

Entire Access for TCP/IP Manual for UNIX

Manual Order Number: OSX421-030UNIX

This document applies to Entire Access for TCP/IP Version 4.2.1 for UNIX 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

© April 2001, Software AG

All rights reserved

Printed in the Federal Republic of Germany

Software AG and/or all Software AG products are either trademarks or registered trademarks of Software AG. Other products and company names mentioned herein may be the trademarks of their respective owners.

TABLE OF CONTENTS

PREFACE	1
What is Entire Access for TCP/IP?	1
Using this Manual	3
1 HOW ENTIRE ACCESS FOR TCP/IP WORKS	5
Accessing Data Sources from Clients	6
Local Data Access	6
Remote Data Access Using TCP/IP	7
Remote Data Access Using Third-Party Network Products	8
2 INSTALLING A SERVER UNDER UNIX	9
Before You Install	9
Client and Server Version Levels	9
Hardware and Operating System Requirements	10
Other Software Requirements	11
UNIX Compiler Support	11
Database Systems and Versions Overview	11
Database System Preparation	12
Environment Variables	13
Installation Procedure	15
Step 1 Log In at the UNIX System Prompt	15
Step 2 Generate the Environment File	15
Step 3 Select the Database Drivers (Natural for UNIX Only)	17
Step 4 Link the Server Daemons on Server Machines	19
Step 5 Bind the Entire Access for TCP/IP Package	20
Step 6 Establish ADABAS SQL Server Run-time Parameters	21
The Entire Access Directory Structure Under UNIX	22

3	INSTALLING A CLIENT UNDER UNIX	23
	Before You Install	23
	Client and Server Version Levels	23
	UNIX Compiler Support	23
	Hardware and Operating System Requirements	24
	Other Software Requirements	24
	Installation Procedure	25
	Step 1 Log In at the UNIX System Prompt	25
	Step 2 Generate the Environment File	25
	Step 3 Select the Database Drivers (Natural for UNIX Only)	27
	Step 4 Relink Natural on UNIX Client Machines	29
4	INSTALLING A CLIENT UNDER WINDOWS 98, 2000, OR NT	31
	Before You Install	32
	Client and Server Version Levels	32
	Client Hardware and Operating System Requirements	32
	Other Software Requirements for Clients	32
	Installation Procedure	33
	Installation CDs	33
	Installation Steps	33
5	INSTALLING A CLIENT UNDER AXP OPENVMS	35
	Before You Install	35
	Operating System Requirements	35
	TCP/IP Connectivity	35
	Supported Application Environment	35
	SAGBASE Requirement	35
	The Installation CD	36
	Installation Prerequisites	36
	Installation Kit Structure	37
	Example of an Initial Installation	38

Installation Procedure	42
Step 1 Run the VMSINSTAL Procedure	42
Step 2 Check the Directory Structure	45
Step 3 Perform Startup and Verification	46
6 DEFINING DATA SOURCES TO NATURAL	49
Natural Global Configuration File	49
UNIX	50
Windows 98, Windows 2000, and Windows NT	51
AXP OpenVMS	52
Local Client Connect Strings	53
Syntax for Local Client Connect Strings	53
Sample Local Client Connect Strings	54
Remote Client Connect Strings	55
Syntax for Remote Client Connect Strings	55
Sample Remote Client Connect Strings	56
7 GENERATING AND LOADING NATURAL DDMs	57
Using the Natural DDM Editor	57
Generating and Loading DDMs On OpenVMS	57
Generating DDMs	58
Loading the Generated DDMs	59
8 SUPPLYING USER ID AND PASSWORD	61
UNIX Clients	62
Before Starting the Natural Session	62
Database Authentication	62
Operating System Authentication	62
Windows 98, Windows 2000, Windows NT Clients	63
Before Starting the Natural Session	63
Database Authentication	63
Operating System Authentication	63

