perform the command esqmigrate with the –h option and a file is generated with the following structure:

```
# !!!!!!!!!!!! START PART1 !!!!!!!!!!!!!
DBA
# ******************************************
# * REPLACE THE FOLLOWING ??? BY A USER NAME  *
# * AND A PASSWORD SUCCESSIVELY!              *
# ******************************************
# CREATE USER ??? PASSWORD ???;
# .
# .
# CREATE USER ??? PASSWORD ???;
# !!!!!!!!!!!! END   PART1 !!!!!!!!!!!!!
# !!!!!!!!!!!! START PART2 !!!!!!!!!!!!!
# DBA
# ***************************************
# * REPLACE THE FOLLOWING ??? BY THE USER *
# * WHO WILL BE THE OWNER OF THE SCHEMA!  *
# ***************************************
# CREATE SCHEMA schema1 AUTHORIZATION ???;
CREATE SCHEMA schema1;
# !!!!!!!!!!!! END   PART2 !!!!!!!!!!!!!
# !!!!!!!!!!!! START PART3 !!!!!!!!!!!!!
# ************************************
# * REPLACE THE FOLLOWING ??? BY THE    *
# * OWNER OF THE SCHEMA schema1!        *
# ************************************
#
CREATE DATABASE...;
CREATE TABLE DESCRIPTION schema1.table1...;
CREATE TABLE DESCRIPTION schema1.tablen...;
# !!!!!!!!!!!! END   PART3 !!!!!!!!!!!!!
# !!!!!!!!!!!! START PART4 !!!!!!!!!!!!!
DBA
# ***************************************
# * REPLACE THE FOLLOWING ??? BY THE USER *
# * WHO WILL BE THE OWNER OF THE SCHEMA!  *
# ***************************************
# CREATE SCHEMA schema2 AUTHORIZATION ???;
  CREATE SCHEMA schema2;
```

```
  # !!!!!!!!!!!! END   PART4 !!!!!!!!!!!!
  # !!!!!!!!!!!! START PART5 !!!!!!!!!!!!
  # ************************************
  # * REPLACE THE FOLLOWING ??? BY THE    *
  # * OWNER OF THE SCHEMA schema2!        *
  # ************************************
  CREATE TABLE DESCRIPTION schema2.table1...;
  ...
CREATE TABLE DESCRIPTION schema2.tablen...;
  # !!!!!!!!!!!! END   PART5 !!!!!!!!!!!!
```

Edit the file as follows:

- copy each part (all between START PARTx – END PARTx) into a file (for example: filex),
- change ”???” to user name or password as indicated in the comment,
- delete the comment signs and CREATE SCHEMA statements where needed.
- call the BASIC Interactive Facilities (ESQBIF) with filex as input.

- **Case II**: If you want to to use database names other than those generated:
  - generating an output file using the command esqmigrate with the –h option.
  - change the database names in the generated CREATE DATABASE statements from old name to new name.
  - replace each old name by a new name in the CREATE TABLE DESCRIPTION statements.
  - calling the BASIC Interactive Facilities (ESQBIF) with file1 as input.

- **Case III**: There are table and/or column names in the version 1.3 directory, which are keywords in the version 1.4. Delimited identifiers are not supported in version 1.4, therefore, the following steps have to be taken:
  - look into the error file esqmig.err and search for warnings: ESQMIG–W–KEYWORD: ”Column/Table name int is also a keyword”
  - if there are warnings of this kind, replace the names in question by new names that are not keywords in Adabas SQL Server Version 1.4
  - Call the BASIC Interactive Facilities (ESQBIF) using the generated file as input.

  Note:
  Make sure to change your applications to reflect the change of identifiers.

Generate Table Description Utility

Target User

Anyone who wants to introduce an existing Adabas file to 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 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 for execution, thus introducing the existing Adabas file to Adabas SQL Server.

The GTD Utility is controlled using an Input File.

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

- The Adabas FDT of the file to be described (mandatory).
- Default rules.
- The input file content.
- A SYSTRANS file.

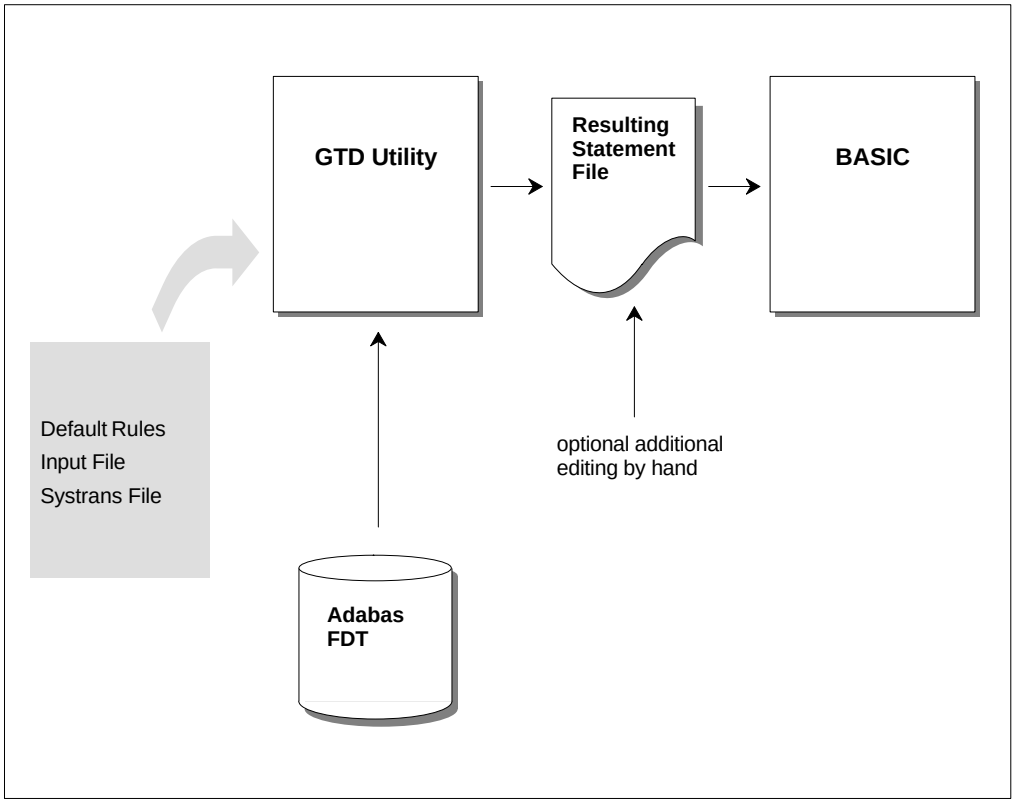


Figure 5-1: GTD information sources

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

The resultant statement output file is not automatically presented to the Basic Interactive Utility. There is always the opportunity to review and edit the generated output file.

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

To invoke the GTD Utility, use the following example JCL or member **J#GTD.J** in the Adabas SQL Server sublibrary:

```
* $$ JOB  JNM=ESQGTD,CLASS=0,DISP=D
* $$ LST  CLASS=A,DISP=D
// JOB ESQGTD --- EXECUTE ESQGTD UTILITY ---
/*
/* ------------------------------------------------------------- *
/* ESQ Generate Table Description Utility (ESQGTD)
/* ------------------------------------------------------------- *
/*
// LIBDEF  PHASE,SEARCH=(SAGLIB.ESQ142S,                          *
              SAGLIB.ESQ142,                                     *
              SAGLIB.ADA621),TEMP
// DLBL    ESQGTDO,'SAGLIB.ESQ142(ESQGTDO.D)'
// ASSGN   SYS011,DISK,VOL=ESQ001,SHR
// DLBL    ESQGTDS,'SYS011,BLKSIZE=6240,LRECL=96,RECFM=FB'
// DLBL    SYS011,'ESQ.ESQGTDS.FILE007',0,SD
// EXTENT  SYS011,ESQ001,1,0,15,15
// DLBL    ESQPARM,'SAGLIB.ESQ142(P#ESQRUN.D)'
// DLBL    ESQPRIN,'SYSLST'
// DLBL    SYSIN,'SYSRDR'
// DLBL    SYSOUT,'SYSLST'
// DLBL    SYSPRIN,'SYSLST'
// EXEC    ESQGTD,SIZE=AUTO
SYSTRANS
DATABASE   198 VSE_PRODUCTION
FILE       007 DDM=PAYROLL
/*
ADARUN     DATABASE=36
ADARUN     DEVICE=8381
ADARUN     MODE=MULTI
ADARUN     PROGRAM=USER
ADARUN     SVC=45
/*
// EXEC    LISTLOG
/&
* $$ EOJ
```

where:

ESQGTDO	the DLBL name indicates a file which will contain the resultant generated statement
ESQGTDS	the DLBL name indicates a file which contains a Natural SYSTRANS FILE (if required). This file must have been created using the Natural SYSTRANS Utility defaults; i.e. RECFM=FB,LRECL=96,BLKSIZE=6240.
SYSIN	the DLBL name indicates a file which contains the user input file.

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.

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.

User Input Control File

The User Input Control File is the most detailed source of information available to the utility. All aspects of the generation can be controlled from this file. General aspects like file number and dbid, DDM file specification, schema identifier, cluster and table identifier can be specified.

The User Input Control File is provided by means of a specification file. This is a file which must conform to the syntax as specified below and provides additional information to the utility.

In addition to the general information, more specific column attributes can be specified. In particular, attributes for columns which deviate from the default can be specified here.

Most Adabas fields can be directly mapped to a simple SQL column without the necessity for further user clarification. However, MU and PE fields do require additional clarification, as they can be interpreted in various distinct ways, which usually cannot be otherwise deduced.

An MU field can be one of the following:

- A rotated field
- A longalpha field, if it is of type character
- A suppressed field
- A subtable or part thereof.

A PE group can be one of the following:

- A suppressed group (i.e. ignored)
- A subtable.

A member of a suppressed PE group can be one of the following:

- A rotated field, as long as it is not an MU field
- A suppressed field.

A member of a non-suppressed PE group can be one of the following:

- A standard column of the subtable, as long as it is not an MU field
- A rotated field
- A longalpha field, if it is of type character
- A suppressed field
- A subtable or part thereof.

A rotated field must have a specified fixed number of occurrences, where each occurrence is mapped to an explicit SQL column. For example, if semantically the field can have twelve occurrences (representing 12 months of a year), then 12 separate columns (January through December) will be created. The number of occurrences must be between 1 and 191.

A longalpha field is where a number of MU occurrences of type character are concatenated to form a logical SQL column of type character. Again the number of characters must be specified. The length must not be greater then 16381 characters.

The above two options do not require the use of a cluster description. However when MU/PE fields are to be interpreted as subtables, then the file must be described in terms of a table cluster. Should no information to the contrary be provided for an MU or PE field, then the default operation is to build a subtable. By default the specification of the primary and foreign key relationship is assumed to be simply the appropriate sequence number. However this relationship can also be more precisely specified if desired.

By default, the utility will always produce a cluster description, where the subtables contain just one MU field. Adabas SQL Server itself can support subtables which consist of more i.e. parallel MUs. Again the combination of two or more MU fields into one subtable can be specified within the syntax.

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 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.

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.

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.

Input File

The database identifier is specified using 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.

Input File

The file number is specified using 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 using 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.

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.

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

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

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

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

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 using 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.

Input File

A column identifier is explicitly specified using 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.

Input File

A rotated field column identifier is explicitly specified using 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 using 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.

Input File

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

Input File

Fields can be explicitly suppressed using 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 using 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
```

The SYSTRANS file is assigned to the GTD Utility using the DLBL ESQGTDS in the JCL.

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 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.
4 Warning generated
8 Error detected.

The error text can be found in SYSOUT.

6

LOGGING FACILITIES

Introduction

Numerous actions and operations performed by 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 therefore 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. Logging has to be specified on 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.

Logging facilities are always available, even in the non-trace version. However, their use 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
Full 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
Security Logging	
User authentication check logging	AUTH
Simple statement logging	SEC
ALL resultant actions logged	SEC=ALL
ESQLNK Call Logging	ENTRY
Explain Logging	EXPL

Adabas Command Logging

How to activate Logging

Logging Facilities Activated by Parameter File Specification

The activation of Adabas Command Logging is an exception to this rule and the handling is the same for all supported platforms. A detailed description of how to activate Adabas Command Logging is found in the section with the same heading, later in this chapter.

Logging Facilities Activated by Environment Variables (ESQLOG)

Place the following macro entries in the VOPRM module, then assemble and link it. Output is written to the dataset.

```
VOENV   NAME=ESQLOG,
        VALUE='CMD,CSC'
```

turns on the brief SQL command logging and the client/server communication logging.

Logging Facilities Reference

The following chapters describe the available 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 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 with ESQLNK

Example:

```
 VOENV  NAME=ESQLOG,VALUE='MODE'

 SAGESQ-I-ESQINFO, ESQLOG is enabled for 'MODE'
 SAGESQ-I-ESQINFO, MT/TR trace is NOT available
 SAGESQ-I-ESQINFO,
 SAGESQ-I-ESQINFO,
 SAGESQ-I-ESQINFO, Th0: Single-user linked-in mode

%ESQINT-I-STARTED, 02-SEP-1997 12:53:50, Version 1.4/2.7 (Mainframe/VSE/DESA)
%ESQINT-I-BATMODE, Running in batch mode

ESQ User:

 DBA
Password:

%ESQINT-I-CONNECT, Server LINKEDIN connected successfully

  SELECT * FROM information_schema.tables;
  SELECT * FROM information_schema.tables;

 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

 CANWEPARSELENGTH18              CHARACTER18TABLE18              BASE TABLE
 CANWEPARSELENGTH18              CHARACTERS18VIEW18              VIEW
---EOF---
 output (col 1-107 of 107):

 QUIT;
%ESQINT-I-TERMINATED, 02-SEP-1997 12:53:56
```

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:

```
VOENV NAME=ESQLOG,VALUE='CMD'

SAGESQ-I-ESQINFO, ESQLOG is enabled for 'CMD'
02-SEP 14:52:02.03 Th0: CONNECT         cnt=   1/   1 usr=MAINFRAM:DBA rsp=0
SAGESQ-I-ESQINFO, Clt: CONNECT         cnt=   1/   2 rsp=0
02-SEP 14:52:02.83 Th0: PREPARE        -cnt=   7/   2 usr=MAINFRAM:DBA rsp=0
SAGESQ-I-ESQINFO, Clt: PREPARE         cnt=   7/   3 rsp=0
02-SEP 14:52:02.88 Th0: DESCRIBE       -cnt=   8/   3 usr=MAINFRAM:DBA rsp=0
SAGESQ-I-ESQINFO, Clt: DESCRIBE        cnt=   8/   4 rsp=0
02-SEP 14:52:02.92 Th0: DESCRIBE       -cnt=   9/   4 usr=MAINFRAM:DBA rsp=0
SAGESQ-I-ESQINFO, Clt: DESCRIBE        cnt=   9/   5 rsp=0
02-SEP 14:52:03.96 Th0: DYN DECL CURSO -cnt=  10/   5 usr=MAINFRAM:DBA rsp=0
SAGESQ-I-ESQINFO, Clt: DYN DECL CURSO  cnt=  10/   6 rsp=0
02-SEP 14:52:03.99 Th0: OPEN CURSOR    -cnt=  11/   6 usr=MAINFRAM:DBA rsp=0
SAGESQ-I-ESQINFO, Clt: OPEN CURSOR     cnt=  11/   7 rsp=0
02-SEP 14:52:03.11 Th0: mFETCH (rf=16)  cnt=  12/   7 usr=MAINFRAM:DBA rsp=0
SAGESQ-I-ESQINFO, Clt: mFETCH (rf=16)  cnt=  12/   8 rsp=0
SAGESQ-I-ESQINFO, Clt: sFETCH (rp=2)   cnt=  13/   8 rsp=0
SAGESQ-I-ESQINFO, Clt: sFETCH (rp=3)   cnt=  14/   8 rsp=0
SAGESQ-I-ESQINFO, Clt: sFETCH (rp=4)   cnt=  15/   8 rsp=0
SAGESQ-I-ESQINFO, Clt: sFETCH (rp=5)   cnt=  16/   8 rsp=0
SAGESQ-I-ESQINFO, Clt: sFETCH (rp=6)   cnt=  17/   8 rsp=0
SAGESQ-I-ESQINFO, Clt: sFETCH (rp=7)   cnt=  18/   8 rsp=0
SAGESQ-I-ESQINFO, Clt: sFETCH (rp=8)   cnt=  19/   8 rsp=0


02-SEP 14:52:10.76 Th0: DISCONNECT      cnt= 187/  19 usr=MAINFRAM:DBA rsp=0
SAGESQ-I-ESQINFO, Clt: DISCONNECT      cnt= 187/  20 rsp=0
```

```
%ESQINT-I-STARTED, 02-SEP-1997 14:51:53, Version 1.4/2.7 (Mainframe/VSE/DESA)
%ESQINT-I-BATMODE, Running in batch mode

ESQ User:

 DBA
Password:

%ESQINT-I-CONNECT, Server LINKEDIN connected successfully

  SELECT * FROM information_schema.tables;
  SELECT * FROM information_schema.tables;

 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

                                                                      TABLE
 CANWEPARSELENGTH18              CHARACTER18TABLE18              BASE TABLE
 CANWEPARSELENGTH18              CHARACTERS18VIEW18              VIEW
---EOF---
 output (col 1-107 of 107):

  QUIT;
%ESQINT-I-TERMINATED, 02-SEP-1997 14:52:10
```

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
- Continuous Stream Format

Example:

```
VOENV NAME=ESQLOG,VALUE='STATEMENT'

SAGESQ-I-ESQINFO, ESQLOG is enabled for 'STATEMENT'
SAGESQ-I-ESQINFO, CONNECT TO DEFAULT USER : "&sql_user 0 "
CHARACTER(,32,0,0,8,33)
SAGESQ-I-ESQINFO, PREPARE : "&sql_id 0 " CHARACTER(,32,0,0,46,33)
FROM : "&small_sql_statement 0 " CHARACTER(,128,0,0,47,129)
SAGESQ-I-ESQINFO, DESCRIBE : "&sql_id 0 " CHARACTER(,32,0,0,55,33)
INTO INPUT: "if_sql_input_sqlda" ( "if_sql_input_sqlda" POINTER ( ,0,0,0,0))
SAGESQ-I-ESQINFO, DESCRIBE : "&sql_id 0 " CHARACTER(,32,0,0,56,33)
INTO OUTPUT: "if_sql_output_sqlda" ( "if_sql_output_sqlda" POINTER
( ,0,0,0,1))
SAGESQ-I-ESQINFO, DECLARE : "&sql_cursor 0 " CHARACTER(,18,0,0,87,19)
CURSOR FOR : "&sql_id 0 " CHARACTER(,32,0,0,88,33)
SAGESQ-I-ESQINFO, OPEN : "&sql_cursor 0 " CHARACTER(,18,0,0,89,19)
SAGESQ-I-ESQINFO, FETCH : "&sql_cursor 0 " CHARACTER(,18,0,0,91,19)
USING DESCRIPTOR : "if_sql_output_sqlda" ( "if_sql_output_sqlda" POINTER
( ,0,0,0,1))
SAGESQ-I-ESQINFO, FETCH : "&sql_cursor 0 " CHARACTER(,18,0,0,91,19)
USING DESCRIPTOR : "if_sql_output_sqlda" ( "if_sql_output_sqlda" POINTER
( ,0,0,0,1))
SAGESQ-I-ESQINFO, FETCH : "&sql_cursor 0 " CHARACTER(,18,0,0,91,19)
USING DESCRIPTOR : "if_sql_output_sqlda" ( "if_sql_output_sqlda" POINTER
( ,0,0,0,1))
SAGESQ-I-ESQINFO, FETCH : "&sql_cursor 0 " CHARACTER(,18,0,0,91,19)
USING DESCRIPTOR : "if_sql_output_sqlda" ( "if_sql_output_sqlda" POINTER
( ,0,0,0,1))
```

```
 SAGESQ-I-ESQINFO, FETCH : "&sql_cursor 0 " CHARACTER(,18,0,0,91,19)
 USING DESCRIPTOR : "if_sql_output_sqlda" ( "if_sql_output_sqlda" POINTER
 ( ,0,0,0,1))
 SAGESQ-I-ESQINFO, FETCH : "&sql_cursor 0 " CHARACTER(,18,0,0,91,19)
 USING DESCRIPTOR : "if_sql_output_sqlda" ( "if_sql_output_sqlda" POINTER
 ( ,0,0,0,1)) NG DESC ,0,0,1))
 SAGESQ-I-ESQINFO, CLOSE : "&sql_cursor 0 " CHARACTER(,18,0,0,90,19)
 SAGESQ-I-ESQINFO, DISCONNECT CURRENT

%ESQINT-I-STARTED, 02-SEP-1997 15:44:39, Version 1.4/2.7 (Mainframe/VSE/DESA)
%ESQINT-I-BATMODE, Running in batch mode

ESQ User:

 DBA
Password:

%ESQINT-I-CONNECT, Server LINKEDIN connected successfully

  SELECT * FROM information_schema.tables;
  SELECT * FROM information_schema.tables;

 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

 CANWEPARSELENGTH18              CHARACTER18TABLE18              BASE TABLE
 CANWEPARSELENGTH18              CHARACTERS18VIEW18              VIEW
---EOF---
 output (col 1-107 of 107):