Entire Access for TCP/IP Manual for UNIX

AXP OpenVMS Clients	64
Before Starting the Natural Session	64
Database Authentication	64
Operating System Authentication	64
APPENDIX A – USING NATURAL WITH ENTIRE ACCESS FOR TCP/IP	65
Generating Natural DDMs	66
Setting Profile Parameters	66
The OPRB Parameter	66
Natural DML Statements	67
BACKOUT TRANSACTION	68
DELETE	68
END TRANSACTION	69
FIND	70
HISTOGRAM	71
READ	71
STORE	72
UPDATE	73
Natural SQL Statements	75
DELETE	75
INSERT	75
PROCESS SQL	76
SELECT	82
SELECT SINGLE	82
UPDATE	83
Flexible SQL	83
Examples:	84
RDBMS-Specific Requirements and Restrictions	85
Case-Sensitive Database Systems	85
SYBASE and Microsoft SQL Server	85

Table of Contents

Data Type Conversion	88
ADABAS D	89
ADABAS SQL Server	90
DB2	91
INFORMIX	92
OpenIngres	93
ORACLE	94
SYBASE and Microsoft SQL Server	95
Date/Time Conversion	96
Conversion Tables	97
Obtaining Diagnostic Information	99
APPENDIX B – INTERSOLV ODBC	101
“chatr” Requirement for the ODBC Server	102
“chatr” Requirement for the ODBC Client	103
“odbc.ini” Examples for the ODBC Server	104
Sample SYBASE ODBC Variables	106
Sample INFORMIX ODBC Variables	107
APPENDIX C – TRANSLATION TABLES	109
APPENDIX D — INSTALLING A CLIENT UNDER UNIX: AN ALTERNATE PROCEDURE	111
Installation Procedure	112
Step 1 Log In at the UNIX System Prompt	112
Step 2 Generate the Environment File	112
Step 3 Select the Database Drivers	114
Step 4 Relink Natural on UNIX Client Machines	116

APPENDIX E — CLIENT AND SERVER TRACES	117
Client Tracing	117
UNIX Client	117
Windows NT client	117
AXP OpenVMS Client	117
OS/390 Client	117
Server Tracing	118
UNIX Server	118
Windows NT Server	118
AXP OpenVMS Server	118
OS/390 Server	118
INDEX	119

PREFACE

This manual is intended for users running Entire Access for TCP/IP on the UNIX server platform. For information about using this product on other server platforms, refer to the following Software AG documentation:

- *Entire Access for TCP/IP Manual for Windows 2000 and Windows NT*
- *Entire Access for TCP/IP Manual for OS/390*
- *Entire Access for TCP/IP Manual for AXP OpenVMS*

What is Entire Access for TCP/IP?

Entire Access for TCP/IP allows client applications running on UNIX, Windows 98, Windows 2000, Windows NT, and OpenVMS client platforms to access data sources on UNIX, Windows 2000 Server, Windows NT, AXP OpenVMS, and OS390 server platforms.

Entire Access for TCP/IP provides concurrent access to as many as seven different data sources for client applications that use ANSI-standard SQL to issue database requests. It provides core support for Natural client applications and, with the addition of specific drivers, supports Bolero applications and ODBC-compliant applications such as Excel and Microsoft query:

Application Type	Client Platform	Server Data Source	Requirements
Natural	UNIX, Windows 98, Windows 2000, Windows NT, AXP OpenVMS	All supported RDBMS	Entire Access for TCP/IP
Bolero and Java	UNIX, Windows 98, Windows 2000, Windows NT AXP OpenVMS OS390	All supported RDBMS	Entire Access for TCP/IP and Entire Access for JDBC
ODBC-compliant	Windows 98, Windows 2000, Windows NT	ADABAS C via ADABAS SQL Server	Entire Access for TCP/IP and Entire Access for ODBC

Entire Access for TCP/IP Manual for UNIX

Entire Access for TCP/IP represents a client-server solution for Software AG database systems and for third-party products. The following table lists the data sources for each supported server platform:

Server Data Source	UNIX	OS390	Windows NT Windows 2000	AXP OpenVMS
ADABAS C via ADABAS SQL Server (ESQ)	x	x	x	
ADABAS D	x		x	
DB2/x and mainframe DB2	x	x	x	
INFORMIX	x		x	
OpenIngres	x		x	
Microsoft SQL Server			x	
ORACLE	x		x	x
SYBASE DBLIB	x		x	
SYBASE CTLIB	x		x	
RMS (Record Management Services)				x

Entire Access for TCP/IP complies with Microsoft's Open Database Connectivity (ODBC) standard. Depending on the products installed at your site, it may be possible to access ODBC-compliant data sources on the same machine and on remote UNIX, Windows 2000, or Windows NT machines. Whether such access is possible depends on the RDBMS. As a minimum, data sources to be accessed using the Entire Access for TCP/IP ODBC driver must comply with the ODBC level 1 API and must accept the SQL core grammar.

Using this Manual

This manual describes the installation and use of Entire Access for TCP/IP on the Windows NT server platform:

Chapter 1 tells you how Entire Access for TCP/IP works.

Chapter 2 explains how to install Entire Access for TCP/IP on a UNIX server.

Chapters 3 through 6 explain how to install the Entire Access for TCP/IP client component on:

- UNIX (chapter 3).
- Windows 98, Windows 2000, or Windows NT (chapter 4).
- AXP OpenVMS (chapter 5).

Chapter 6 describes the process of defining data sources (servers) to Natural on the client machines.

Chapter 7 provides information about generating and loading Natural DDMs used to define the structure of relational tables and views to Natural.

Chapter 8 considers issues associated with supplying user IDs and passwords for RDBMS that require them.

Appendix A describes the special uses of Natural with Entire Access for TCP/IP.

Appendix B describes the use of Intersolv ODBC.

Appendix C provides translation tables.

Appendix D provides an alternate method for installing the Entire Access client component on UNIX in case the procedure presented in chapter 3 is inappropriate for your site.

Appendix E provides client and server traces.

HOW ENTIRE ACCESS FOR TCP/IP WORKS

Entire Access for TCP/IP supports local and remote databases. It consists of an application program interface (API) and each supported RDBMS. Multiple heterogeneous RDBMS can be accessed concurrently from within the same client application.

The API provides a common SQL interface; it receives ANSI-standard SQL requests from the client application and routes them to the driver for the target data source. Entire Access for TCP/IP forms the backbone for RDBMS access.

Natural applications use the Entire Access for TCP/IP backbone directly. The API supports the use of Natural DML and SQL statements in the same program.

Bolero and Java applications use the client driver Entire Access for JDBC in addition to the Entire Access for TCP/IP backbone.

ODBC-compliant applications such as Excel, MS Query, Crystal Reports use the client driver Entire Access for ODBC in addition to the Entire Access for TCP/IP backbone.

The database driver

- converts data to ensure consistent data types;
- emulates RDBMS-specific functions; and
- automatically coordinates user requests with replies from the RDBMS.

The database drivers are reentrant; thus, after an application establishes a connection to a data source, other applications can access the data source during the same Natural session without having to reestablish the connection.

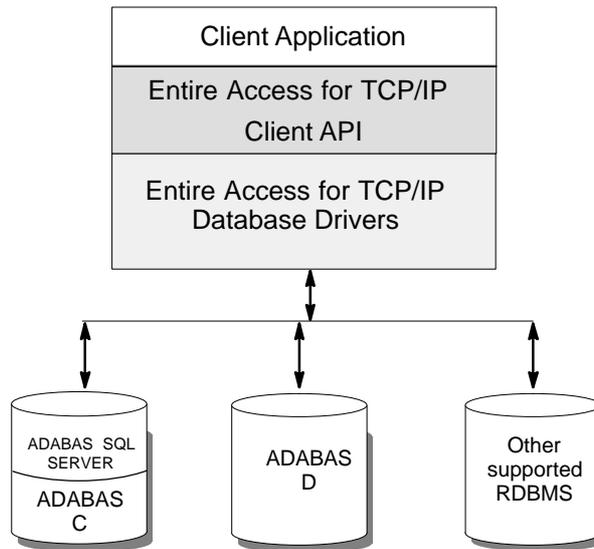
Accessing Data Sources from Clients

The same procedure is used to access data sources from all client platforms: UNIX, Windows 98, Windows 2000, Windows NT, and AXP OpenVMS:

1. On the server machine, start the database and then start Entire Access for TCP/IP.
2. On the client machine, set the connect string and then start the client application.