  QUIT;
%ESQINT-I-TERMINATED, 02-SEP-1997 15:45:21
```

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
- Continuous Stream Format

Example:

```
VOENV NAME=ESQLOG,VALUE='DYNHVS'

 SAGESQ-I-ESQINFO, ESQLOG is enabled for 'DYNHVS'
 SAGESQ-I-ESQINFO, HV01: sql_i
 SAGESQ-I-ESQINFO, HV02:  SELECT * FROM information_schema.table
 IEF142I GEBBIF ESQVER - STEP WAS EXECUTED - COND CODE 0000

%ESQINT-I-STARTED, 02-SEP-1997 17:04:13, Version 1.4/2.7 (Mainframe/VSE/DESA)
%ESQINT-I-BATMODE, Running in batch mode

ESQ User:

 DBA
Password:

%ESQINT-I-CONNECT, Server LINKEDIN connected successfully

  SELECT * FROM information_schema.tables;
  SELECT * FROM information_schema.tables;

 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


 CANWEPARSELENGTH18              CHARACTER18TABLE18              BASE TABLE
 CANWEPARSELENGTH18              CHARACTERS18VIEW18              VIEW
---EOF---
 output (col 1-107 of 107):

  QUIT;
%ESQINT-I-TERMINATED, 02-SEP-1997 17:04:24
```

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 one 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:

```
VOENV NAME=ESQLOG,VALUE='CSCI'

 SAGESQ-I-ESQINFO, ESQLOG is enabled for 'CSCI'
 SAGESQ-I-ESQINFO, Clt: CSCI ident 1/1
 SAGESQ-I-ESQINFO, Clt: sending an OPEN to GEBNET:ESQ1569 ...
 SAGESQ-I-ESQINFO, Clt: ONN cid=0 sndl=0 rcvl=0 retl=0
 SAGESQ-I-ESQINFO, Clt: ONN cid=1 sndl=0 rcvl=0 retl=0
 SAGESQ-I-ESQINFO, Clt: ... OPEN was successful
 SAGESQ-I-ESQINFO, Clt: sending a CSCI request ...
 SAGESQ-I-ESQINFO, Clt: QNS cid=1 sndl=3596 rcvl=338 retl=0
 SAGESQ-I-ESQINFO, Clt: QNS cid=1 sndl=3596 rcvl=338 retl=338
 SAGESQ-I-ESQINFO, Clt: ... received CSCI reply
 SAGESQ-I-ESQINFO, Clt: sending a CSCI request ...
 SAGESQ-I-ESQINFO, Clt: QNS cid=1 sndl=432 rcvl=342 retl=338
 SAGESQ-I-ESQINFO, Clt: QNS cid=1 sndl=432 rcvl=342 retl=342
 SAGESQ-I-ESQINFO, Clt: ... received CSCI reply
 SAGESQ-I-ESQINFO, Clt: sending a CSCI request ...
 SAGESQ-I-ESQINFO, Clt: QNS cid=1 sndl=306 rcvl=1532 retl=342
 SAGESQ-I-ESQINFO, Clt: QNS cid=1 sndl=306 rcvl=1532 retl=1532
 SAGESQ-I-ESQINFO, Clt: ... received CSCI reply
 SAGESQ-I-ESQINFO, Clt: sending a CSCI request ...
 SAGESQ-I-ESQINFO, Clt: QNS cid=1 sndl=306 rcvl=1532 retl=1532
 SAGESQ-I-ESQINFO, Clt: QNS cid=1 sndl=306 rcvl=1532 retl=1532
 SAGESQ-I-ESQINFO, Clt: ... received CSCI reply
 SAGESQ-I-ESQINFO, Clt: sending a CSCI request ...
 SAGESQ-I-ESQINFO, Clt: QNS cid=1 sndl=304 rcvl=310 retl=1532
 SAGESQ-I-ESQINFO, Clt: QNS cid=1 sndl=304 rcvl=310 retl=310
 SAGESQ-I-ESQINFO, Clt: ... received CSCI reply
 SAGESQ-I-ESQINFO, Clt: sending a CSCI request ...
 SAGESQ-I-ESQINFO, Clt: QNS cid=1 sndl=264 rcvl=310 retl=310
 SAGESQ-I-ESQINFO, Clt: QNS cid=1 sndl=264 rcvl=310 retl=310
 SAGESQ-I-ESQINFO, Clt: ... received CSCI reply
 SAGESQ-I-ESQINFO, Clt: sending a CSCI request ...
 SAGESQ-I-ESQINFO, Clt: QNS cid=1 sndl=324 rcvl=2118 retl=310
 SAGESQ-I-ESQINFO, Clt: QNS cid=1 sndl=324 rcvl=2118 retl=2118
 SAGESQ-I-ESQINFO, Clt: ... received CSCI reply
 SAGESQ-I-ESQINFO, Clt: sending a CSCI request ...
 SAGESQ-I-ESQINFO, Clt: QNS cid=1 sndl=324 rcvl=2118 retl=2118
 SAGESQ-I-ESQINFO, Clt: QNS cid=1 sndl=324 rcvl=2118 retl=2118
 SAGESQ-I-ESQINFO, Clt: ... received CSCI reply
 SAGESQ-I-ESQINFO, Clt: sending a CSCI request ...
 SAGESQ-I-ESQINFO, Clt: QNS cid=1 sndl=324 rcvl=2118 retl=2118
 SAGESQ-I-ESQINFO, Clt: QNS cid=1 sndl=324 rcvl=2118 retl=2118
 SAGESQ-I-ESQINFO, Clt: ... received CSCI reply
 SAGESQ-I-ESQINFO, Clt: sending a CSCI request ...
```

```
 SAGESQ–I–ESQINFO, Clt: QNS cid=1 sndl=324 rcvl=2118 retl=2118
 SAGESQ–I–ESQINFO, Clt: QNS cid=1 sndl=324 rcvl=2118 retl=2118
 SAGESQ–I–ESQINFO, Clt: ... received CSCI reply
 SAGESQ–I–ESQINFO, Clt: sending a CSCI request ...
 SAGESQ–I–ESQINFO, Clt: QNS cid=1 sndl=324 rcvl=2118 retl=2118
 SAGESQ–I–ESQINFO, Clt: QNS cid=1 sndl=324 rcvl=2118 retl=2118
 SAGESQ–I–ESQINFO, Clt: ... received CSCI reply
 SAGESQ–I–ESQINFO, Clt: sending a CSCI request ...
 SAGESQ–I–ESQINFO, Clt: QNS cid=1 sndl=324 rcvl=2118 retl=2118
 SAGESQ–I–ESQINFO, Clt: QNS cid=1 sndl=324 rcvl=2118 retl=2118
 SAGESQ–I–ESQINFO, Clt: ... received CSCI reply
 SAGESQ–I–ESQINFO, Clt: sending a CSCI request ...
 SAGESQ–I–ESQINFO, Clt: QNS cid=1 sndl=324 rcvl=2118 retl=2118
 SAGESQ–I–ESQINFO, Clt: QNS cid=1 sndl=324 rcvl=2118 retl=2118
 SAGESQ–I–ESQINFO, Clt: ... received CSCI reply
 SAGESQ–I–ESQINFO, Clt: sending a CSCI request ...
 SAGESQ–I–ESQINFO, Clt: QNS cid=1 sndl=324 rcvl=2118 retl=2118
 SAGESQ–I–ESQINFO, Clt: QNS cid=1 sndl=324 rcvl=2118 retl=2118
 SAGESQ–I–ESQINFO, Clt: ... received CSCI reply
 SAGESQ–I–ESQINFO, Clt: sending a CSCI request ...
 SAGESQ–I–ESQINFO, Clt: QNS cid=1 sndl=324 rcvl=2118 retl=2118
 SAGESQ–I–ESQINFO, Clt: QNS cid=1 sndl=324 rcvl=2118 retl=2118
 SAGESQ–I–ESQINFO, Clt: ... received CSCI reply
 SAGESQ–I–ESQINFO, Clt: sending a CSCI request ...
 SAGESQ–I–ESQINFO, Clt: QNS cid=1 sndl=324 rcvl=2118 retl=2118
 SAGESQ–I–ESQINFO, Clt: QNS cid=1 sndl=324 rcvl=2118 retl=2118
 SAGESQ–I–ESQINFO, Clt: ... received CSCI reply
 SAGESQ–I–ESQINFO, Clt: sending a CSCI request ...
 SAGESQ–I–ESQINFO, Clt: QNS cid=1 sndl=324 rcvl=2118 retl=2118
 SAGESQ–I–ESQINFO, Clt: QNS cid=1 sndl=324 rcvl=2118 retl=2118
 SAGESQ–I–ESQINFO, Clt: ... received CSCI reply
 SAGESQ–I–ESQINFO, Clt: sending a CSCI request ...
 SAGESQ–I–ESQINFO, Clt: QNS cid=1 sndl=264 rcvl=310 retl=2118
 SAGESQ–I–ESQINFO, Clt: QNS cid=1 sndl=264 rcvl=310 retl=310
 SAGESQ–I–ESQINFO, Clt: ... received CSCI reply
 SAGESQ–I–ESQINFO, Clt: sending a CSCI request ...
 SAGESQ–I–ESQINFO, Clt: QNS cid=1 sndl=264 rcvl=310 retl=310
 SAGESQ–I–ESQINFO, Clt: QNS cid=1 sndl=264 rcvl=310 retl=310
 SAGESQ–I–ESQINFO, Clt: ... received CSCI reply
 SAGESQ–I–ESQINFO, Clt: sending a CLOSE ...
 SAGESQ–I–ESQINFO, Clt: CNS cid=1 sndl=0 rcvl=0 retl=310
 SAGESQ–I–ESQINFO, Clt: CNS cid=1 sndl=0 rcvl=0 retl=310
 SAGESQ–I–ESQINFO, Clt: ... CLOSE was successful


%ESQINT–I–STARTED, 02–SEP–1997 17:57:41, Version 1.4/2.7 (Mainframe/VSE/DESA)
%ESQINT–I–BATMODE, Running in batch mode
```

```
ESQ User:

 DBA
Password:

%ESQINT–I–CONNECT, Server ESQ1569 connected successfully

  SELECT * FROM information_schema.tables;
  SELECT * FROM information_schema.tables;

 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

 CANWEPARSELENGTH18              CHARACTER18TABLE18              BASE TABLE
 CANWEPARSELENGTH18              CHARACTERS18VIEW18              VIEW
---EOF---
 output (col 1–107 of 107):

  QUIT;
%ESQINT–I–TERMINATED, 02–SEP–1997 17:57:46

 ESGACS1001–* ACCESS node 1,569 is initialised
 ESGVTM0001–* Initialization for VTAM Application DAEFEQ01 starting
 ESGBPM0016–* BP VTAM Allocated successfully
 ESGBPM0006–* SP RPLS(1) Esize=112 Eno=39 Size=4,368 Loc=ANY
 ESGVTM0002–* Initialization for VTAM Application DAEFEQ01 complete
 SAGESQ–I–ESQINFO, ESQLOG is enabled for 'CMD'
 SAGESQ–I–ESQINFO, Software AG ADABAS SQL Server starting up on
                   Mainframe/VSE/DESA
 SAGESQ–I–ESQINFO, Version:         1.4.2.7
 SAGESQ–I–ESQINFO, Startup time:    02–SEP–1997 17:45:52
 SAGESQ–I–ESQINFO, Server name:     ESQ1569 on node MAINFRAME NETWORK
 SAGESQ–I–ESQINFO, Server type:     CSCI
 SAGESQ–I–ESQINFO, Catalog files:   DB 63, Files 21–23
 ESGSVR0018–* Server ESQ1569 Function: INIT SDE: 000487F8
 ESGSVR0019–*        Length: 029 DAT: 0000,C=5,ESQINI,
 ESGSVR0014–* Server ESQ1569 Register failed
 ESGSVR0018–* Server ESQ Function: INIT SDE: 00048820
```

```
ESGSVR0019-*        Length: 025 DAT: 0000,C=1,TSVRCSS
ESGSVR0014-* Server ESQ Register failed
ESGINI0005-* Nucleus size: 151K below, 270K above
ESGOPC0000-* ENTIRE/SRVMIS INITIALISED
SAGESQ-I-ESQINFO, ESQLOG is enabled for 'CSCI'
02-SEP 17:57:43.65 Th0: sending the reply
02-SEP 17:57:43.72 Th0: f=PQS c=3BC900,GEB     ,         l=338,0,0
02-SEP 17:57:43.73 Th0: waiting for a request ...
02-SEP 17:57:43.74 Th0: f=LQS c=3BC900,GEB     ,         l=432,342,432
02-SEP 17:57:44.19 Th0: sending the reply
02-SEP 17:57:44.29 Th0: f=PQS c=3BC900,GEB     ,         l=342,0,0
02-SEP 17:57:44.31 Th0: waiting for a request ...
02-SEP 17:57:44.31 Th0: f=LQS c=3BC900,GEB     ,         l=306,1532,306
02-SEP 17:57:44.32 Th0: sending the reply
02-SEP 17:57:44.40 Th0: f=PQS c=3BC900,GEB     ,         l=1532,0,0
02-SEP 17:57:44.41 Th0: waiting for a request ...
02-SEP 17:57:44.42 Th0: f=LQS c=3BC900,GEB     ,         l=306,1532,306
02-SEP 17:57:44.43 Th0: sending the reply
02-SEP 17:57:44.51 Th0: f=PQS c=3BC900,GEB     ,         l=1532,0,0
02-SEP 17:57:44.52 Th0: waiting for a request ...
02-SEP 17:57:44.53 Th0: f=LQS c=3BC900,GEB     ,         l=304,310,304
02-SEP 17:57:44.54 Th0: sending the reply
02-SEP 17:57:44.59 Th0: f=PQS c=3BC900,GEB     ,         l=310,0,0
02-SEP 17:57:44.60 Th0: waiting for a request ...
02-SEP 17:57:44.62 Th0: f=LQS c=3BC900,GEB     ,         l=264,310,264
02-SEP 17:57:44.63 Th0: sending the reply
02-SEP 17:57:44.68 Th0: f=PQS c=3BC900,GEB     ,         l=310,0,0
02-SEP 17:57:44.70 Th0: waiting for a request ...
02-SEP 17:57:44.71 Th0: f=LQS c=3BC900,GEB     ,         l=324,2118,324
02-SEP 17:57:44.86 Th0: sending the reply
02-SEP 17:57:44.94 Th0: f=PQS c=3BC900,GEB     ,         l=2118,0,0
02-SEP 17:57:44.94 Th0: waiting for a request ...
02-SEP 17:57:44.95 Th0: f=LQS c=3BC900,GEB     ,         l=324,2118,324
02-SEP 17:57:44.97 Th0: sending the reply
02-SEP 17:57:45.03 Th0: f=PQS c=3BC900,GEB     ,         l=2118,0,0

02-SEP 17:57:45.05 Th0: waiting for a request ...
02-SEP 17:57:46.79 Th0: accept CLOSE
ESGOPC0030-* ENTIRE/SRVMTERMINATION IN PROGRESS
ESGOPC0001-* VTAM STOP COMPLETED.
ESGVTM0001-* Termination for VTAM Application DAEFEQ01 starting
ESGVTM0002-* Termination for VTAM Application DAEFEQ01 complete
ESGBPM0017-* BP VTAM Deleted successfully
ESGOPC0001-* ACCESS STOP COMPLETED.
ESGACS4001-* This ACCESS node is no longer active
ESGOPC0053-* ENTIRE/SRVMTHREADS QUIESCED
```

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:

```
VOENV NAME=ESQLOG,VALUE='SES'
 SAGESQ-I-ESQINFO, ESQLOG is enabled for 'SES'

%ESQINT-I-STARTED, 03-SEP-1997 14:51:16, Version 1.4/2.7 (Mainframe/VSE/DESA)
%ESQINT-I-BATMODE, Running in batch mode

ESQ User:

 DBA
Password:


%ESQINT-I-CONNECT, Server ESQ1569 connected successfully

  SELECT * FROM information_schema.tables;
  SELECT * FROM information_schema.tables;

 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
```

```
CANWEPARSELENGTH18              CHARACTER18TABLE18              BASE TABLE
 CANWEPARSELENGTH18              CHARACTERS18VIEW18              VIEW
---EOF---
 output (col 1-107 of 107):

  QUIT;
%ESQINT-I-TERMINATED, 03-SEP-1997 14:51:22

 ESGOPC0000-* ENTIRE/SRVMIS INITIALISED
 SAGESQ-I-ESQINFO, ESQLOG is enabled for 'SES'
 03-SEP 14:51:22.37 Th0: CLOSE SESSION              csusr=:GEB
```

Schema Identifier Logging

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

Example:

```
%ESQINT–I–STARTED, 03–SEP–1997 16:15:30, Version 1.4/2.7 (Mainframe/VSE/DESA)
%ESQINT–I–BATMODE, Running in batch mode

ESQ User:

 DBA
Password:


%ESQINT–I–CONNECT, Server ESQ1569 connected successfully

  SELECT * FROM information_schema.tables;
  SELECT * FROM information_schema.tables;

 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


 CANWEPARSELENGTH18              CHARACTER18TABLE18              BASE TABLE
 CANWEPARSELENGTH18              CHARACTERS18VIEW18              VIEW
---EOF---
 output (col 1–107 of 107):

  QUIT;
%ESQINT–I–TERMINATED, 03–SEP–1997 16:15:34
```

```
ESGOPC0000–* ENTIRE/SRVMIS INITIALISED
SAGESQ–I–ESQINFO, ESQLOG is enabled for 'SCHEMAIDENT'
03–SEP 16:15:33.33 Th0: schema–ident srv=DBA clt=DBA
03–SEP 16:15:33.84 Th0: schema–ident srv=DBA clt=DBA
03–SEP 16:15:33.86 Th0: schema–ident srv=DBA clt=DBA
03–SEP 16:15:33.88 Th0: schema–ident srv=DBA clt=DBA
03–SEP 16:15:33.90 Th0: schema–ident srv=DBA clt=DBA
03–SEP 16:15:33.92 Th0: schema–ident srv=DBA clt=DBA
03–SEP 16:15:34.19 Th0: schema–ident srv=DBA clt=DBA
03–SEP 16:15:34.23 Th0: schema–ident srv=DBA clt=DBA
03–SEP 16:15:34.29 Th0: schema–ident srv=DBA clt=DBA
03–SEP 16:15:34.34 Th0: schema–ident srv=DBA clt=DBA
03–SEP 16:15:34.40 Th0: schema–ident srv=DBA clt=DBA
03–SEP 16:15:34.46 Th0: schema–ident srv=DBA clt=DBA
03–SEP 16:15:34.53 Th0: schema–ident srv=DBA clt=DBA
03–SEP 16:15:34.61 Th0: schema–ident srv=DBA clt=DBA
03–SEP 16:15:34.66 Th0: schema–ident srv=DBA clt=DBA
03–SEP 16:15:34.72 Th0: schema–ident srv=DBA clt=DBA
```

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:

```
VOENV NAME=ESQLOG,VALUE='USEREXIT'

 SAGESQ-I-ESQINFO, ESQLOG is enabled for 'USEREXIT'
 SAGESQ-I-ESQINFO, Clt: > uex1(cmd=3,usr=GEB)
 SAGESQ-I-ESQINFO, Clt: < uex1(sqlcode=0,ul=3,usr=GEB)
 SAGESQ-I-ESQINFO, Clt: > uex2(sqlcode=0)
 SAGESQ-I-ESQINFO, Clt: < uex2(sqlcode=0)
 SAGESQ-I-ESQINFO, Clt: > uex1(cmd=108)
 SAGESQ-I-ESQINFO, Clt: < uex1(sqlcode=0)
 SAGESQ-I-ESQINFO, Clt: > uex2(sqlcode=0)
 SAGESQ-I-ESQINFO, Clt: < uex2(sqlcode=0)
 SAGESQ-I-ESQINFO, Clt: > uex1(cmd=101)
 SAGESQ-I-ESQINFO, Clt: < uex1(sqlcode=0)
 SAGESQ-I-ESQINFO, Clt: > uex2(sqlcode=0)
 SAGESQ-I-ESQINFO, Clt: < uex2(sqlcode=0)
 SAGESQ-I-ESQINFO, Clt: > uex1(cmd=101)
 SAGESQ-I-ESQINFO, Clt: < uex1(sqlcode=0)
 SAGESQ-I-ESQINFO, Clt: > uex2(sqlcode=0)
 SAGESQ-I-ESQINFO, Clt: < uex2(sqlcode=0)
 SAGESQ-I-ESQINFO, Clt: > uex1(cmd=102)
 SAGESQ-I-ESQINFO, Clt: < uex1(sqlcode=0)

%ESQINT-I-STARTED, 05-SEP-1997 12:28:28, Version 1.4/2.7 (Mainframe/VSE/DESA)
%ESQINT-I-BATMODE, Running in batch mode

ESQ User:

 GEB
Password:


%ESQINT-I-CONNECT, Server ESQ1569 connected successfully
```

```
  select yacht_id from sagtours.yacht;
  select yacht_id from sagtours.yacht;

    YACHT_ID

         144
         145
         146
         147
         148
         149
         150
         151
         152
         153
         154
         155
         156
         157
        5001
        6200
        6228
        6230
---EOF---
 output (col 1-12 of 12):