Local Data Access

With local access, the application client and Entire Access for TCP/IP reside on the same platform as the database server; the Entire Access for TCP/IP driver communicates directly with the data source:



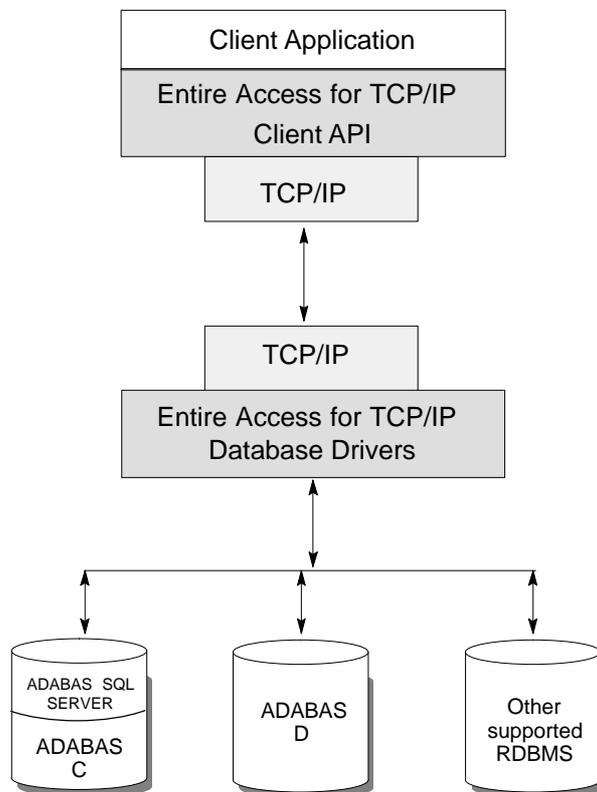
Note:

Local access to data sources on the OS/390 platform is supported only for Bolero application clients using Entire Access for JDBC for OS/390. Natural application clients are not supported under OS/390.

Remote Data Access Using TCP/IP

With remote access, application clients communicate with Entire Access for TCP/IP on the server site using the TCP/IP communications protocol. Entire Access for TCP/IP and TCP/IP must be installed on each client and server machine. The API on the client machine uses TCP/IP to route requests to the database driver on the server machine.

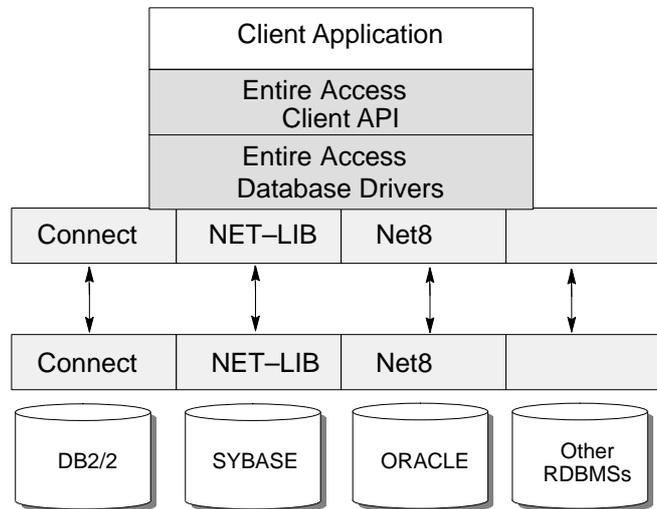
The following diagram shows remote access using TCP/IP:



Remote Data Access Using Third-Party Network Products

It may be possible to use network products supplied by the vendors of third-party RDBMS. The network product for **each** server RDBMS must be installed on the client and server machines. Entire Access communicates with these network products in the same way it communicates with a local data source. The network product is responsible for transmitting and coordinating the requests and replies between the client and the data source.