  QUIT;
%ESQINT-I-TERMINATED, 05-SEP-1997 12:28:46 ESGOPC0000-* ENTIRE/SRVMIS
INITIALISED
 SAGESQ-I-ESQINFO, ESQLOG is enabled for 'USEREXIT'
 SAGESQ-I-ESQINFO, Th0: > uex5(sid:2, con:-1, cc:OP)
 SAGESQ-I-ESQINFO, Th0: < uex5(rc=0)
 SAGESQ-I-ESQINFO, Th0: > uex6(sid:2, con:-1, cc:OP, adabas_rc:0)
 SAGESQ-I-ESQINFO, Th0: < uex6(rc=0)
 SAGESQ-I-ESQINFO, Th0: > uex5(sid:2, con:-1, cc:LF)
 SAGESQ-I-ESQINFO, Th0: < uex5(rc=0)
 SAGESQ-I-ESQINFO, Th0: > uex6(sid:2, con:-1, cc:LF, adabas_rc:0)
 SAGESQ-I-ESQINFO, Th0: < uex6(rc=0)
 SAGESQ-I-ESQINFO, Th0: > uex5(sid:2, con:-1, cc:LF)
 SAGESQ-I-ESQINFO, Th0: < uex5(rc=0)
 SAGESQ-I-ESQINFO, Th0: > uex6(sid:2, con:-1, cc:LF, adabas_rc:0)
 SAGESQ-I-ESQINFO, Th0: < uex6(rc=0)
 SAGESQ-I-ESQINFO, Th0: > uex5(sid:2, con:-1, cc:L2)
 SAGESQ-I-ESQINFO, Th0: < uex5(rc=0)
 SAGESQ-I-ESQINFO, Th0: > uex6(sid:2, con:-1, cc:L2, adabas_rc:0)
 SAGESQ-I-ESQINFO, Th0: < uex6(rc=0)
```

```
 SAGESQ-I-ESQINFO, Th0: > uex5(sid:2, con:-1, cc:S1)
 SAGESQ-I-ESQINFO, Th0: < uex5(rc=0)
 SAGESQ-I-ESQINFO, Th0: > uex6(sid:2, con:-1, cc:S1, adabas_rc:0)
 SAGESQ-I-ESQINFO, Th0: < uex6(rc=0)

 SAGESQ-I-ESQINFO, ESQLOG is enabled for 'TIME'
 SAGESQ-I-ESQINFO, Clt: ti-quot     tot     clt     net     srv  vo/net     ada
 SAGESQ-I-ESQINFO, Clt:      ms     350    -785     960      68     107       0
 SAGESQ-I-ESQINFO, Clt:      %      100     -223     274      19      31
0
 SAGESQ-I-ESQINFO, Clt: ti-quot     tot     clt     net     srv  vo/net     ada
 SAGESQ-I-ESQINFO, Clt:      ms     322       0       8    -656     617     353
 SAGESQ-I-ESQINFO, Clt:      %      100       0       2     -203     192
110
 SAGESQ-I-ESQINFO, Clt: ti-quot     tot     clt     net     srv  vo/net     ada
 SAGESQ-I-ESQINFO, Clt:      ms      14       0       7       5       2       0
 SAGESQ-I-ESQINFO, Clt:      %      100       0      50      36      14       0
 SAGESQ-I-ESQINFO, Clt: ti-quot     tot     clt     net     srv  vo/net     ada
 SAGESQ-I-ESQINFO, Clt:      ms      33       0      30       1       2       0
 SAGESQ-I-ESQINFO, Clt:      %      100       0      91       3       6       0
 SAGESQ-I-ESQINFO, Clt: ti-quot     tot     clt     net     srv  vo/net     ada
 SAGESQ-I-ESQINFO, Clt:      ms      23       1      16       2       4       0
 SAGESQ-I-ESQINFO, Clt:      %      100       4      70       9      17       0


 SAGESQ-I-ESQINFO, Clt: ti-quot     tot     clt     net     srv  vo/net     ada
 SAGESQ-I-ESQINFO, Clt:      ms     251       2     162      87       0       0
 SAGESQ-I-ESQINFO, Clt:      %      100       1      65      35       0       0
 SAGESQ-I-ESQINFO, Clt: elapsed time for ESQ session    3184 ms

%ESQINT-I-STARTED, 05-SEP-1997 13:49:14, Version 1.4/2.7 (Mainframe/VSE/DESA)
%ESQINT-I-BATMODE, Running in batch mode

ESQ User:

 GEB
Password:


%ESQINT-I-CONNECT, Server ESQ1569 connected successfully

  select * from information_schema.tables;
  select * from information_schema.tables;

 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


---EOF---
 output (col 1-107 of 107):

  QUIT;
%ESQINT-I-TERMINATED, 05-SEP-1997 13:49:55
```

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

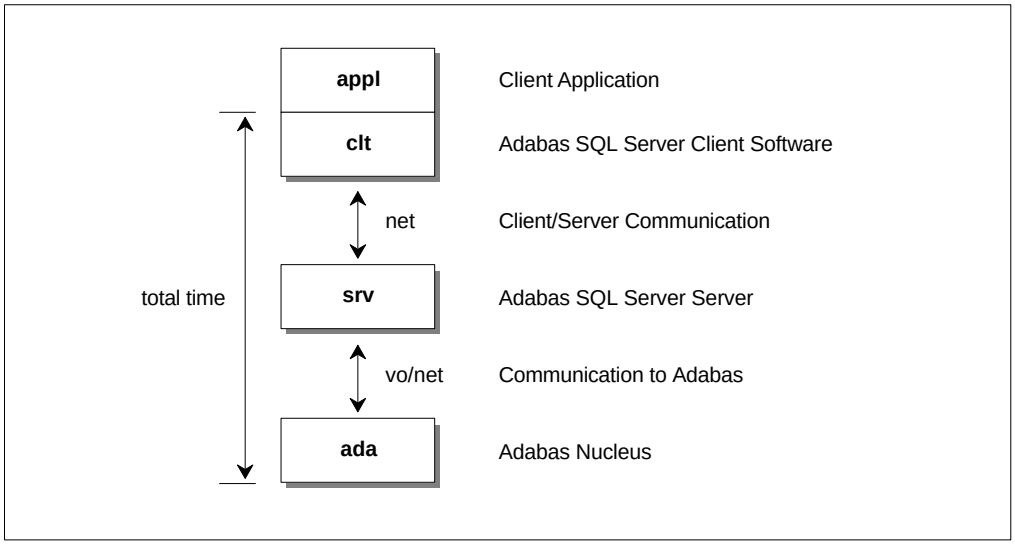


Figure 6-1: Data Flow and Time Portions

Example:

```
SAGESQ–I–ESQINFO, ESQLOG is enabled for ’TIME’
SAGESQ–I–ESQINFO, Clt: ti–quot     tot     clt     net     srv  vo/net     ada
SAGESQ–I–ESQINFO, Clt:       ms     275    –856     911      94     115      11
SAGESQ–I–ESQINFO, Clt:      %      100    –310     331      34      42       4
SAGESQ–I–ESQINFO, Clt: ti–quot     tot     clt     net     srv  vo/net     ada
SAGESQ–I–ESQINFO, Clt:       ms     653       1       8      92     494      58
SAGESQ–I–ESQINFO, Clt:      %      100       0       1      14      76       9
SAGESQ–I–ESQINFO, Clt: ti–quot     tot     clt     net     srv  vo/net     ada
SAGESQ–I–ESQINFO, Clt:       ms      24       1      19       2       2       0
SAGESQ–I–ESQINFO, Clt:      %      100       4      79       8       8       0
SAGESQ–I–ESQINFO, Clt: ti–quot     tot     clt     net     srv  vo/net     ada
SAGESQ–I–ESQINFO, Clt:       ms      22       1      15       2       4       0
SAGESQ–I–ESQINFO, Clt:      %      100       5      68       9      18       0
SAGESQ–I–ESQINFO, Clt: ti–quot     tot     clt     net     srv  vo/net     ada
SAGESQ–I–ESQINFO, Clt:       ms      31       1      25       1       4       0
SAGESQ–I–ESQINFO, Clt:      %      100       3      81       3      13       0
SAGESQ–I–ESQINFO, Clt: ti–quot     tot     clt     net     srv  vo/net     ada
SAGESQ–I–ESQINFO, Clt:       ms      65       1      16       3      43       2
SAGESQ–I–ESQINFO, Clt:      %      100       2      25       5      66       3
SAGESQ–I–ESQINFO, Clt: ti–quot     tot     clt     net     srv  vo/net     ada
SAGESQ–I–ESQINFO, Clt:       ms      41       1      12       3      23       2
SAGESQ–I–ESQINFO, Clt:      %      100       2      29       7      56       5
SAGESQ–I–ESQINFO, Clt: ti–quot     tot     clt     net     srv  vo/net     ada
SAGESQ–I–ESQINFO, Clt:       ms       0       0       0       0       0       0
SAGESQ–I–ESQINFO, Clt:      %      100       0       0       0       0       0
SAGESQ–I–ESQINFO, Clt: ti–quot     tot     clt     net     srv  vo/net     ada
SAGESQ–I–ESQINFO, Clt:       ms       0       0       0       0       0       0
SAGESQ–I–ESQINFO, Clt:      %      100       0       0       0       0       0
SAGESQ–I–ESQINFO, Clt: ti–quot     tot     clt     net     srv  vo/net     ada
SAGESQ–I–ESQINFO, Clt:       ms       0       0       0       0       0       0
SAGESQ–I–ESQINFO, Clt:      %      100       0       0       0       0       0
SAGESQ–I–ESQINFO, Clt: ti–quot     tot     clt     net     srv  vo/net     ada
SAGESQ–I–ESQINFO, Clt:       ms      20       1      18       1       0       0
SAGESQ–I–ESQINFO, Clt:      %      100       5      90       5       0       0
SAGESQ–I–ESQINFO, Clt: ti–quot     tot     clt     net     srv  vo/net     ada
SAGESQ–I–ESQINFO, Clt:       ms      39       2      25      12       0       0
SAGESQ–I–ESQINFO, Clt:      %      100       5      64      31       0       0
SAGESQ–I–ESQINFO, Clt: elapsed time for ESQ session    1170 ms
```

Security Logging

The logging mechanism of Adabas SQL Server also enables logging of GRANT and REVOKE statements and user system access using CONNECT statements. Logging for this can be enabled using the environment variable ESQLOG.

Example 1:

```
VOENV NAME=ESQLOG,VALUE='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:

```
VOENV NAME=ESQLOG,VALUE='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:

```
VOENV NAME=ESQLOG,VALUE='–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 ESQBIF and
- the point in time when control is returned to the application

Example:

```
SAGESQ-I-ESQINFO, ESQLOG is enabled for 'ENTRY'
SAGESQ-I-ESQINFO, Clt: > ESQLNK()        cnt=   1
SAGESQ-I-ESQINFO, Clt: < ESQLNK()        cnt=   1      rsp=0
SAGESQ-I-ESQINFO, Clt: > esqinf()        cnt=   2
SAGESQ-I-ESQINFO, Clt: < esqinf()        cnt=   2      rsp=0
SAGESQ-I-ESQINFO, Clt: > esqinf()        cnt=   3
SAGESQ-I-ESQINFO, Clt: < esqinf()        cnt=   3      rsp=0
SAGESQ-I-ESQINFO, Clt: > esqinf()        cnt=   4
SAGESQ-I-ESQINFO, Clt: < esqinf()        cnt=   4      rsp=0
SAGESQ-I-ESQINFO, Clt: > esqinf()        cnt=   5
SAGESQ-I-ESQINFO, Clt: < esqinf()        cnt=   5      rsp=0
SAGESQ-I-ESQINFO, Clt: > esqinf()        cnt=   6
SAGESQ-I-ESQINFO, Clt: < esqinf()        cnt=   6      rsp=0
SAGESQ-I-ESQINFO, Clt: > ESQLNK()        cnt=   7
SAGESQ-I-ESQINFO, Clt: < ESQLNK()        cnt=   7      rsp=0
SAGESQ-I-ESQINFO, Clt: > ESQLNK()        cnt=   8
SAGESQ-I-ESQINFO, Clt: < ESQLNK()        cnt=   8      rsp=0
SAGESQ-I-ESQINFO, Clt: > ESQLNK()        cnt=   9
SAGESQ-I-ESQINFO, Clt: < ESQLNK()        cnt=   9      rsp=0
SAGESQ-I-ESQINFO, Clt: > ESQLNK()        cnt=  10
SAGESQ-I-ESQINFO, Clt: < ESQLNK()        cnt=  10      rsp=0
SAGESQ-I-ESQINFO, Clt: > ESQLNK()        cnt=  11
SAGESQ-I-ESQINFO, Clt: < ESQLNK()        cnt=  11      rsp=0
SAGESQ-I-ESQINFO, Clt: > ESQLNK()        cnt=  12
SAGESQ-I-ESQINFO, Clt: < ESQLNK()        cnt=  12      rsp=0
SAGESQ-I-ESQINFO, Clt: > ESQLNK()        cnt=  13
SAGESQ-I-ESQINFO, Clt: < ESQLNK()        cnt=  13      rsp=0
SAGESQ-I-ESQINFO, Clt: > ESQLNK()        cnt=  14
SAGESQ-I-ESQINFO, Clt: < ESQLNK()        cnt=  14      rsp=0
SAGESQ-I-ESQINFO, Clt: > ESQLNK()        cnt=  15
SAGESQ-I-ESQINFO, Clt: < ESQLNK()        cnt=  15      rsp=0
SAGESQ-I-ESQINFO, Clt: > ESQLNK()        cnt=  16
SAGESQ-I-ESQINFO, Clt: < ESQLNK()        cnt=  16      rsp=0
SAGESQ-I-ESQINFO, Clt: > ESQLNK()        cnt=  17
SAGESQ-I-ESQINFO, Clt: < ESQLNK()        cnt=  17      rsp=0
SAGESQ-I-ESQINFO, Clt: > ESQLNK()        cnt=  18
```

```
 SAGESQ-I-ESQINFO, Clt: < ESQLNK()        cnt=  18      rsp=0
 SAGESQ-I-ESQINFO, Clt: > ESQLNK()        cnt=  19
 SAGESQ-I-ESQINFO, Clt: < ESQLNK()        cnt=  19      rsp=0
 SAGESQ-I-ESQINFO, Clt: > ESQLNK()        cnt=  20
 SAGESQ-I-ESQINFO, Clt: < ESQLNK()        cnt=  20      rsp=0
 SAGESQ-I-ESQINFO, Clt: > ESQLNK()        cnt=  21
 SAGESQ-I-ESQINFO, Clt: < ESQLNK()        cnt=  21      rsp=0
 SAGESQ-I-ESQINFO, Clt: > ESQLNK()        cnt=  22
 SAGESQ-I-ESQINFO, Clt: < ESQLNK()        cnt=  22      rsp=100
 SAGESQ-I-ESQINFO, Clt: > ESQLNK()        cnt=  23
 SAGESQ-I-ESQINFO, Clt: < ESQLNK()        cnt=  23      rsp=0
 SAGESQ-I-ESQINFO, Clt: > ESQLNK()        cnt=  24
 SAGESQ-I-ESQINFO, Clt: < ESQLNK()        cnt=  24      rsp=0

%ESQINT-I-STARTED, 08-SEP-1997 16:09:25, Version 1.4/2.7 (Mainframe/VSE/DESA)
%ESQINT-I-BATMODE, Running in batch mode

ESQ User:

 DBA
Password:

 %ESQINT-I-CONNECT, Server ESQ1569 connected successfully

  select * from information_schema.users;
  select * from information_schema.users;

 USER_ID

 DBA
 HU
 CUGINI
 MCGINN
 SULLIVAN1
 SULLIVAN
 SCHANZLE
 CANWEPARSELENGTH18
 GEB
 ESQ
---EOF---
 output (col 1-33 of 33):

  QUIT;
%ESQINT-I-TERMINATED, 08-SEP-1997 16:09:28
```

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.

```
VOENV NAME=ESQLOG,VALUE='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 full statement execution
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.

For example:

- The created view:

```
create view sagtours.view_1 as
select start_harbor, destination_harbor
from sagtours.cruise
where cruise_price < 2000;
```

- The executed statement:

```
select destination_harbor
from sagtours.view1;
```

Internally, the statement will go through the view merging process, yielding a new (invisible) statement:

```
select destination_harbor
from sagtours.cruise
where cruise_price < 2000
and start_harbor = 'MIAMI';
```

whose compilation is thereafter subject to explain logging

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

for (L2 ;SAGTOURS.CRUISE-0; L2 )  /* GET NEXT */
{
  (((SAGTOURS.CRUISE.CRUISE_PRICE < 2000 )
    (SAGTOURS.CRUISE.START_HARBOR = 'MIAMI' )))
}

      where start_harbor = 'MIAMI';

********    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 using 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 short name 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 know 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 sagtours.cruise where exists

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

if (first execution for SAGTOURS.YACHT–0; )
{
  S1 (AA >= 00A0 EOF )
}
  for (R1 ;SAGTOURS.YACHT–0; L1N)  /* GET NEXT */
  {
    (((SAGTOURS.YACHT.YACHT_ID > 0 )))
  }
  for (L2 ;SAGTOURS.CRUISE–0; L2 )  /* GET NEXT */
  {
  }

  (select *from sagtours.yacht where yacht_id > 0);

********    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 sagtours.yacht

  group by yacht_type

********   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’ )))
}

  having min(yacht_name) = ’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.

For 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

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.

Adabas Command Logging

In terms of Adabas, Adabas SQL Server is an application program which processes SQL requests. To process SQL requests, Adabas SQL Server uses 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 enable the monitoring and analysis of an application, Adabas SQL Server offers the possibility to log the usage of Adabas. In addition to this, Adabas command logging may be used for trouble shooting.

Due to its internal structure, Adabas SQL Server has two software layers which logically interface to Adabas:

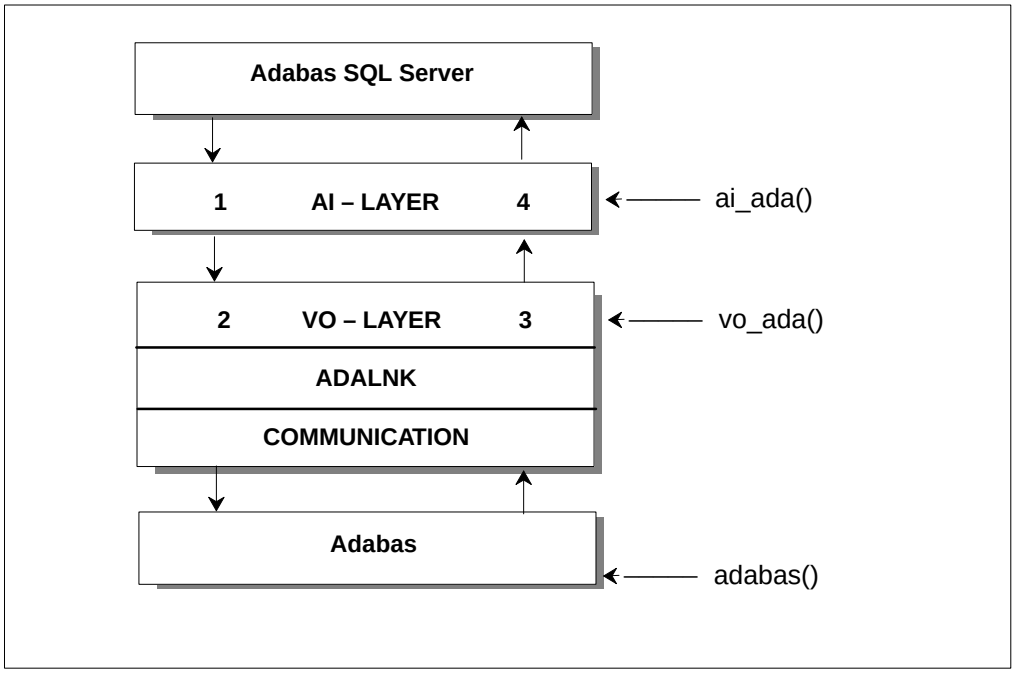
- The AI (Adabas Interface) layer

  This layer maps CLIENT contexts to the matching Adabas session contexts (User queue elements) and performs various optimizations (Multifetch, RC optimization, etc..)

- The VO (Virtual Operating-system) layer

  This layer performs the real “call Adabas()” for different contexts.

Based on this, Adabas SQL Server offers four logging points:



Lo.Dg 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

Adabas SQL Server Adabas command logging is enabled and controlled using 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:   ESQRUN
*
* DESCRIPTION: ESQ RUNTIME PARAMETER FILE
*
* VERSION:     1.4 94/07/27 12:34:26 (CO) SOFTWARE AG
*
####################################################################*/

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

- Non-Mainframe Platforms.

  On non-mainframe platforms, the logging information is written to the logical channel “operator console”, which is mapped to STDOUT. The logging may be redirected either using 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.

- Mainframe Platforms

  On mainframe platforms, the logging information is written to the file defined for ESQCLOG (DLBL), with LRECL=80.

Examples of an Adabas Command Log

1. ADABAS LOG (CLOG)

```
*
09-SEP 11:15:59.35 CLOG 2 *ESQRTS  1 OP..   0/ 63 ....   0 RB=”.”
09-SEP 11:15:59.68 CLOG 2 *ESQRTS  2 LF S  21/ 63        0 ISN:0
09-SEP 11:15:59.79 CLOG 2 *ESQRTS  3 LF S  22/ 63        0 ISN:0
09-SEP 11:16:00.98 CLOG 2 *ESQRTS  4 L2M   21/ 63 IP01+  0 ISN:1
09-SEP 11:16:00.03 CLOG 2 *ESQRTS  1 S1 I  21/ 63 IM00+  0 ISQ:1
09-SEP 11:16:00.04 CLOG 2 *ESQ     2 OP..   0/ 63 ....   0 RB=”.”
09-SEP 11:16:00.10 CLOG 2 *ESQRTS  1 L3 V  23/ 63 IRMP+  0 ISN:1
09-SEP 11:16:00.18 CLOG 2 *ESQRTS  2 S1 I  21/ 63 IA00+  0 ISQ:1
09-SEP 11:16:00.18 CLOG 2 *ESQRTS  3 S1 I  21/ 63 IB00   0 ISQ:11
09-SEP 11:16:00.19 CLOG 2 *ESQRTS  4 S2 I  21/ 63 IB00+  0 ISQ:11
09-SEP 11:16:00.20 CLOG 2 *ESQRTS  5 L1MN  21/ 63 IB00+  0 ISN:4964
09-SEP 11:16:00.21 CLOG 2 *ESQRTS  6 RCS   21/ 63 IB00+  0 ISN:0
09-SEP 11:16:00.22 CLOG 2 *ESQ     7 LF.S  45/ 63 ....   0 ISN:0
09-SEP 11:16:00.23 CLOG 2 *ESQRTS  1 L3 V  23/ 63 IRMP+  0 ISN:2
09-SEP 11:16:00.28 CLOG 2 *ESQRTS  1 L3 V  23/ 63 IRMP+  0 ISN:3
09-SEP 11:16:00.29 CLOG 2 *ESQRTS  1 L3 V  23/ 63 IRMP+  0 ISN:2036
09-SEP 11:16:00.32 CLOG 2 *ESQRTS  1 L3 V  23/ 63 IRMP+  0 ISN:2037
09-SEP 11:16:00.33 CLOG 2 *ESQ     1 L2M   45/ 63 S001   0 ISN:19
09-SEP 11:16:00.36 CLOG 2 *ESQ     1 L2M   45/ 63 S001   0 ISN:35
09-SEP 11:16:00.40 CLOG 2 *ESQ     1 BT     0/ 63 ....   0 ISN:0
09-SEP 11:16:00.40 CLOG 2 *ESQ     2 CL     0/ 63 ....   0 ISN:0
```

Discussion of CLOG-style Logging

Each executed Adabas call is logged with one clog line. The clog line can be read as follows:

```
 9–SEP 17:47:07.81    CLOG     2  ESQRTS      4 L1MN 215/230 IB00+  0 ISN:3943


                      ^      ^     ^        ^ ^ ^^ ^   ^   ^      ^ ^
                      |      |     |        | | || |   |   |      | |
                      |      |     |        | | || |   |   |      | |
       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  ---------+
```

```
/*######################################################################
*
* FILE NAME:   ESQRUN
*
* DESCRIPTION: ESQ RUNTIME PARAMETER FILE
*
* VERSION:     1.4 94/07/27 12:34:26 (CO) SOFTWARE AG
*
######################################################################*/

GLOBAL
BEGIN
.
.
.
   ADABAS    LOG   (CLOG,VO IN(FB(FULL)))
END
```

```
*****************************************************************************
*
*                                 ESQCLOG
*
*
* NODE: MAINFRAME NETWORK  TARGET: Mainframe VSE/DESA   DATE: 09-SEP-1997 11:15
*
*
*****************************************************************************
*
09-SEP 11:15:59.35 CLOG 2 *ESQRTS  1 OP..   0/ 63 ....   0 RB=”.”
09-SEP 11:15:59.68 CLOG 2 *ESQRTS  2 LF S  21/ 63        0 ISN:0
09-SEP 11:15:59.79 CLOG 2 *ESQRTS  3 LF S  22/ 63        0 ISN:0
ADALOG 2 *ESQRTS > VO_ADA FB FULL (LENGTH: 300)
0017E4B8 00000000:   C6C16BF3 F26BC16B C6C26BF1 6BC16BC6    FA,32,A,FB,1,A,F
0017E4C8 00000010:   C36BF86B C26BC6C4 6BF3F26B C16BC6C5    C,8,B,FD,32,A,FE
0017E4D8 00000020:   6BF3F26B C16BC6C6 6BF16BC1 4B000000    ,32,A,FF,1,A....
0017E4E8 00000030:   00000000 00000000 00000000 00000000    ................
                14 lines identical to line above
0017E5D8 00000120:   00000000 00000000 00000000             ............
09-SEP 11:16:00.98 CLOG 2 *ESQRTS  4 L2M   21/ 63 IP01+  0 ISN:1
ADALOG 2 *ESQRTS > VO_ADA FB FULL (LENGTH: 300)
0017F428 00000000:   E2C16BF3 F26BC16B E2C26BF3 F26BC24B    SA,32,A,SB,32,B.
0017F438 00000010:   00000000 00000000 00000000 00000000    ................
                16 lines identical to line above
0017F548 00000120:   00000000 00000000 00000000             ............
09-SEP 11:16:00.03 CLOG 2 *ESQRTS  1 S1 I  21/ 63 IM00+  0 ISQ:1
09-SEP 11:16:00.04 CLOG 2 *ESQ     2 OP..   0/ 63 ....   0 RB=”.”
ADALOG 2 *ESQRTS > VO_ADA FB FULL (LENGTH: 57)
09E79148 00000000:   C1C16BF3 F26BC16B C1C26BF3 F26BC16B    AA,32,A,AB,32,A,
09E79158 00000010:   C1C36BF3 6BC16BC1 C46BF86B C26BC1C5    AC,3,A,AD,8,B,AE
09E79168 00000020:   6BF86BC2 6BC1C66B F46BC26B C1C7F160    ,8,B,AF,4,B,AG1-
09E79178 00000030:   F1F06BF1 F0F06BC2 4B                   10,100,B.
09-SEP 11:16:00.10 CLOG 2 *ESQRTS  1 L3 V  23/ 63 IRMP+  0 ISN:1
ADALOG 2 *ESQRTS > VO_ADA FB FULL (LENGTH: 300)
0017E718 00000000:   C8C16BF3 F26BC16B C8C26BF3 F26BC16B    HA,32,A,HB,32,A,
0017E728 00000010:   C8C36BF4 F06BC16B C8C46BF8 6BC26BC8    HC,40,A,HD,8,B,H
0017E738 00000020:   C56BF46B C66BC8C6 6BF46BC6 6BC8C76B    E,4,F,HF,4,F,HG,
0017E748 00000030:   F3F26BC1 6BC8C86B F26BC66B F2E76BC8    32,A,HH,2,F,2X,H
0017E758 00000040:   C96BF46B C66BC8D1 C36BF26B C66BC8D1    I,4,F,HJC,2,F,HJ
0017E768 00000050:   F160F26B F2F5F36B C16BC8D2 C36BF26B    1-2,253,A,HKC,2,
0017E778 00000060:   C66BC8D2 F160F26B F2F5F36B C16BC8D3    F,HK1-2,253,A,HL
0017E788 00000070:   6BF4F06B C16BC8D4 6BF16BC1 6BC8D56B    ,40,A,HM,1,A,HN,
0017E798 00000080:   F16BC16B C8D66BF1 6BC16BC8 D76BF16B    1,A,HO,1,A,HP,1,
0017E7A8 00000090:   C16BC8D8 6BF16BC1 6BF3E76B C8D96BF4    A,HQ,1,A,3X,HR,4
0017E7B8 000000A0:   6BC66BC8 E26BF46B C66BC8E3 6BF46BC6    ,F,HS,4,F,HT,4,F
```

```
0017E7C8 000000B0:   6BC8E46B F46BC66B C8E56BF4 6BC66BC8    ,HU,4,F,HV,4,F,H
0017E7D8 000000C0:   E66BF46B C64B0000 00000000 00000000    W,4,F...........
0017E7E8 000000D0:   00000000 00000000 00000000 00000000    ................
                4 lines identical to line above
0017E838 00000120:   00000000 00000000 00000000             ............
09-SEP 11:16:00.18 CLOG 2 *ESQRTS  2 S1 I  21/ 63 IA00+  0 ISQ:1
ADALOG 2 *ESQRTS > VO_ADA FB FULL (LENGTH: 1)
09E6BD65 00000000:   4B                                      .
09-SEP 11:16:00.18 CLOG 2 *ESQRTS  3 S1 I  21/ 63 IB00   0 ISQ:11
ADALOG 2 *ESQRTS > VO_ADA FB FULL (LENGTH: 1)
09E6BD65 00000000:   4B                                      .
09-SEP 11:16:00.19 CLOG 2 *ESQRTS  4 S2 I  21/ 63 IB00+  0 ISQ:11
ADALOG 2 *ESQRTS > VO_ADA FB FULL (LENGTH: 300)
0017E848 00000000:   C9C16BF3 F26BC16B C9C26BF3 F26BC16B    IA,32,A,IB,32,A,
0017E858 00000010:   C9C36BF3 F26BC16B C9C46BF4 6BC66BC9    IC,32,A,ID,4,F,I
0017E868 00000020:   C56BF26B C66BC9C6 6BF26BC1 6BC9C76B    E,2,F,IF,2,A,IG,
0017E878 00000030:   F4F06BC1 6BC9C86B F26BC16B C9C96BF2    40,A,IH,2,A,II,2
0017E888 00000040:   6BC66BC9 D16BF4F0 6BC16BC9 D26BF46B    ,F,IJ,40,A,IK,4,
0017E898 00000050:   C66BC9D3 6BF46BC6 6BC9D4C3 6BF26BC6    F,IL,4,F,IMC,2,F
0017E8A8 00000060:   6BC9D4F1 60F26BF2 F5F36BC1 6BC9D56B    ,IM1-2,253,A,IN,
0017E8B8 00000070:   F16BC16B C9D66BF1 6BC16BC9 D76BF16B    1,A,IO,1,A,IP,1,
0017E8C8 00000080:   C16BC9D8 6BF16BC1 6BC9D96B F16BC16B    A,IQ,1,A,IR,1,A,
0017E8D8 00000090:   F3E76BC9 E26BF46B C64B0000 00000000    3X,IS,4,F.......
0017E8E8 000000A0:   00000000 00000000 00000000 00000000    ................
                7 lines identical to line above
0017E968 00000120:   00000000 00000000 00000000             ............
09-SEP 11:16:00.20 CLOG 2 *ESQRTS  5 L1MN  21/ 63 IB00+  0 ISN:4964
09-SEP 11:16:00.21 CLOG 2 *ESQRTS  6 RCS   21/ 63 IB00+  0 ISN:0
09-SEP 11:16:00.22 CLOG 2 *ESQ     7 LF.S  45/ 63 ....   0 ISN:0
ADALOG 2 *ESQRTS > VO_ADA FB FULL (LENGTH: 57)
09E79148 00000000:   C1C16BF3 F26BC16B C1C26BF3 F26BC16B    AA,32,A,AB,32,A,
09E79158 00000010:   C1C36BF3 6BC16BC1 C46BF86B C26BC1C5    AC,3,A,AD,8,B,AE
09E79168 00000020:   6BF86BC2 6BC1C66B F46BC26B C1C7F160    ,8,B,AF,4,B,AG1-
09E79178 00000030:   F1F06BF1 F0F06BC2 4B                   10,100,B.
09-SEP 11:16:00.23 CLOG 2 *ESQRTS  1 L3 V  23/ 63 IRMP+  0 ISN:2
ADALOG 2 *ESQRTS > VO_ADA FB FULL (LENGTH: 57)
09E79148 00000000:   C1C16BF3 F26BC16B C1C26BF3 F26BC16B    AA,32,A,AB,32,A,
09E79158 00000010:   C1C36BF3 6BC16BC1 C46BF86B C26BC1C5    AC,3,A,AD,8,B,AE
09E79168 00000020:   6BF86BC2 6BC1C66B F46BC26B C1C7F160    ,8,B,AF,4,B,AG1-
09E79178 00000030:   F1F06BF1 F0F06BC2 4B                   10,100,B.
09-SEP 11:16:00.28 CLOG 2 *ESQRTS  1 L3 V  23/ 63 IRMP+  0 ISN:3
ADALOG 2 *ESQRTS > VO_ADA FB FULL (LENGTH: 57)
09E79148 00000000:   C1C16BF3 F26BC16B C1C26BF3 F26BC16B    AA,32,A,AB,32,A,
09E79158 00000010:   C1C36BF3 6BC16BC1 C46BF86B C26BC1C5    AC,3,A,AD,8,B,AE
09E79168 00000020:   6BF86BC2 6BC1C66B F46BC26B C1C7F160    ,8,B,AF,4,B,AG1-
09E79178 00000030:   F1F06BF1 F0F06BC2 4B                   10,100,B.
```

```
09–SEP 11:16:00.29 CLOG 2 *ESQRTS  1 L3 V  23/ 63 IRMP+  0 ISN:2036
ADALOG 2 *ESQRTS > VO_ADA FB FULL (LENGTH: 57)
09E79148 00000000:   C1C16BF3 F26BC16B C1C26BF3 F26BC16B    AA,32,A,AB,32,A,
09E79158 00000010:   C1C36BF3 6BC16BC1 C46BF86B C26BC1C5    AC,3,A,AD,8,B,AE
09E79168 00000020:   6BF86BC2 6BC1C66B F46BC26B C1C7F160    ,8,B,AF,4,B,AG1–
09E79178 00000030:   F1F06BF1 F0F06BC2 4B                   10,100,B.
09–SEP 11:16:00.32 CLOG 2 *ESQRTS  1 L3 V  23/ 63 IRMP+  0 ISN:2037
ADALOG 2 *ESQ    > VO_ADA FB FULL (LENGTH: 7)
000BA378 00000000:   C1C16BF4 6BC64B                       AA,4,F.
09–SEP 11:16:00.33 CLOG 2 *ESQ     1 L2M   45/ 63 S001   0 ISN:19
ADALOG 2 *ESQ    > VO_ADA FB FULL (LENGTH: 7)
000BA378 00000000:   C1C16BF4 6BC64B                       AA,4,F.
09–SEP 11:16:00.36 CLOG 2 *ESQ     1 L2M   45/ 63 S001   0 ISN:35
09–SEP 11:16:00.40 CLOG 2 *ESQ     1 BT     0/ 63 ....   0 ISN:0
09–SEP 11:16:00.40 CLOG 2 *ESQ     2 CL     0/ 63 ....   0 ISN:0

%ESQINT–I–STARTED, 10–SEP–1997 11:15:57, Version 1.4/2.7 (Mainframe/VSE/DESA)
%ESQINT–I–BATMODE, Running in batch mode

ESQ User:

 ESQ
Password:


%ESQINT–I–CONNECT, Server ESQ1569 connected successfully

  select yacht_id from sagtours.yacht;
  select yacht_id from sagtours.yacht;

    YACHT_ID

         144

        6230
---EOF---
 output (col 1–12 of 12):

  quit;
%ESQINT–I–TERMINATED, 10–SEP–1997 11:16:00
```

Adabas SQL Trace Facilities

Tracing facilities have been included in some of the components of Adabas SQL Server. The “TRACE” version of a component has been compiled in a special way so that a trace file may be generated at runtime, which may be needed to diagnose software problems.

Trace output is written to the file assigned the DLBL ESQTRAC.

The following types of trace are available :

- Standard Trace (TR)
- Mini Trace (MT)
- System Calls Trace (VO-Trace)

Note:
Before using the Adabas SQL Client/Server trace facilities, please consult the section ***Activating the Trace Facilities*** *for a description of the required installation modifications.*

Warning:
Trace output is very large; tracing should not be used without Software AG support.

Standard Trace (TR)

Static Trace Control

Static Trace Control means that the trace objects are statically specified before program start and are valid until program end. Standard Trace is controlled using statements in the corresponding Adabas SQL parameter file (ESQPARM):

- one or more software components — TRACE (INCLUDE COMPONENTS (’c1’,...,’cn’) ;)
- all software components — TRACE (INCLUDE COMPONENTS (’*’) ;)
- one or more functions — TRACE (INCLUDE FUNCTIONS (’f1’,...,’fn’) ;)
- all software functions — TRACE (INCLUDE FUNCTIONS (’*’) ;)

Note:
The semi-colon “;” is a part of the syntax and is required.

Dynamic Trace Control (Precompiler Only)

The Adabas SQL precompiler also offers a dynamic way of controlling standard trace in order to focus trace on selected portions of the user’s program code as well as keeping precompile time short and trace output file small.

Dynamic trace control must be enabled in the precompiler parameter file:

```
TRACE ( DYNAMIC TRACE = ON; )
```

Trace control statements may then be added to the source code, which is to be precompiled, for example:

```
EXEC SQL TRACE “INCLUDE COMPONENTS (’c1’,...,’cn’) ;”;
...
EXEC SQL TRACE ON;
...
EXEC SQL TRACE OFF;
```

Note:
The semi–colon “;” is a part of the syntax and is required.

Mini Trace (MT)

Mini Trace (MT) is a static trace feature , which is available at a very early stage of an Adabas SQL program execution. It is especially useful when tracing is necessary for Adabas SQL system initialization, termination and client/server communication coding. In other words: Mini Trace is needed when the standard Adabas SQL trace is not yet ready to run or has already terminated.

Mini Trace is controlled by the environment variable ESQMITRC and can be assigned using the VOENV–Macro in the module VOPRM.

```
VOENV        NAME=ESQMITRC,VALUE='<value>'