There are many possible configurations of RDBMS and third-party network products. Be sure to evaluate a particular configuration to determine whether it will work with Entire Access; if necessary, contact your Software AG representative for assistance. The following diagram illustrates possible uses of third-party network products with Entire Access:



Entire Access treats databases accessed through the third-party communications product as local databases. Due to the local illusion implemented by various third-party networking products, Entire Access is completely unaware of the remoteness of the database.

Warning:

When using third-party networking products, be aware that it is completely the responsibility of the third-party vendor to implement the local illusion. For more information, contact the respective third-party vendor.

INSTALLING A SERVER UNDER UNIX

This chapter tells you how to install Entire Access for TCP/IP on a UNIX server. It includes a discussion of

- hardware and software prerequisites,
- installation procedures, and
- the Entire Access for TCP/IP directory structure.

Client platforms are UNIX, Windows 98, Windows 2000, Windows NT, and OpenVMS. For information about the installation procedure, hardware and operating system requirements, and additional software requirements, see the chapter for that platform.

Special requirements for INTERSOLV ODBC clients and servers are described in Appendix B, page 101, using HP-UX as an example.

Before You Install

Client and Server Version Levels

Entire Access for TCP/IP client and server software are normally at the same version level. Whenever a server is used, the client uses the client software that was provided with the server.

However, Entire Access for TCP/IP allows client and server version mixing for up to two versions of the product.

Example

Note:

In the following tables, product codes are used to represent the product names:

- *OSX is the product code for Entire Access for TCP/IP.*
- *OSY is the product code for Entire Access for JDBC.*
- *OSZ is the product code for Entire Access for ODBC.*

The following combination of client and server versions is preferred:

Client	Server
OSX 4.2.1, OSY 4.2.1, OSZ 4.2.1	OSX 4.2.1 and any supported RDBMS

However, the following combinations are also acceptable:

Client	Server
OSX 4.1.1, OSZ 4.1.1	OSX 4.2.1 and any supported RDBMS
OSX 4.2.1, OSY 4.2.1, OSZ 4.2.1	OSX 4.1.1 and any supported RDBMS

Note:

If a problem log for OSX411 is resolved in OSX421, the solution is to upgrade to the higher version.

Hardware and Operating System Requirements

The following table shows the minimum operating system versions and hardware requirements for selected UNIX platforms supported by Entire Access for TCP/IP. For a complete list of supported platforms, contact your Software AG technical support representative.

Platform / Operating System	Hardware Requirements
AIX 4.3.3	IBM Series RS/6000
Digital UNIX 4.0D (Tru64), 5.0	Compaq (Digital ALPHA)
HP-UX 11.0 32-bit or 64-bit	HP9000 series S700 or S800
SCO UnixWare 7.01	Intel
SNI Reliant UNIX 5.44	SNI RMx00/R10000
Solaris 2.6	Sun SPARC, Sun Ultra SPARC
Solaris 7 32-bit	Sun Ultra SPARC
Solaris 8 64-bit	Sun Ultra SPARC

Note:

HP-UX 11.0 64-bit and Solaris 8 64-bit only support 64-bit databases and 64-bit Entire Access products.

Other Software Requirements

TCP/IP is required on both client and server machines for remote access.

For information about additional client machine requirements, see the chapter for the client platform.

For a brief discussion of the use of third-party network products, see the section **Remote Data Access Using Third-Party Network Products** on page 8.

UNIX Compiler Support

Entire Access for TCP/IP supports the vendor compiler, including the Linker or Loader, for each supported UNIX platform. For example:

- On the SUN platform Entire Access for TCP/IP supports SUN Workshop Pro, the supported SUN compiler; it does not support the GNU compiler.
- On the HP 11.0 32-bit platform, Entire Access for TCP/IP supports GNU, the officially supported compiler for the platform; on other platforms, GNU and UCB are not officially supported compilers.

Database Systems and Versions Overview

Support for specific databases depends on the UNIX platform(s). Except where noted, access can be either local or remote.