where – <value>

"xx"     Trace one or more component(s), where "xx" is one or more component
         short names separated by commas, for example – "SL" (SQL Link).
"*"      Trace all components (while standard trace is not up)
"*,-BM"  Trace all components, without BM (Buffer Manager)
"X<..>"  Do not switch to standard trace (TR)
"SL=LOG" Trace logging of SL (client)
"RD=LOG" Trace logging of RD (server)
"VO=ERR" Trace VO calls only in case of VO errors
"VO=MCH" Each trace call (MT/TR) checks memory using VO (def: no check)
```

Unless redirected, all Adabas SQL trace is written to the dataset assigned the file link name ESQTRAC.

System Calls Trace (VO–Trace)

The VO Trace is currently not available on the mainframe platforms.

A

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 using 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	D	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	–	–	–	–

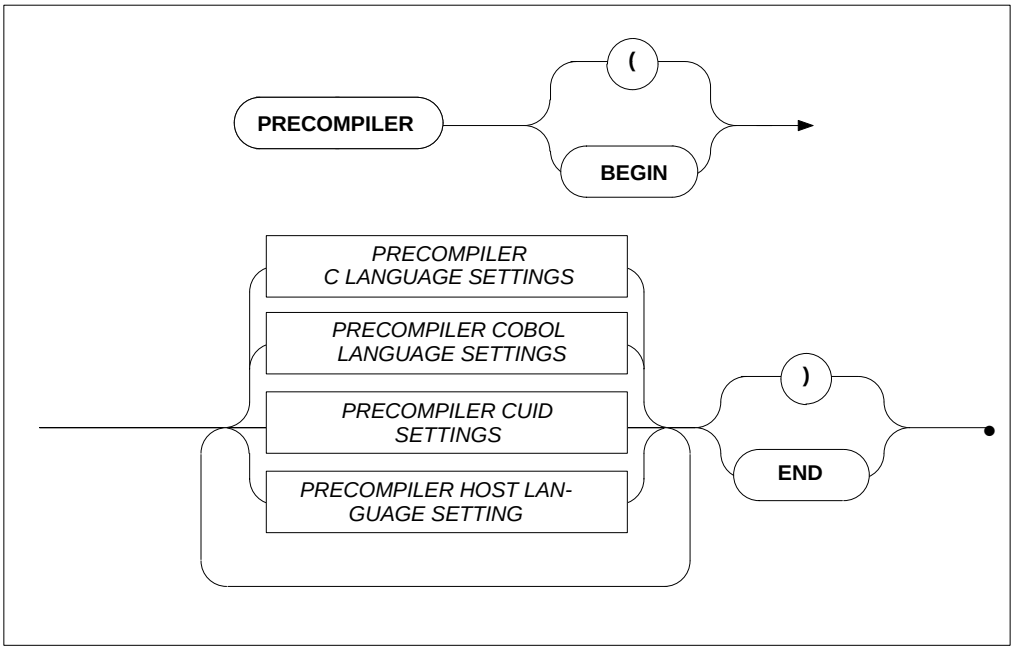
PRECOMPILER Settings

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

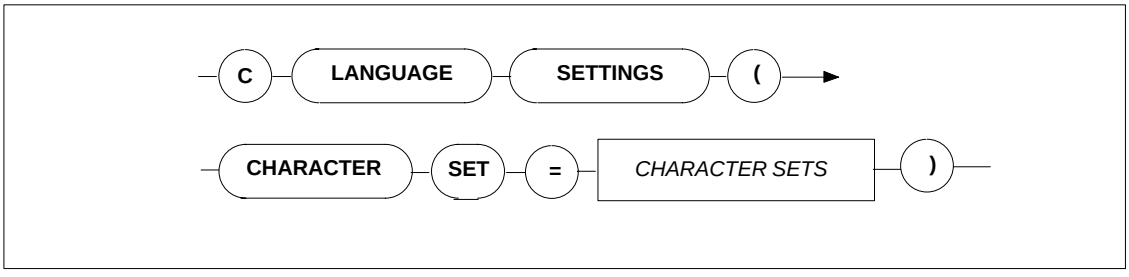


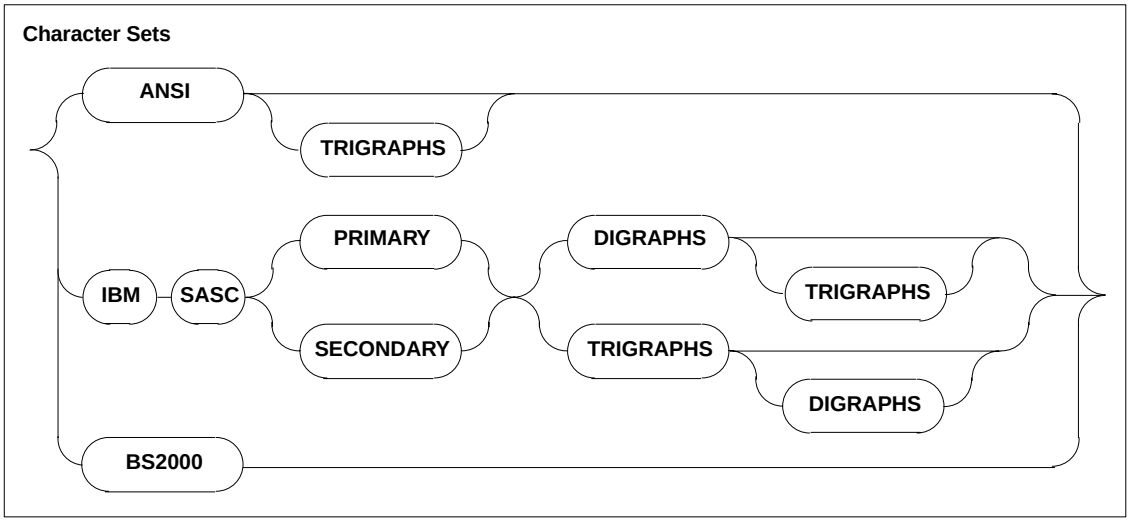
PRECOMPILER C LANGUAGE Settings

Function

To specify the C language environment.

Syntax





Description

These settings enable 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. (Currently not supported)

Limitations

None.

Examples

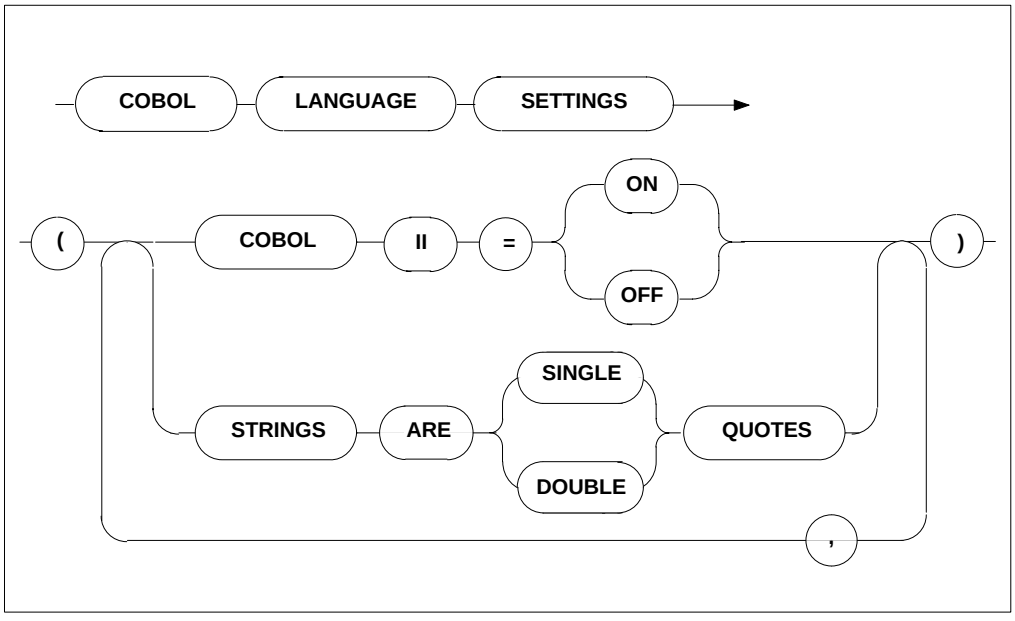
```
PRECOMPILER
BEGIN
     C LANGUAGE SETTINGS ( CHARACTER SET = ANSI TRIGRAPHS)
END

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

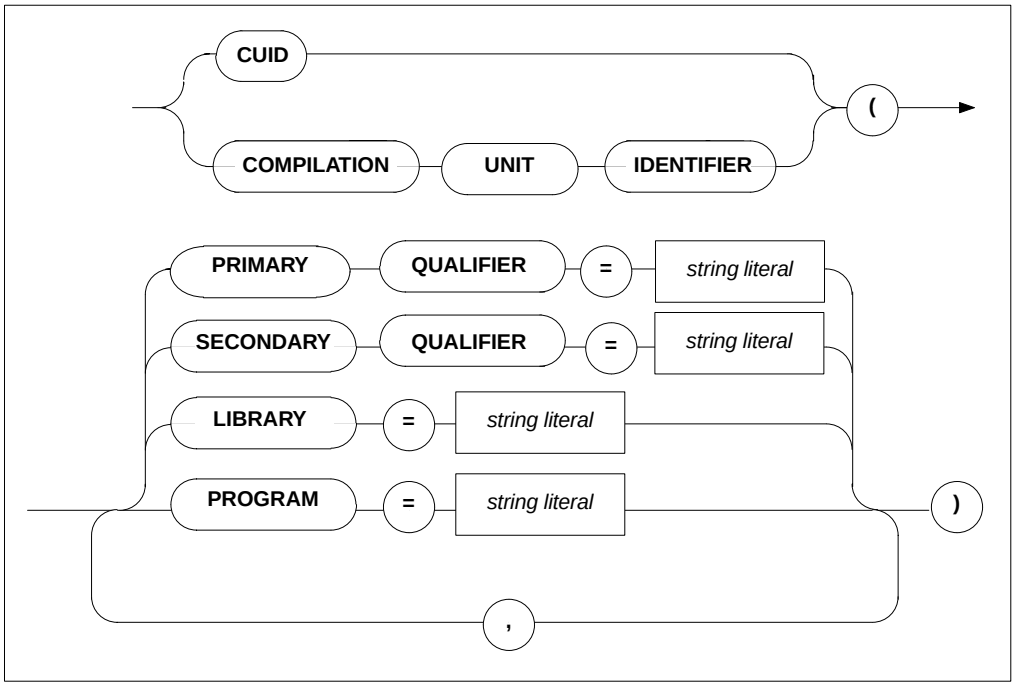
```
PRECOMPILER
BEGIN
    COBOL LANGUAGE SETTINGS ( COBOL II = ON, STRINGS ARE SINGLE QUOTES )
END
```

PRECOMPILER COMPILATION UNIT IDENTIFIER Settings

Function

To set the compilation unit ID. The 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

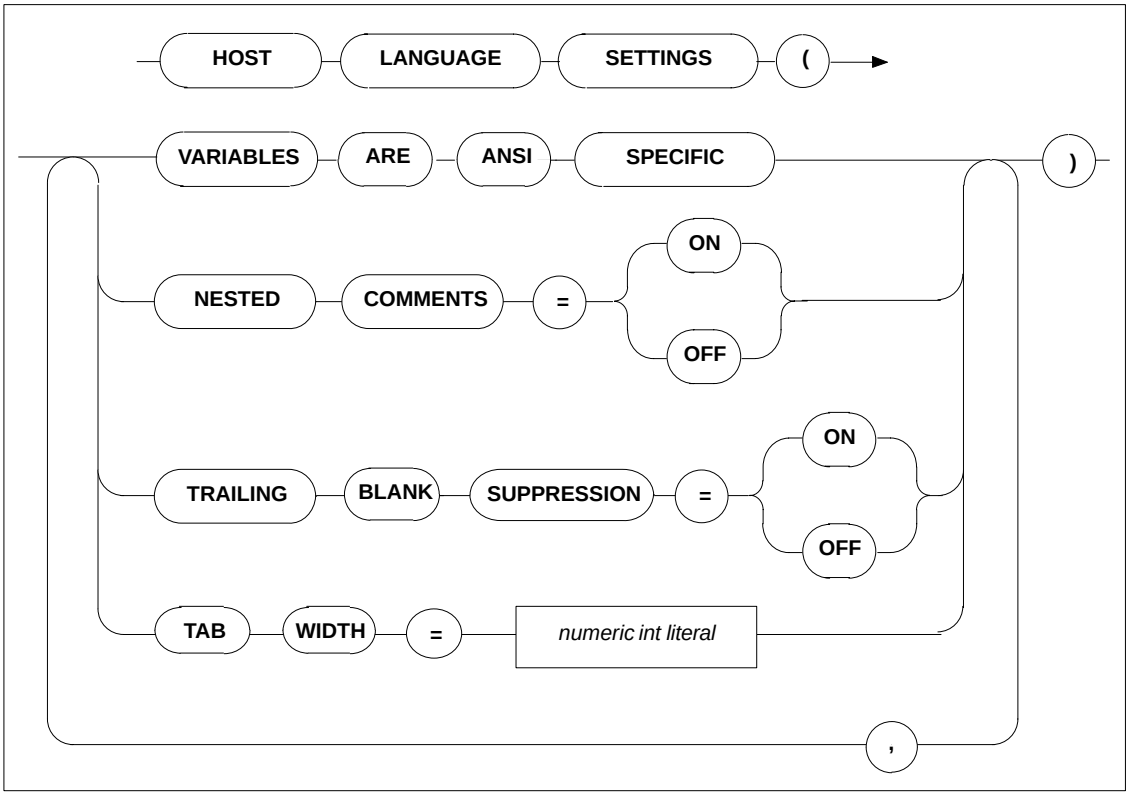
```
PRECOMPILER
BEGIN
     CUID ( LIBRARY = 'ESQ', PROGRAM = 'CR_BASE' )
END

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
```

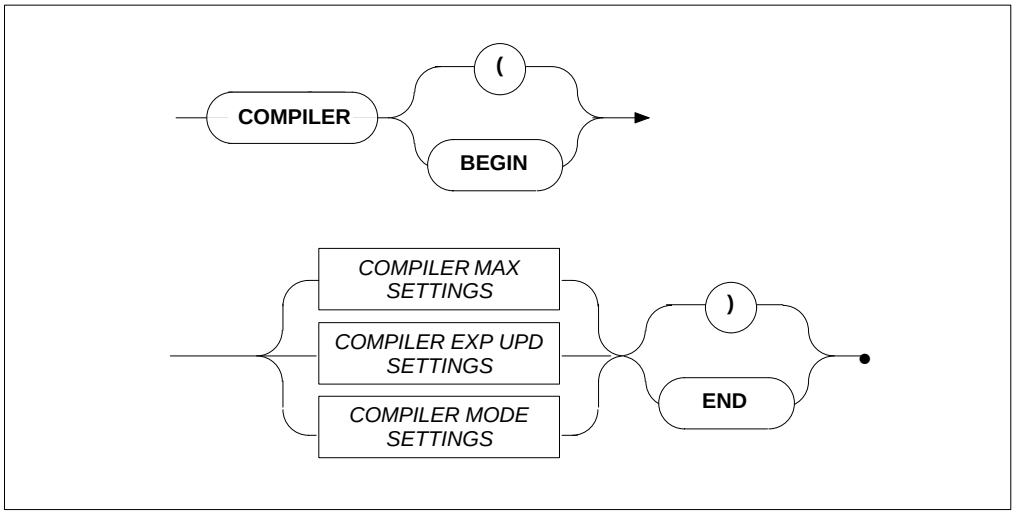
COMPILER Settings

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

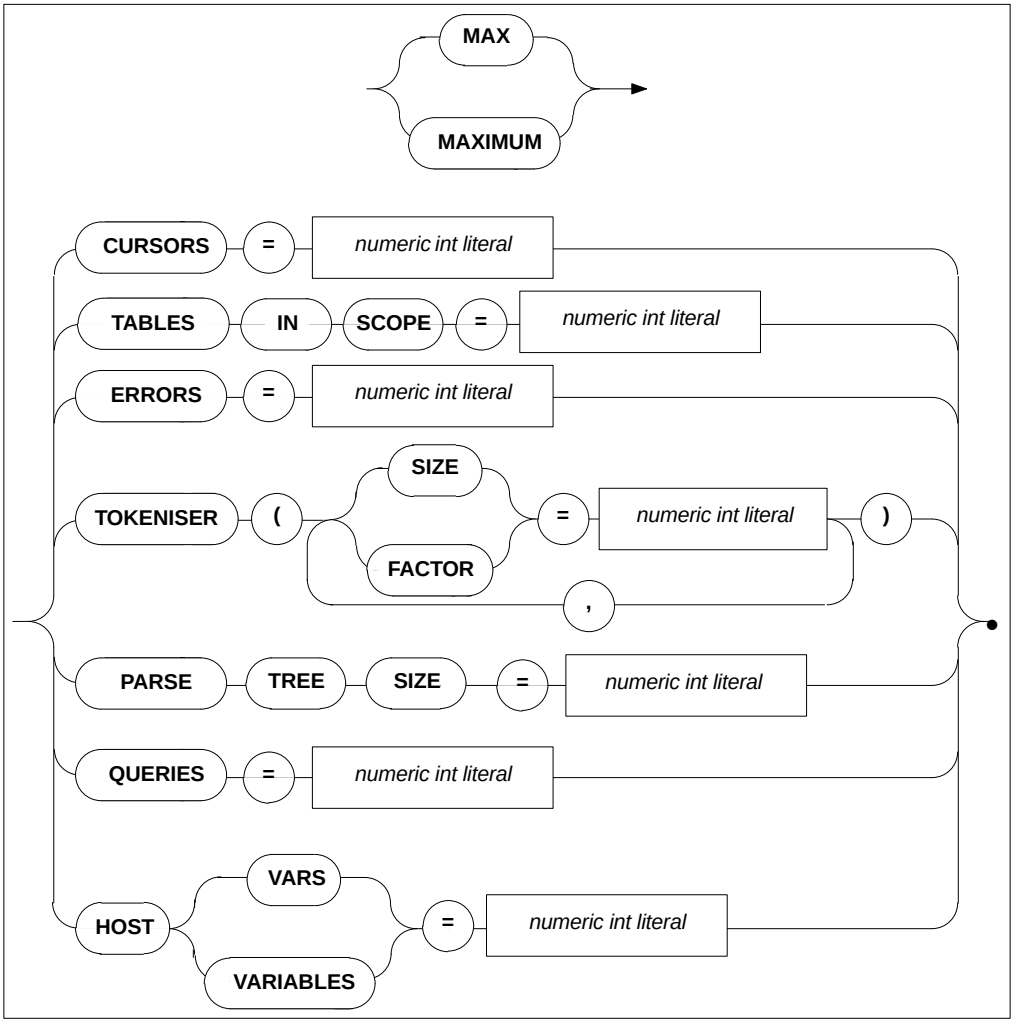


COMPILER MAXIMUM Settings

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. Even though it is dynamically extended (by doubling), choosing a larger initial size may reduce the overhead that is created by allocating these extents. Default (as well as minimum value): 10,000.
TOKENISER SIZE	The size of a tokeniser buffer in bytes during compilation of a statement either at precompile time or runtime. Even though it is dynamically extended (by doubling), choosing a larger initial size may reduce the overhead that is created by allocating these extends.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

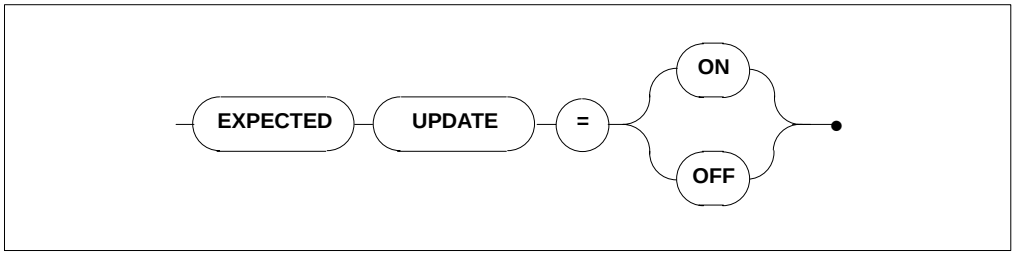
```
COMPILER
BEGIN
    MAX TOKENISER ( SIZE = 32000 )
    MAX CURSORS = 32
    MAX TABLES IN SCOPE = 15
    MAX ERRORS = 50
    MAX QUERIES = 5
    MAX HOST VARS = 32
    MAX PARSE TREE SIZE = 20000
END
```

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

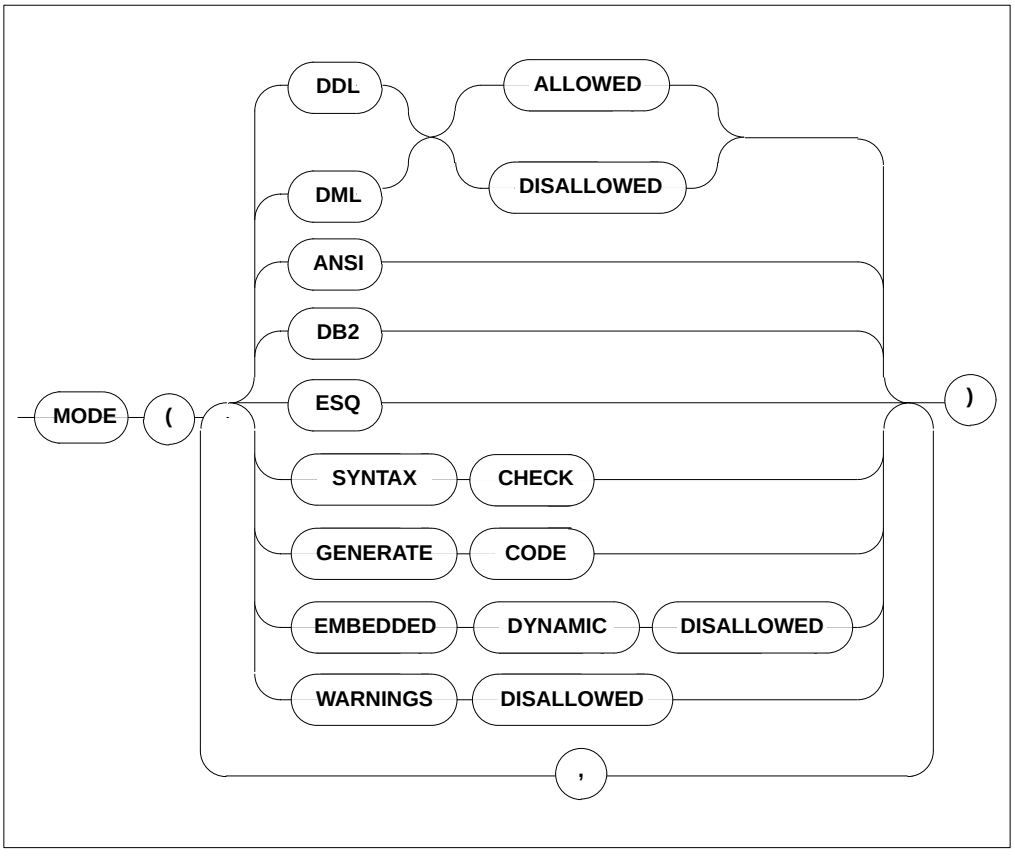
```
COMPILER
BEGIN
     EXPECTED UPDATE = ON
END


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/DML statements can be explicitly allowed or disallowed. There is no special keyword for DCL, but DCL statements are a part of DDL in this particular context.
DML ALLOWED/DISALLOWED	DML statements can be explicitly allowed or disallowed.
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 ANSI standard statements will be permitted and transaction-control statements (COMMIT, ROLLBACK) are issued for a CICS environment. No DB2-specific SQL statements are supported.
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 (ESQ) with the goal to disallowed 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.

In Adabas SQL Server and DB2 modes, however, it is possible to ALLOW DML and DDL at the same time.

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
```

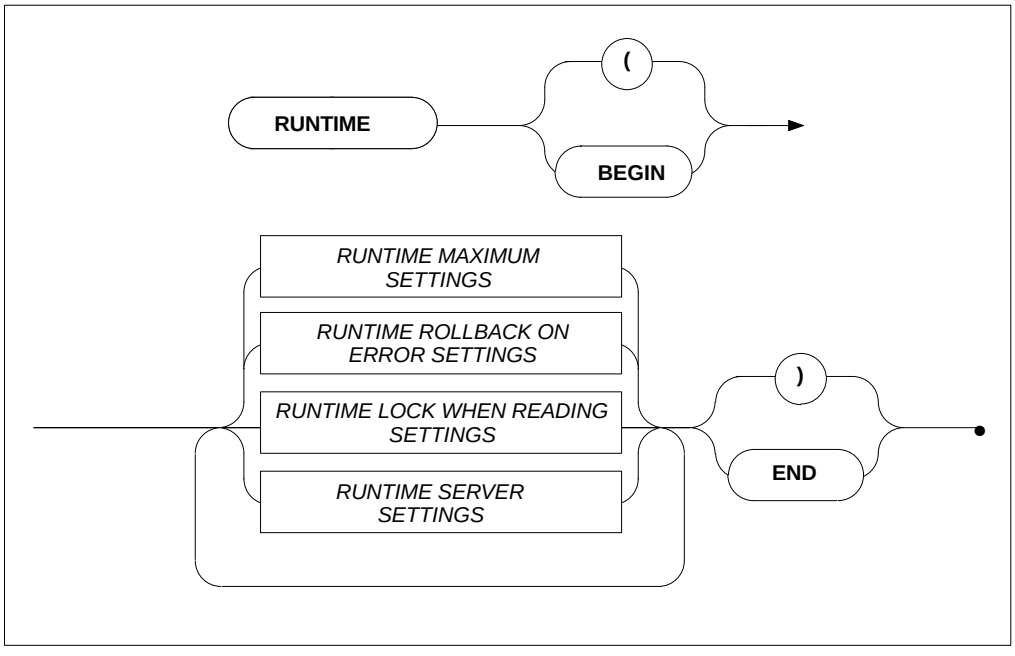
RUNTIME Settings

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

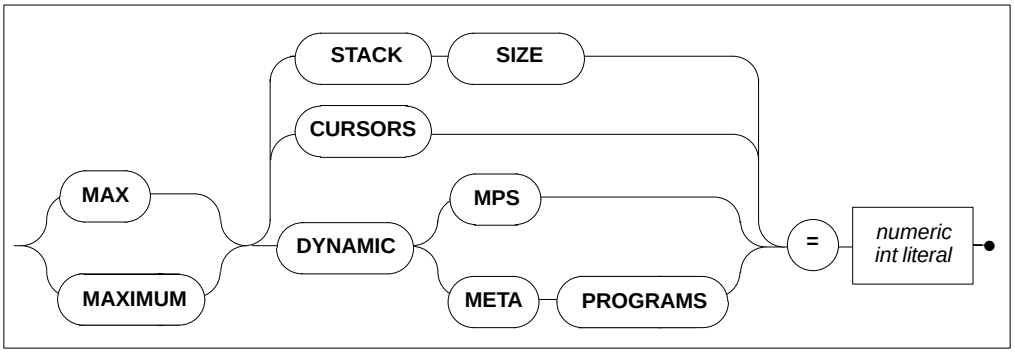


RUNTIME MAXIMUM Settings

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. The stack size may have to be increased higher than the default. The actual requirement may be determined by the count of host variables, indicators, dynamic SQLDA elements and their indicators plus a small number (less than 10) of generally used stack entries. Each stack entry occupies 12 bytes. This setting therefore affects the runtime memory requirement by the factor of 12.
MAX CURSORS	The maximum number of cursors expected to be opened. Default value is 3. This setting affects the runtime memory requirement by a factor of 248.

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. This setting affects the runtime memory requirement by a factor of 36.

Limitations

The higher these values are set, the more memory is required. However, if they are set too low, the Runtime System will terminate with errors.

Examples

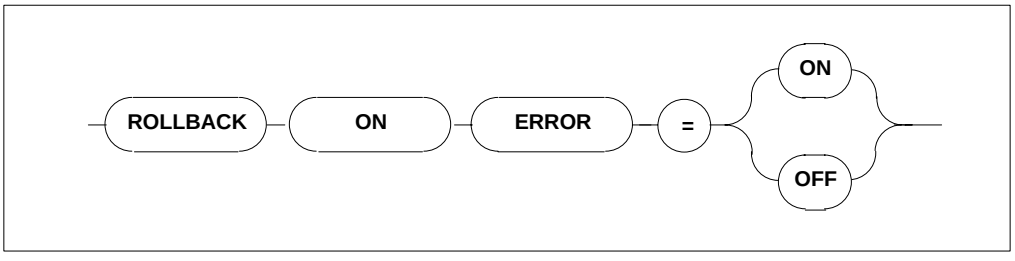
```
RUNTIME
BEGIN
     MAX STACK SIZE = 64 MAX CURSORS = 32
END
```

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

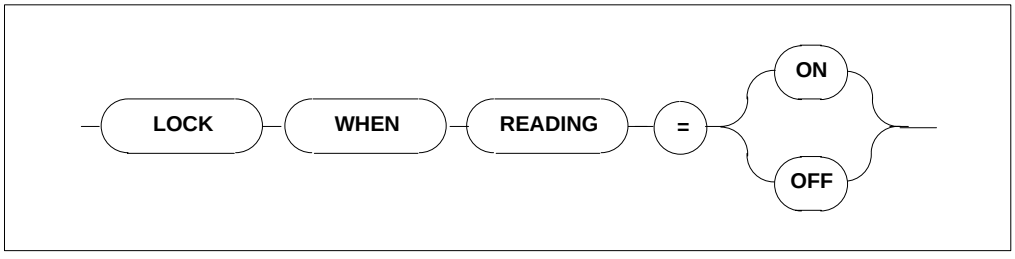
```
RUNTIME
BEGIN
     ROLLBACK ON ERROR = ON
END
```

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

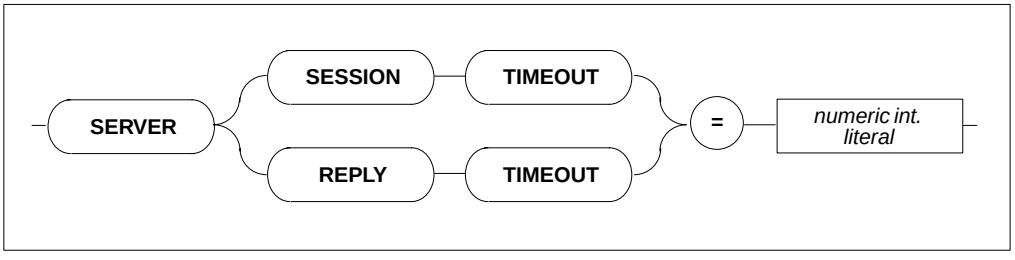
```
RUNTIME
BEGIN
     LOCK WHEN READING = ON
END

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 be set to how long it takes for a timeout in minutes.

Note:
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
```

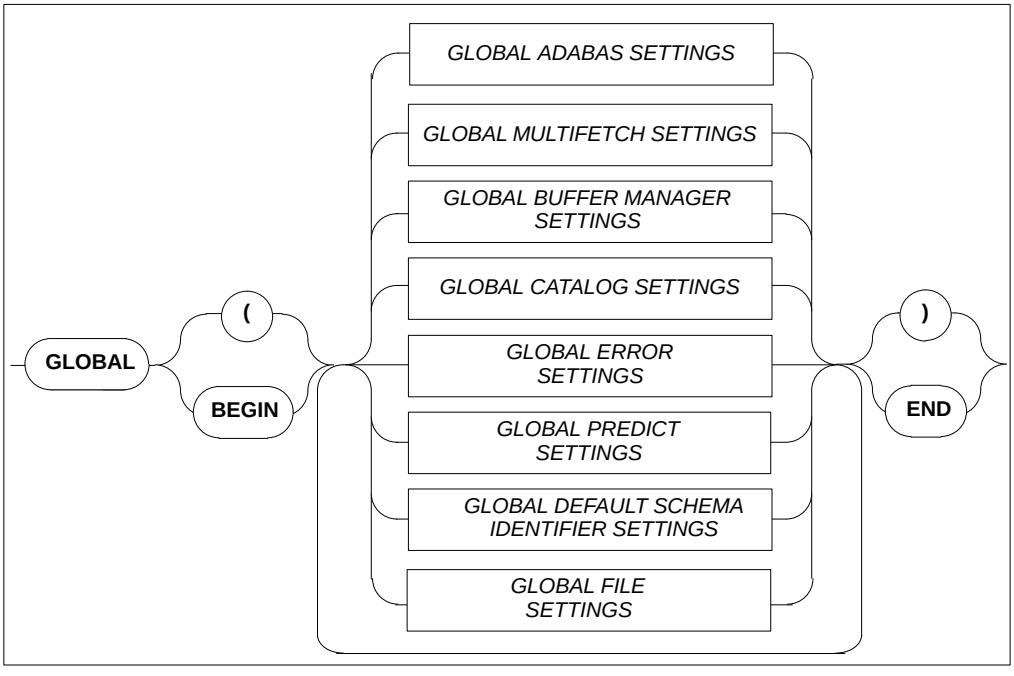
GLOBAL Settings

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

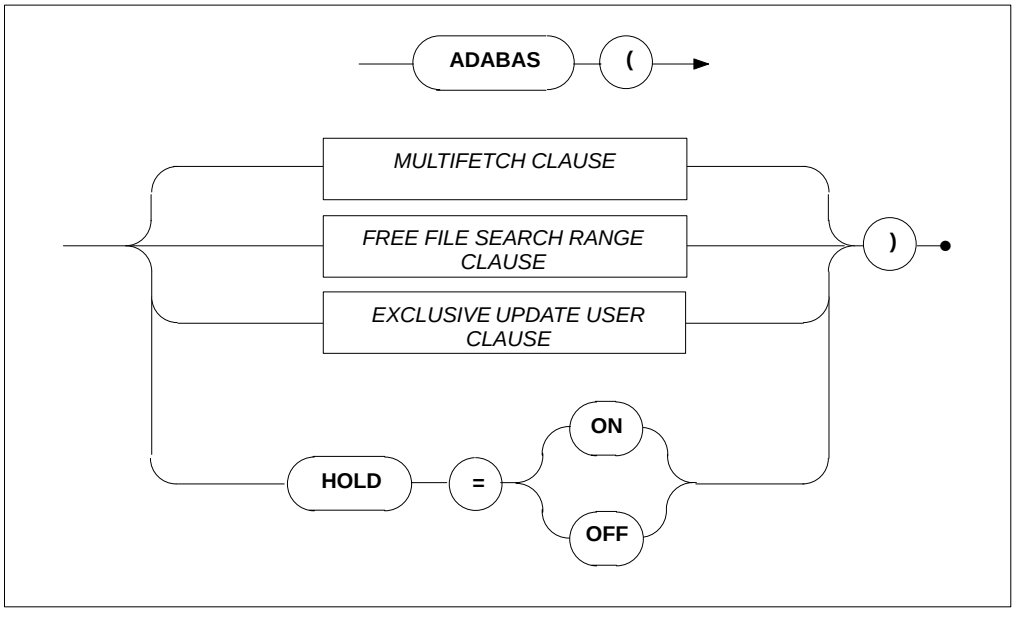

GLOBAL
(
BEGIN
GLOBAL ADABAS SETTINGS
GLOBAL MULTIFETCH SETTINGS
GLOBAL BUFFER MANAGER SETTINGS
GLOBAL CATALOG SETTINGS
GLOBAL ERROR SETTINGS
GLOBAL PREDICT SETTINGS
GLOBAL DEFAULT SCHEMA IDENTIFIER SETTINGS
GLOBAL FILE SETTINGS
)
END

GLOBAL ADABAS Settings

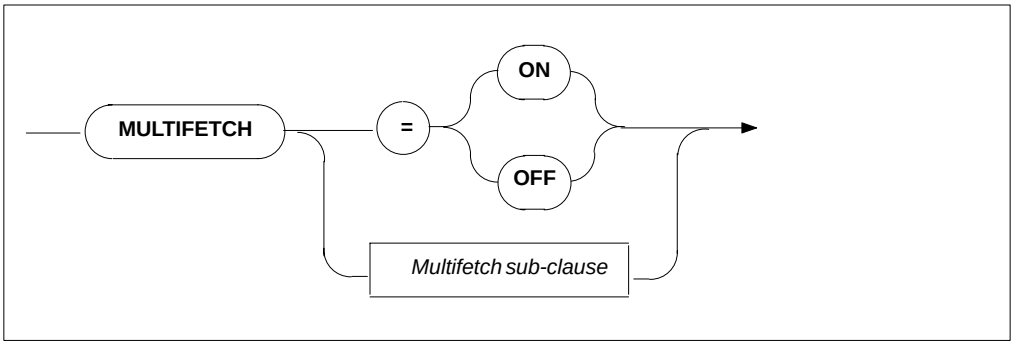
Function

To set Adabas-specific values that are valid across all parts of the system (from precompile time through to runtime).

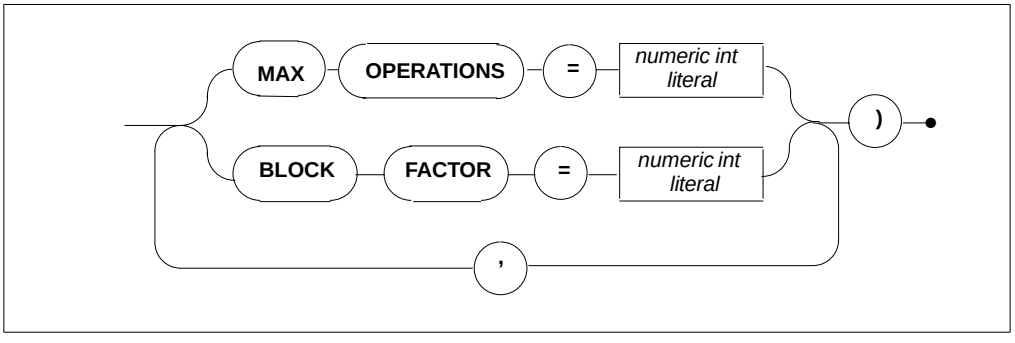
Syntax



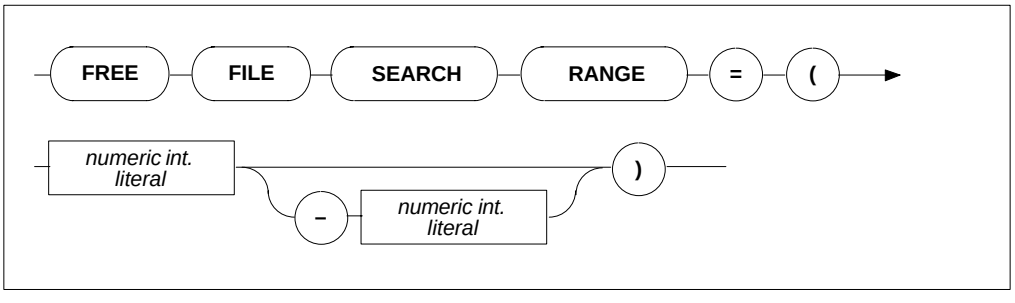
Multifetch Clause



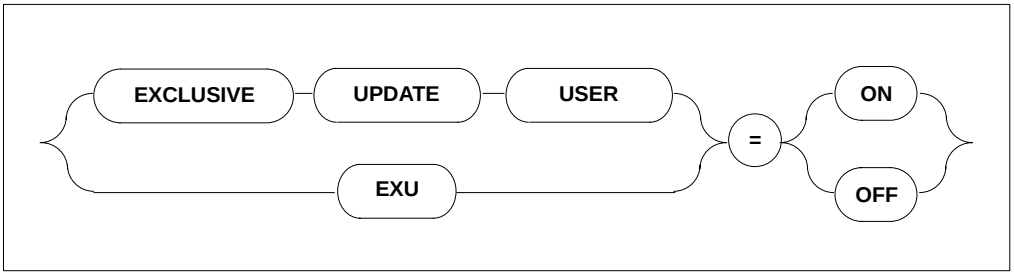
Multifetch Sub-clause



Free File Search Range Clause



Exclusive Update User Clause



Description

Allows the setting of the following Adabas-specific information:

MULTIFETCH ON/OFF	Turns ON/OFF the Adabas MULTIFETCH feature between Adabas and Adabas SQL Server. If multifetching is not used, severe performance losses can be expected. Default is: ON
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 FACTOR	To optimize the data transfer between Adabas SQL BLOCK 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	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	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*, **Adabas SQL Server and other Software AG Products**. The default value is OFF.
HOLD = ON	Indicates that if a record is locked by another user, the application will wait until the record becomes available.
HOLD = OFF	Default value. Indicates that if a record is locked by another user, the application will NOT wait, but will get an appropriate response code.

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
```

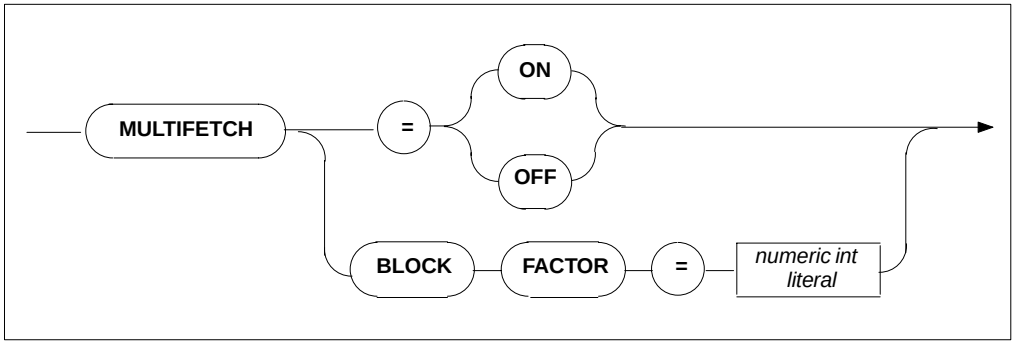
GLOBAL MULTIFETCH Settings

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 Adabas SQL Server with one FETCH statement. Minimum value: 8 Maximum value: 512 Default value: 16

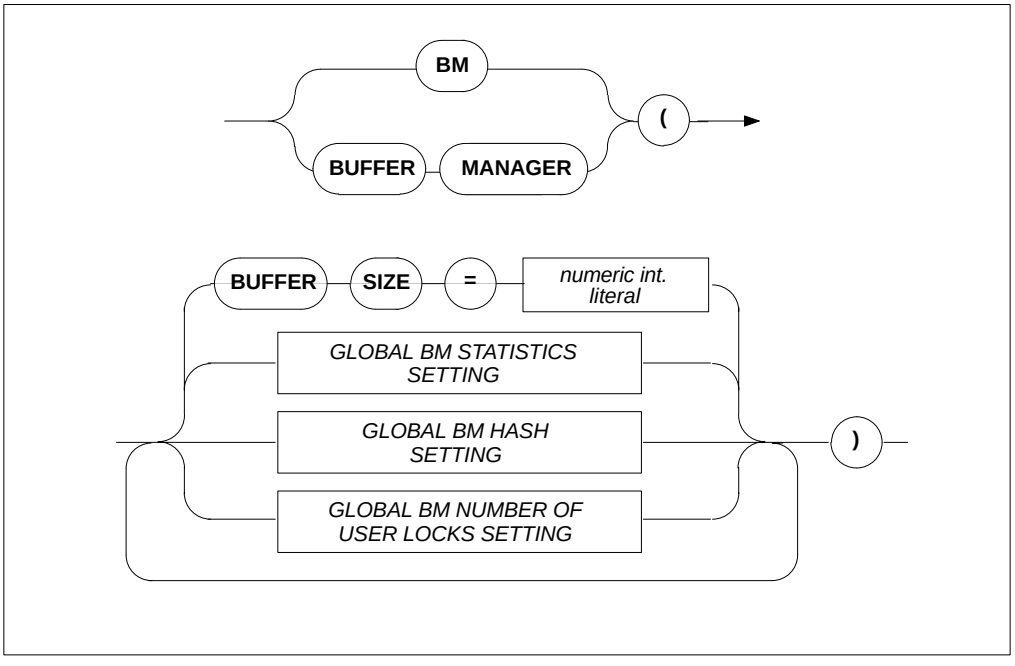
Note:
If a cursor has been defined to be updatable, the use of the MULTIFETCH logic will be turned off automatically by the client.

GLOBAL BUFFER MANAGER (BM) Settings

Function

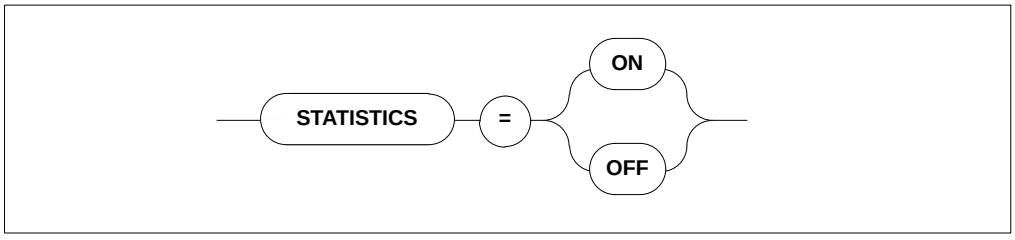
The Adabas SQL Server catalog buffer stores the objects of the catalog. The buffer is a shared memory in a multi-user environment and a process local memory in a single-user environment.

Syntax



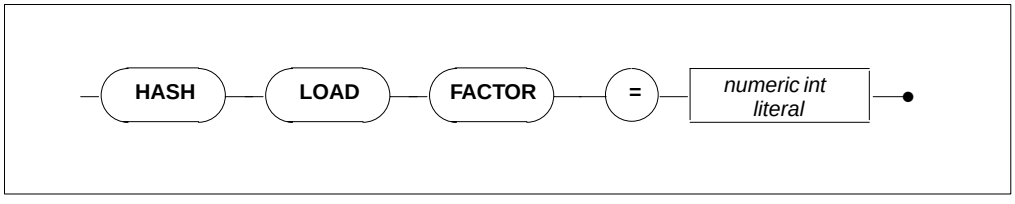
BUFFER SIZE — The size of the catalog buffer in bytes.
Default: 256 KB
Minimum: 32 KB (also depends on the length of the hash table), for details refer to the section **GLOBAL BM HASH SETTINGS** below.

GLOBAL BM STATISTICS SETTING



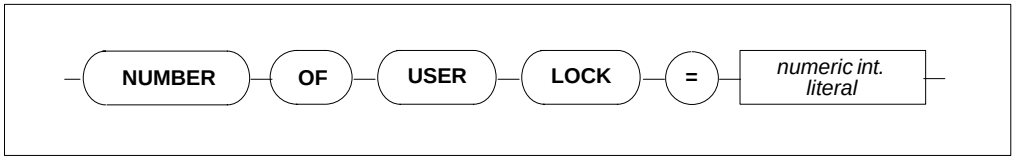
STATISTICS	Defines whether statistics are collected concerning the operation of the buffer manager (slows the system down). Default: OFF

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



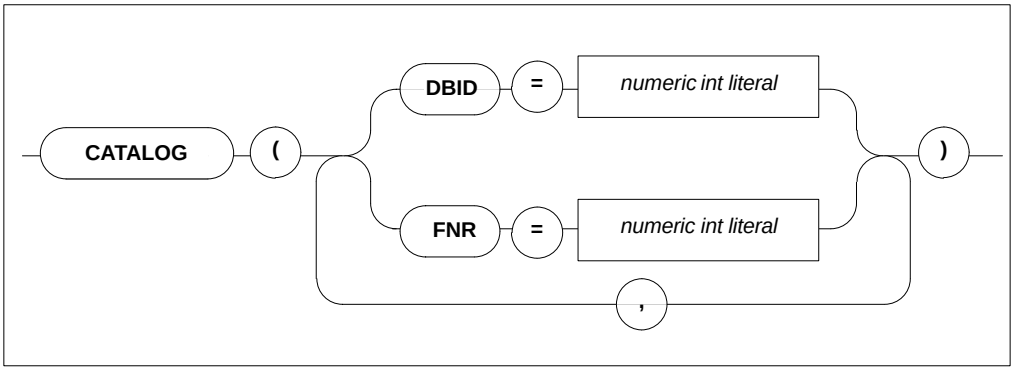
NUMBER OF USER LOCKS	Defines the default number of locks per user if no parameter was specified using BM_R_LOGON(). Default value is 256, minimum is 1 and maximum is 32767.

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

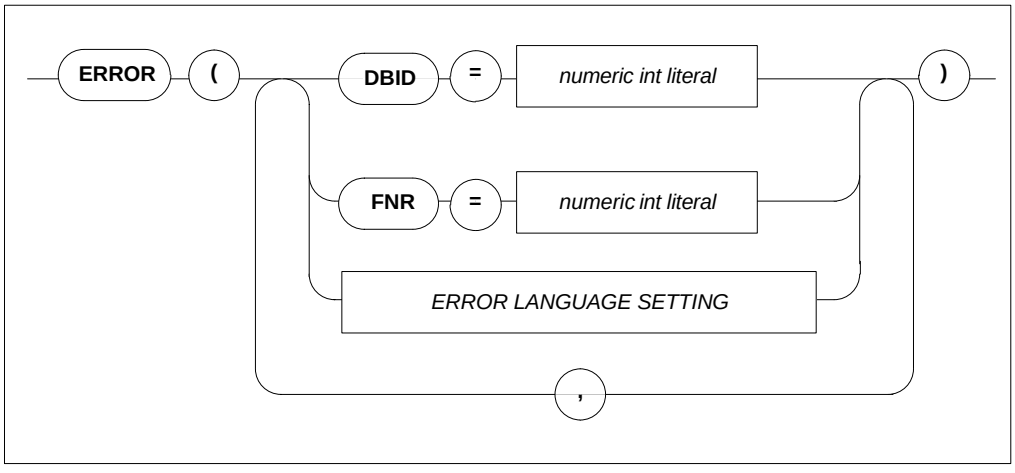
```
GLOBAL
BEGIN
     CATALOG ( DBID = 1, FNR = 13 )
END
```

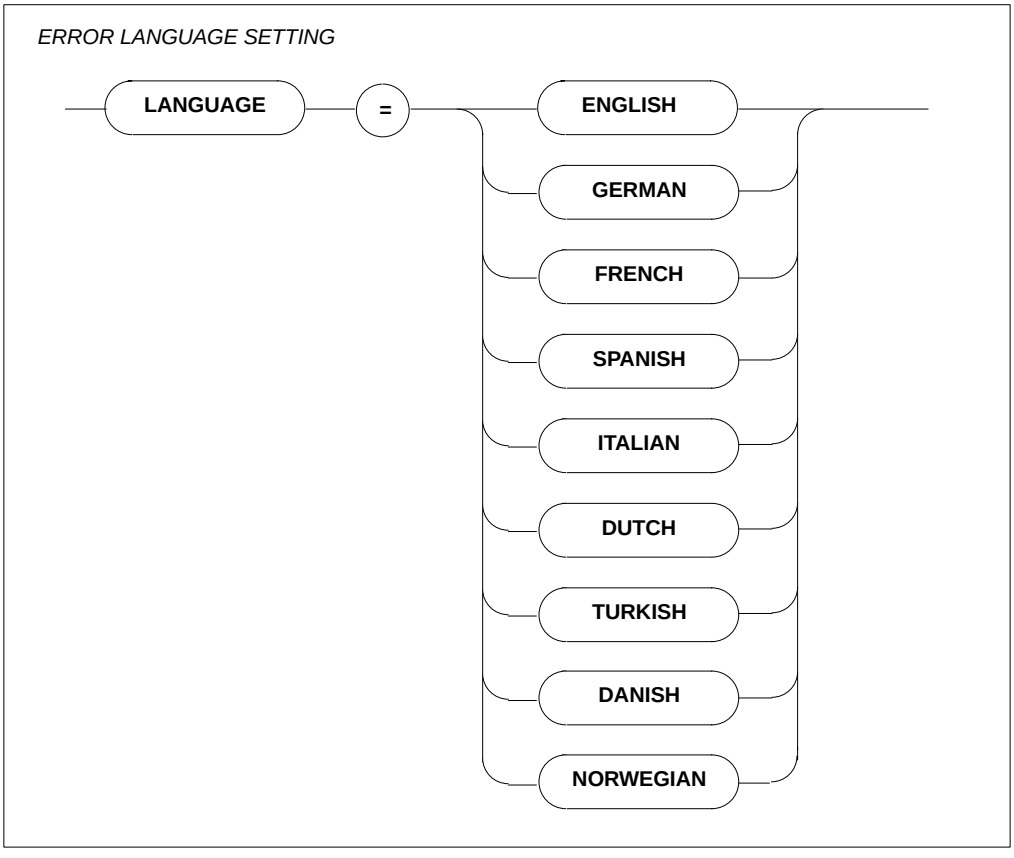
GLOBAL ERROR Settings

Function

To set where the error messages can be found and to specify the desired language.

Syntax




ERROR LANGUAGE SETTING
LANGUAGE
=
ENGLISH
GERMAN
FRENCH
SPANISH
ITALIAN
DUTCH
TURKISH
DANISH
NORWEGIAN

Description

Defines in which Adabas database and which Adabas file the SQL error messages can be found.

Default values are DBID = 1 and FNR = 9 for all operating systems other than UNIX, for which the defaults are DBID = 240 and FNR = 14. Minimum value is 1 and maximum value is 255.

The Global Error Language Setting lists all languages in which error messages are available.

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

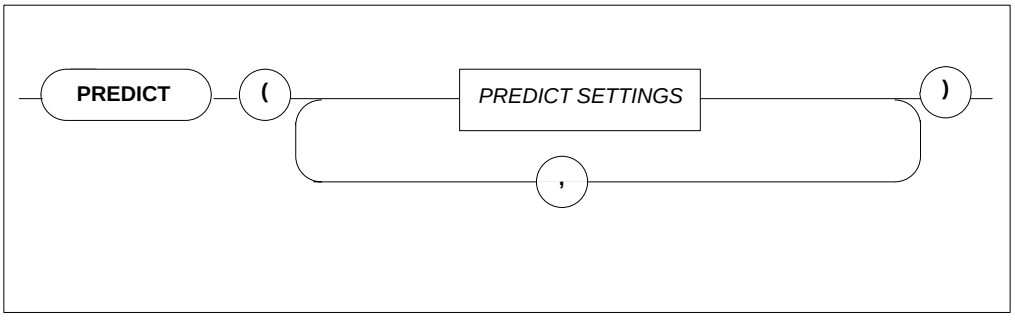
```
GLOBAL
BEGIN
     ERROR ( DBID = 13, FNR = 3 )
END
```

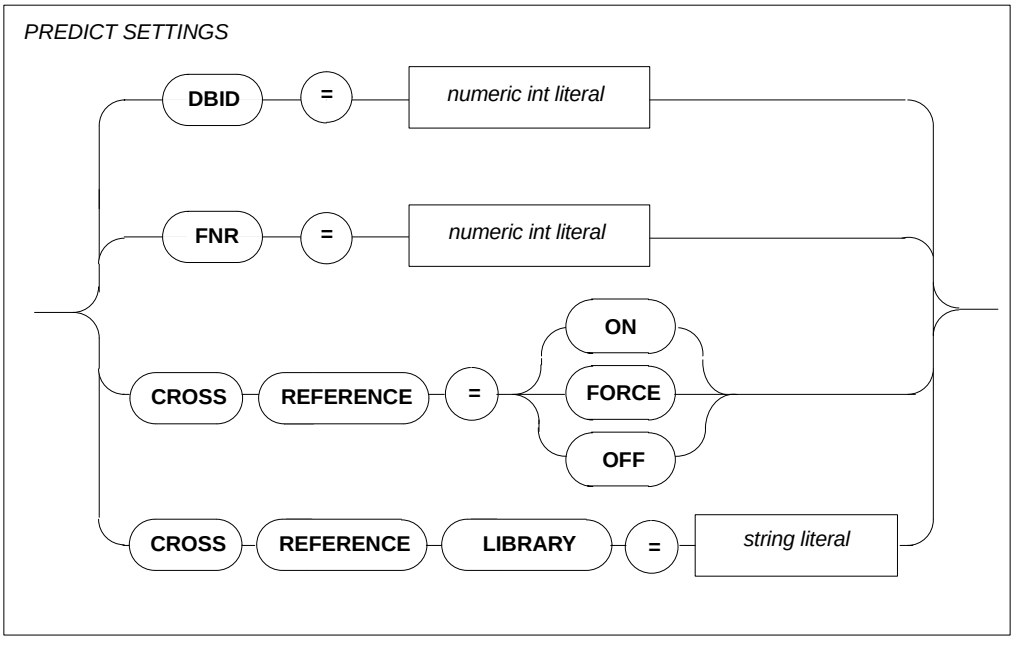
GLOBAL PREDICT Settings

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

DBID and FNR are only necessary in client/server mode.

The maximum length of a Predict Cross Reference Library name is 8 characters.

Examples

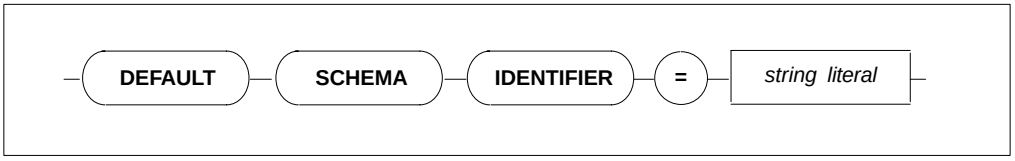
```
GLOBAL
BEGIN
     PREDICT (CROSS REFERENCE = ON)
END
```

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 using 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

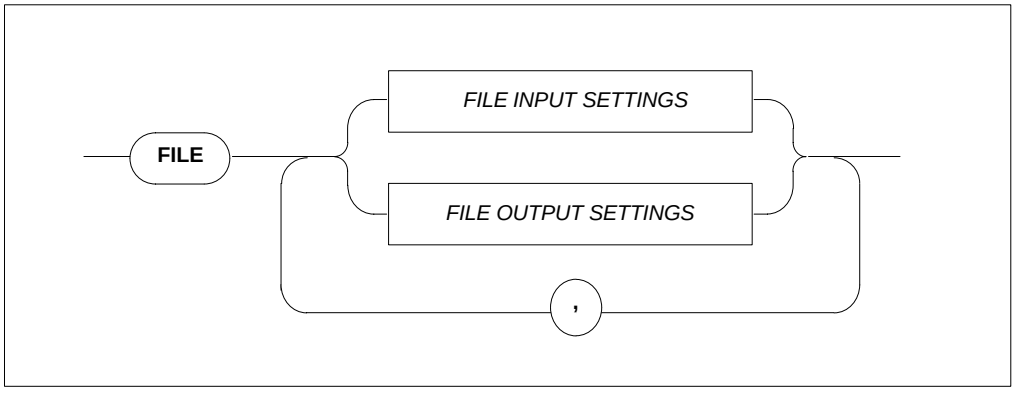
```
GLOBAL
BEGIN
     DEFAULT SCHEMA IDENTIFIER = “ESQ”
END

GLOBAL
BEGIN
     DEFAULT SCHEMA IDENTIFIER = “robert”
END
```

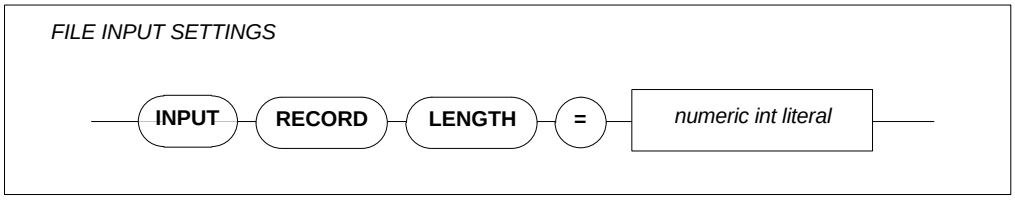
GLOBAL FILE Settings

Function

Sets the input/output record length.

Syntax





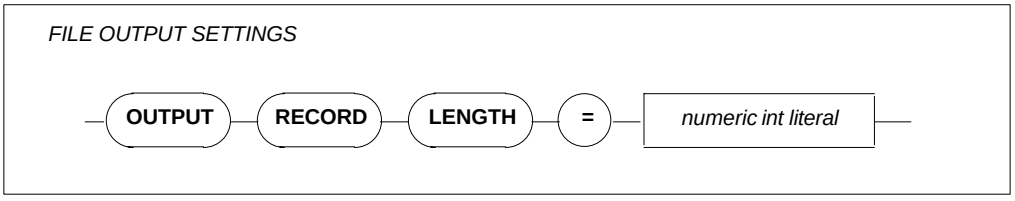
INPUT RECORD LENGTH Maximum record length (integer).

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

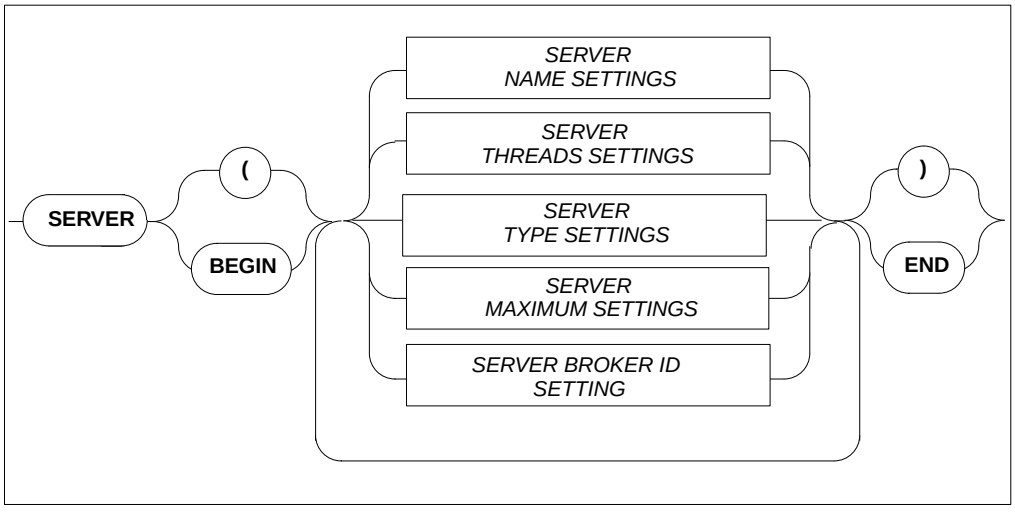
Function

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.

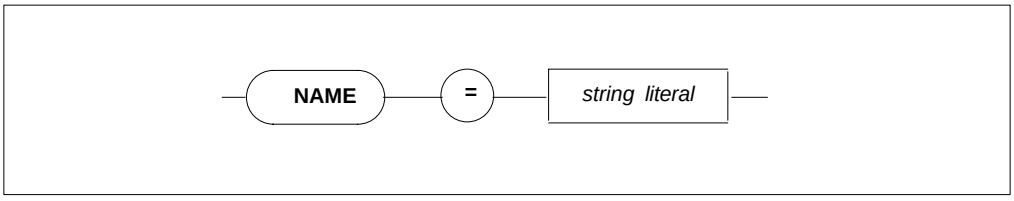
The types of settings for the SERVER section are:

- SERVER NAME SETTINGS
- SERVER THREAD SETTINGS
- SERVER TYPE SETTINGS
- SERVER MAXIMUM SETTINGS
- SERVER BROKER ID SETTING

Syntax

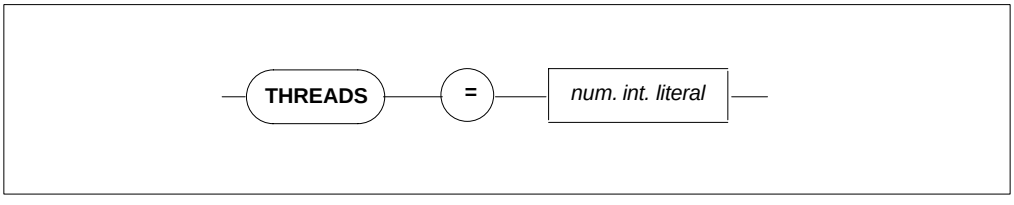


SERVER NAME Setting



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.

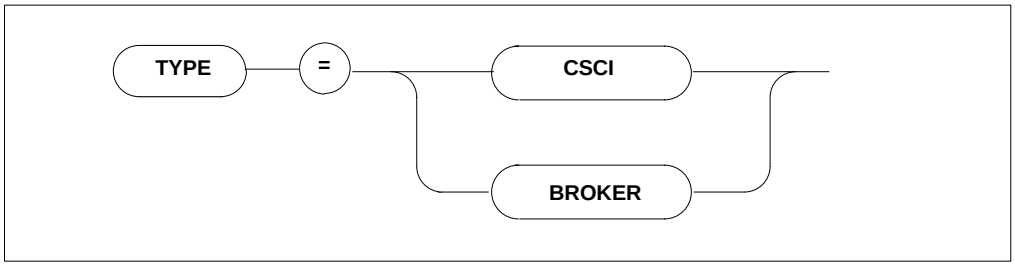
SERVER THREADS Setting



Specifies the number of threads per server. More threads mean more resources required.

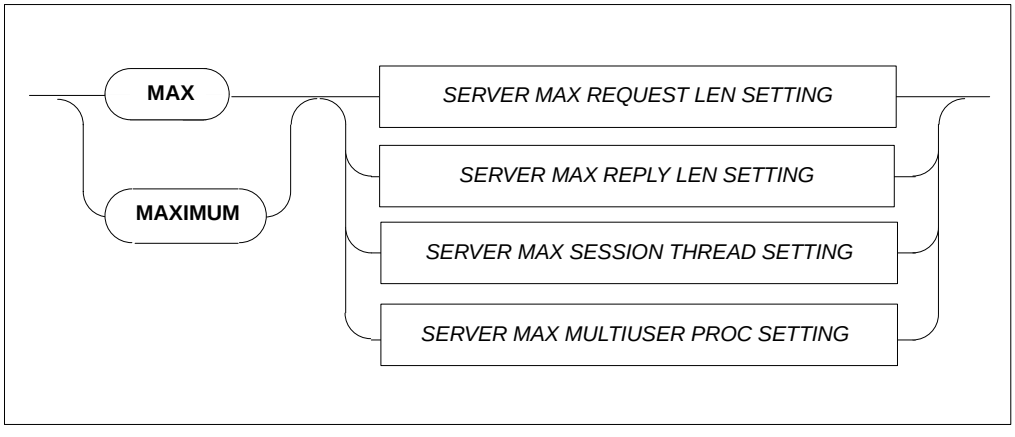
Default value: 5, minimum value: 0.

SERVER TYPE Setting



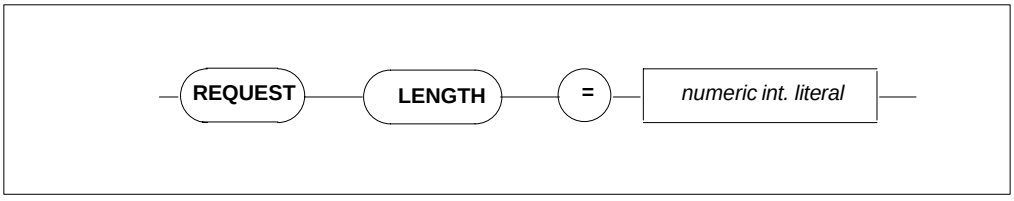
Specifies the type of client-server communications mechanism. Must be set to BROKER if the SERVER BROKER ID Setting is to be specified.

SERVER MAXIMUM Settings



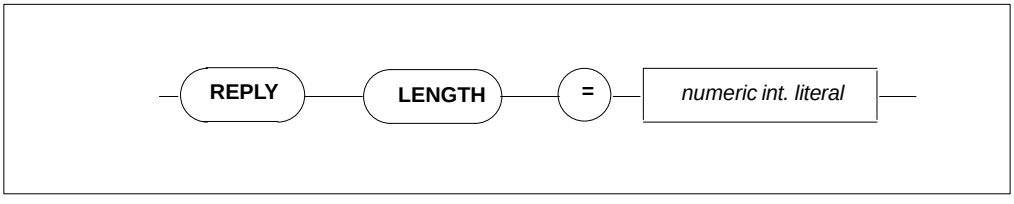
Sets the maximum values for the server.

SERVER MAXIMUM REQUEST LENGTH Setting



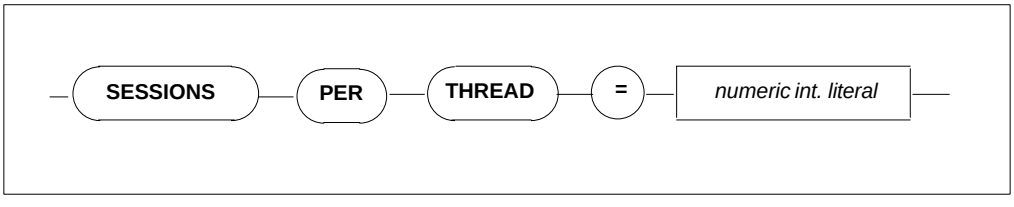
Specifies the maximum request length for the client-server communications. The default value is 8000 bytes, maximum length is 64000 bytes.

SERVER MAXIMUM REPLY LENGTH Setting



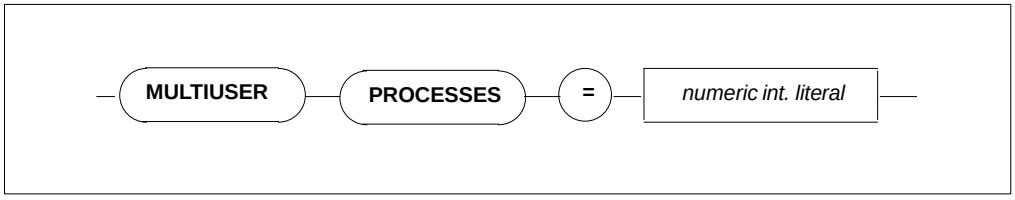
Specifies the reply length for the client/server communications. The default value is 8000 bytes, maximum length is 64000 bytes.

SERVER MAXIMUM SESSION THREAD Setting



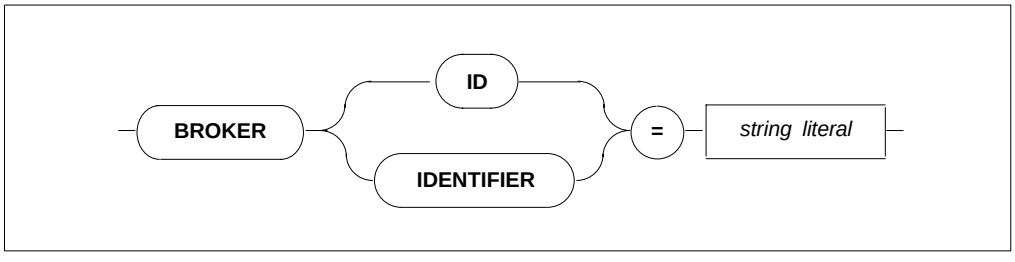
Specifies the maximum number of sessions per thread; each user process is one session. Zero means infinite (∞). For Version 1.4 the maximum number of sessions per thread must be 1.

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 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.

B

APPENDIX B — USEFUL INSTALLATION INFORMATION

The information contained within this section can be useful to the reader during the installation of Adabas SQL Server. Some of this information can be found in other locations, but it has been included here to simplify the reader's search efforts.

This section contains information on the following topics:

- The ESQAID utility is an installation aid, which can be used to customize the example JCL and members to fit the installation requirements. The usage of this utility is not mandatory, but simplifies the tedious editing and replacement of the variable symbols in the sample members.
- Replacement Variables Symbols used in the example JCL and source member, which are included in the Adabas SQL Server sublibrary.
- A list of response codes, which are issued by the CSCI Interface. This list is useful, when trying to troubleshoot the CSCI communication.
- A list of VO sub–codes, which are used in Adabas SQL Server messages.

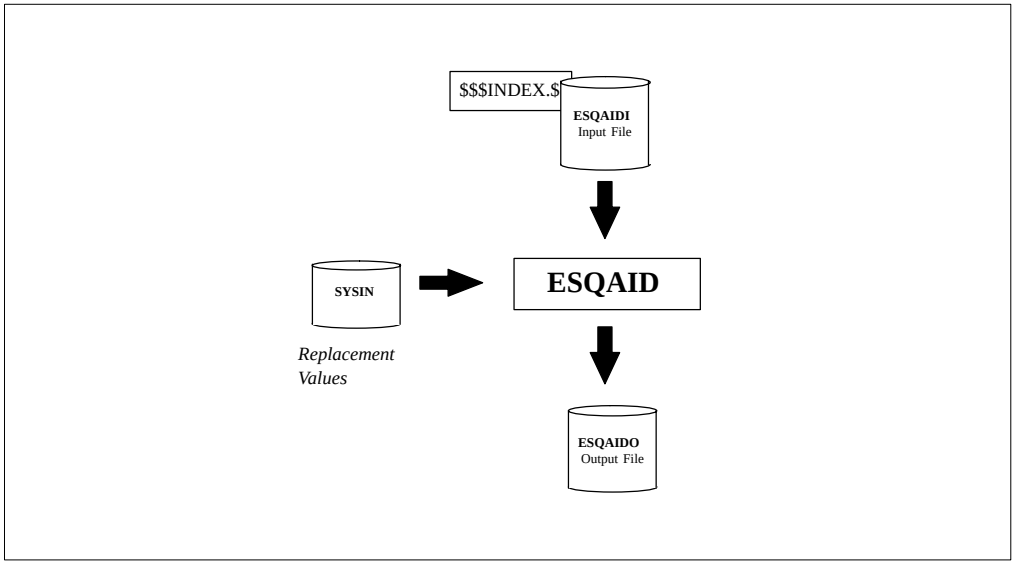
Installation Aid Utility

The installation aid utility (ESQAID) may be used to simplify the installation of Adabas SQL Server. The use of this utility is optional. This utility must not be used during an SMA installation.

ESQAID processes the members in the Adabas SQL Server job sublib substituting commonly used variables with the values which were provided using the SYSIN. The modified members are written to a new sublibrary (ESQAIDO).

The input file (ESQAIDI) is not modified during this operation.

The member **$$$INDEX.$** should not be modified. It contains the list of members in the input dataset, which are to be processed. Only those members included in **$$$INDEX.$** are processed by the ESQAID utility, all other members are ignored.



Installation Aid Utility

Executing the ESQAID

- Edit the example member **J#AID.J** in the Adabas SQL Server sublibrary and update it to match the correct dataset names and parameter values for your site.
- Submit **J#AID.J** after making the relevant modifications.

A list of replacement variable symbols included in **J#AID.J** are listed below. A complete list of replacement variables symbols is provided in the next section.

An evolutionary approach can be used to customize the installation jobs. The ESQAID utility can be used repetitively to make successive changes to the sublibrary.

Remember, changes made within the output file are overwritten by the modified member from the input file.

Replacement Variable Symbols used in member J#AID.J

The following is a list of replacement variable symbols which are used in the member **J#AID.J.** A complete list of all symbols used within the Adabas SQL Server context is provided in the following section of this appendix.

Symbol	Description
$ADADBID	Database Id where the Adabas SQL Server system files are located (see also $ESQDBID).
$ADALIB	Name of the Adabas sublibrary.
$ESQCATF	File number of the first Adabas SQL Server catalog file
$ESQCATF1–3	File numbers of the Adabas SQL Server catalog file(s). The values of $ESQCATF and $ESQCATF1 both refer to the first catalog file. The Catalog file requires 3 contiguous file numbers: *n, n+1, n+2.*
$ESQDBID	Database ID where the Adabas SQL Server system files are located.
$ESQDBNAME	Logical Name of the Database to be used during the catalog generation.

Symbol	Description
$ESQERRF	File number of the Adabas SQL Server message file.
$ESQFREE1–2	Range of File Numbers, which are to be used by Adabas SQL Server when creating tables. Ensure that the Adabas system files are not included in this range.
$ESQLIB	Name of the Adabas SQL Server sublibrary.
$ESQPREFIX	Dataset name prefix of the Adabas SQL Server Files on the Installation Tape (INPL, ERRN, SYS1–4).
$ESQSEC	The value of this parameter is used to activate the Adabas SQL Server security features. Refer to the section **Installing the Server with/without the Security Features** in chapter **Adabas SQL Server Interfaces** for further information. The default value is NO.
$ESQSINGLELIB	Name of the sublibrary created during Phase A. This sublibrary contains the VOPRM module for executing Adabas SQL Server in LINKED–IN (single user) mode.
$ESQSRCE	Name of the Adabas SQL Server sublibrary.
$SORTLIB	Name of the SORT sublibrary. The external SORT is required during the installation and operation of Adabas SQL Server.

Replacement Variable Symbols

The following table lists all replacement variable symbols used in the example JCL and source members, which are delivered with Adabas SQL Server. The ESQAID utility can be used to customize the example members to fit the installation requirements. The ESQAID utility is described earlier in this appendix.

Symbol	Description
$ACCESSID	ACCESS Id number
$ACCESSSVC	ACCESS SVC number
$ADADBID	Adabas DBID (DatabaseId)
$ADADEVICE	Adabas ASSO device type
$ADAFILE	Adabas files procedure
$ADALIB	Adabas sublibrary
$ADAPROC	Adabas procedure
$ADASORTDEVICE	Adabas SORT device type
$ADASORTSIZE	Adabas SORT size
$ADASVC	Adabas SVC number
$ADATEMPDEVICE	Adabas TEMP device type
$ADATEMPSIZE	Adabas TEMP size
$ESGLIB	Entire Service Manager (ESG) sublibrary
$ESGPREFIX	Entire Service Manager (ESG) dataset name prefix
$ESGPROC	Entire Service Manager (ESG) procedure name
$ESGUSERLIB	Entire Service Manager (ESG) user sublibrary
$ESQCATF	File number of first Adabas SQL Server Catalog file
$ESQCATF1	File number of first Adabas SQL Server Catalog file
$ESQCATF2	File number of second Adabas SQL Server Catalog file
$ESQCATF3	File number of third Adabas SQL Server Catalog file
$ESQDBAPASSWORD	Password for the Administration User DBA
$ESQDBID	Database Id where Adabas SQL Catalog file allocated
$ESQDBNAME	Database name assigned to Adabas SQL Server
$ESQERRF	Number of Adabas SQL Server Error Messages file
$ESQFREE1	First file number in the Adabas free file range

Symbol	Description
$ESQFREE2	Last file number in the Adabas free file range
$ESQGTDOUT	ESQGTD output file name
$ESQGTDSYS1	ESQGTD input SYSTRANS file SYS number
$ESQGTDSYSIN	ESQGTD input parameter(s)
$ESQGTDSYSTRANS	ESQGTD input SYSTRANS file
$ESQGTDVOLSER1	ESQGTD input SYSTRANS file volume serial number
$ESQLIB	Adabas SQL Server sublibrary
$ESQLOGINUSER	System login–ID of user
$ESQLOGINPSWD	System login–password of user
$ESQMIGOUT	ESQMIG output file name
$ESQMLIB	Pre–Compiler metaprogram library name key
$ESQMODE	Compiler MODE setting (ESQ!ANSI!DB2)
$ESQPCIN	COBOL Include facility COBOL source (input)
$ESQPCLIB	COBOL Include facility Copybook sublibrary
$ESQPCOUT	COBOL Include facility COBOL modified source (output)
$ESQPREFIX	Adabas SQL Server dataset name prefix
$ESQPROC	Adabas SQL Server procedure
$ESQPROG	Pre–Compiler program name key
$ESQQUOTE	Pre–Compiler COBOL quotes (SINGLE!DOUBLE)
$ESQSCHID	Globals default Schema Id
$ESQSEC	Globals Security Server (ON!OFF)
$ESQSINGLELIB	Adabas SQL Server single–user sublibrary
$ESQSINGLEWORK	Adabas SQL Server single–user work file
$ESQTAPELABEL1	Tape label number (SYS1)
$ESQTAPELABEL2	Tape label number (SYS2)
$ESQTAPELABEL3	Tape label number (SYS3)
$ESQTAPELABEL4	Tape label number (SYS4)
$ESQTAPELABEL5	Tape label number (INPL)
$ESQTAPELABEL6	Tape label number (ERRN)
$ESQTAPEVOLSER	Tape volume serial number

Symbol	Description
$ESQWORK	Adabas SQL Server multi–user work file
$ESQWORKVOLSER	Adabas SQL Server work file volume serial number
$NATLIB	Natural sublibrary
$NATPROC	Natural procedure
$NATURAL	Natural module name
$NETWORKID	CSCI NETWORK node name
$ROLLSYS1	Entire Service Manager (ESG) ROLL file SYS number
$ROLLVOLSER	Entire Service Manager (ESG) ROLL file volume serial number
$SORTLIB	SORT sublibrary
$SORTPROC	SORT procedure
$SORTSYS1	SORT work file 1 SYS number
$SORTSYS2	SORT work file 2 SYS number
$SORTSYS3	SORT work file 3 SYS number
$SORTVOLSER1	SORT work file 1 volume serial number
$SORTVOLSER2	SORT work file 2 volume serial number
$SORTVOLSER3	SORT work file 3 volume serial number
$SORTWORK1	SORT work file 1 file name
$SORTWORK2	SORT work file 2 file name
$SORTWORK3	SORT work file 3 file name
$SYSONE	Work file for verification routines (SYSIPT/SYSPCH)
$SYSONEVOLSER	Work file volume serial number for verification routines
$SYSTWO	Work file for verification routines (SYSIPT/SYSPCH)
$SYSTWOVOLSER	Work file volume serial number for verification routines
$USERLIB	Adabas SQL Server user sublibrary
$USERPROC	Adabas SQL Server user procedure
$VSECATALOG	Catalog name for VSAM files
$VSECATALOG-NAME	Catalog name for VSAM files

CSCI Error Codes

CSCI is the Network Protocoll layer used by Adabas SQL Server to communicate between client and server. The error message format is given as follows:

```
ESQ1450, Th0: CSCI error 60370/59650 on ”L” command ...
```

In the message above the primary code is 60370 and the secondary code is 59650.

All codes are listed below. Some may be self–explanatory, others may require the help of SAG support personnel.

Code	Description
58402	Feature not implemented yet
58410	Global section not attached
58474	Internal error in signal handling
58482	Receive Timeout
58490	Timeout – Link to service obviously lost
58498	Attach to shared memory failed
58506	Detach from shared memory failed
58514	Illegal parameter in SADM
58522	Wait on semaphore failed
58530	Release of semaphore failed
59130	A filename, to store the memory dump in, in use
59138	Undefined PS typ detected – Internal error
59146	Error while accessing the dump file
59154	No shared memory available
59162	General IO Error
59170	Error in message handling system
59178	Error in Module sadm detected
59186	Memory already exists – Can’t be created
59194	Memory doesn’t exist
59202	Memory attached

Code	Description
59210	Deadlock situation occured
59218	Memory can’t be deleted, there are other users in system
59226	Error in Module adm detected
59234	ADM Table not initialized
59242	Service is not available
59250	Internal datastructure corrupted
59258	Conflict with service initialization value
59266	No free adm entry available
59274	Internal parameter fault
59282	Number of allowed replicates exceeded
59290	Service already exists
59298	Remove operation inconsistent
59306	Pid not available
59314	Number of parallel connections exceeded
59322	Connection already exists
59330	Connection doesn’t exist
59338	Multiple termination of the replicate
59346	Internal undefined returncode
59354	Serverprocess inactive
59362	Illegal server typ CNOS, CNLS, CMPS
59370	Conflict with a compatibility mode service
59378	No service entry found
59386	No replicate entry found
59394	No connection entry found
59402	Error in Module ab detected
59410	No space into the attached buffer
59418	No free request queue entry
59426	Internal undefined returncode

Code	Description
59434	Request queue entry is not in use
59442	Attached buffer not initialized
59450	Error in Module ada0 detected
59458	Shared memory not attached
59466	Ada0 queues not opened
59474	Append/Insert without dequeue
59482	Module Ada0 not initialized
59490	Internal parameter fault
59498	Buffer overflow
59506	Adress queue overflow
59514	Message queue open call failed
59522	Message queue not opened
59530	General IO error
59538	Partnerprocess inactive
59546	Internal error – Datastructure corrupted
59554	Ada0 is already opened
59562	Error in module cslnk detected
59570	Option value in the CB undefined
59578	Function value in the CB undefined
59586	Internal datastructure corrupted
59594	Illegal CB syntax
59602	Desired feature actually not supported
59610	Illegal service type
59618	Maximum number of connections not allowed
59626	Impossible situation PANIC
59634	Connection doesn’t exist
59642	Connection already terminated
59650	New table active – Connection lost

Code	Description
59658	Process not active
59666	Replybuffer to big
59674	Receivebuffer to small
59682	Contextindicator not defined
59690	Structure level of database not compatible
59698	Error in NET–WORK context detected
59706	NET–WORK not available
59714	NET–WORK buffer exceeded
59722	NET–WORK link disconnected
59730	Error in module csnlnk detected
59738	Undefined type symbol
59746	ASCII/EBCDIC conversion error
59754	Integer conversion error
59762	Illegal function parameter – Internal error
59770	Error in the open format
59778	Error in the accept format
59786	Error in the request format
59794	Error in the reply format
59802	Illegal returncode – Internal error
59810	Illegal conversion type
59826	Illegal integer length just 2 or 4 bytes
59834	Illegal float format
59818	Buffer overflow – Buffer for conversion too small
59842	CSCI connection establishment rejected by
59850	CSCI invalid function code
59858	CSCI function not available
59866	CSCI invalid parameter

Code	Description
59874	CSCI invalid connection ID
59882	CSCI invalid server ID
59890	CSCI message truncated
59898	CSCI invalid mandatory or optional parameter
59906	CSCI to many outstanding requests, no more resources
59914	CSCI to many connections, no more resources
59922	CSCI ACCEPT or REJECT expected
59930	CSCI REPLY expected
59938	CSCI internal – initialization of internal blocks failed
59946	CSCI internal – initialization of SCBs failed
59954	CSCI internal – initialization of RCBs failed
59962	CSCI internal – init of CCBs failed
59970	CSCI to many server, no more resources
59978	CSCI internal – max number of server excee
59986	CSCI internal – server ID mismatch
59994	CSCI asynchronous mode not supported
60002	CSCI mode not valid
60010	CSCI invalid version number as IDENT
60018	CSCI invalid length
60026	CSCI invalid buffer address
60034	CSCI invalid request ID
60042	CSCI invalid type
60050	CSCI requested type not supported
60058	CSCI requested mode not supported
60066	CSCI invalid number of replicates
60074	CSCI invalid maximum number of connections

Code	Description
60082	CSCI internal – io message file failed
60090	CSCI internal – call of communication IF
60098	CSCI encode/decode failed
60106	CSCI server not active
60114	CSCI reply generated by CSCI
60122	CSCI server terminated
60138	Invalid index value
60154	Type not defined – Global section may be corrupt
60162	The index doesn't refer to a service entry
60170	The index doesn't refer to a replicate entry
60362	NET–WORK node not known
60370	CSCI Error
60378	CSCI Warning
60386	CSCI Information
60394	CSCI Error in the remote system
60402	CSCI CS_LAST option not supported
60410	CSCI invalid timeout value
60418	CSCI CS_BOTH option not supported
60426	CSCI invalid mandatory option1 parameter
60442	CSCI connection canceled by service entity
60450	CSCI no more resources
60458	CSCI connections timeout
60434	CSCI invalid con_max parameter
60466	CSCI data too long – truncated
60474	CSCI failure to load stub CINIT
60482	CSCI failure to load stub CSICSCI

Response Codes Set by VO

The following table lists the response codes and sub–codes issued by the Virtural Operating System (VO) components. The sub–codes are sometimes used in Adabas SQL Server Messages:

Response Code	VO Sub–Code	Description
0	–0	successful completion
4095	–1	Function not implemented
4094	–2	operating system error
4093	–3	not found
4092	–4	no more space available
4091	–5	not allocated
4090	–6	already existing
4089	–7	no message of of type pres
4088	–8	lock not granted
4087	–9	parameter error
4086	–10	not opened
4085	–11	input truncated
4084	–12	overflow on VO tables
4083	–13	undefined id
4082	–14	already opened
4081	–15	openmode mismatch
4080	–16	time out occurred
4079	–17	execution denied
4078	–18	data too long
4077	–19	access error on file
4076	–20	part of file name no dir
4075	–21	file name error
4074	–22	map address different
4073	–23	missing initialisation
4072	–24	Adabas utility error

Response Code	VO Sub–Code	Description
4071	–25	end of file encountered
4070	–26	Missing DDCARD statement
4069	–27	Missing DDKARTE statement

INDEX

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

Z