The following database servers are supported:

ADABAS SQL Server 1.4.3
ADABAS D 10.0 and 11.0
DB2 5.0*, 6.1 and 7.1
INFORMIX-OnLine 7.3 and 9.x
OpenIngres 2.0
ORACLE 7.3, 7.3.4, 8.0.5 and above, 8.1.5 and above
SYBASE 11.5+ CTLIB and DBLIB
SYBASE 12 CTLIB
ODBC-compliant (local access)

*DB2 version 6.1 has limited support; Fix Pak 2 is currently required. Problems involving DB2 v5.x may require an upgrade to the highest DB2 version level. DB2 v5.x has limited support.

INFORMIX

INFORMIX users must perform the following steps:

- Use INFORMIX-ESQL/C (embedded SQL for C), which is also known as INFORMIX Client SDK. It provides the libraries required to build the server daemon and/or the Natural that includes the driver. INFORMIX-SE is not supported.
- Turn on (buffered or unbuffered) logging for each database to be accessed with Entire Access for TCP/IP. Refer to the INFORMIX-OnLine documentation for information about enabling the logging facility.

ORACLE

Several ORACLE 7 patch levels have library structures that may result in unresolved symbols when linking Natural and/or the server daemon for ORACLE 7 access. If this occurs, contact your Oracle Corporation support representative to obtain the correct sequence of the ORACLE 7 libraries.

Database System Preparation

Before you begin to install Entire Access for TCP/IP, perform the following steps:

- Install your DBMS software.
- Set the environment variables for the relevant RDBMS. See the table below for a list of the environment variables required for each RDBMS.
- DB2 users: Create the links for the DB2 libraries with the command **db2ln**. Refer to the DATABASE 2 UNIX installation guides for more information.

Environment Variables

RDBMS users must set the environment variables according to the shell being used, as shown in the following table. Except where noted, these variables are required at build time and/or run time.

RDBMS	Environment Variable	Build/Run Time
ADABAS SQL Server	ESQPARMS	Run
	ESQDIR	Build
	ESQVERS	Build
ADABAS D v10	DBROOT	Build/Run
	DBNAME (optional) ¹	Run
	PATH=\$PATH:DBROOT/bin	Run
	ESDDIR	Build/Run
	ESDVERS	Build/Run
ADABAS D v11	DBROOT	Build/Run
	DBNAME (optional) ¹	Run
	PATH=\$PATH:DBROOT/bin	Run
DB2 ²	DB2_HOME	Build
	DB2INSTANCE	Run
	PATH=\$PATH:\$DB2_HOME/sqlib/adm: \$DB2_HOME/sqlib/bin: \$DB2_HOME/sqlib/misc	Run
INFORMIX 7.3 and INFORMIX 9.x	INFORMIXDIR	Build
	PATH=\$PATH:\$INFORMIXDIR/bin	Run
	SQLEXEC=\$INFORMIXDIR/lib/sqlrm	Run
	INFORMIXSERVER=host-name	Run
	INFORMIXSHMBASE=shmbase / 1024 ³	Run
OpenIngres	II_SYSTEM	Build
ORACLE	ORACLE_HOME	Build/Run
	ORACLE_SID	Run
	TWO_TASK (optional) ⁴	Run

RDBMS	Environment Variable	Build/Run Time
SYBASE DBLIB and SYBASE CTLIB	SYBASE DSQUERY	Build Run

- 1 If DBNAME is set, it will override the database specified in the connect string (that is, Entire Access for TCP/IP will connect to the database specified in DBNAME and not to the database specified in the connect string); for further information on connect strings, see the section **Define the Data Source(s)** on page 50.
- 2 DB2 version 6.1 may establish two different areas for the installation: 1. Installation home, as defined at install time (i.e., /RDBMS/db2); 2. Actual home, as typically defined by /opt/IBMdb2/v6.1. The DB2_HOME should be set to the **installation home** directory.
- 3 When you use Natural to access a local INFORMIX 7.3 or 9.x database, you may receive INFORMIX error -25588 if the default Shared Memory Address from INFORMIX conflicts with the Natural buffer pool. To resolve this error, set the environment variable INFORMIXSHMBASE to a value that depends on the INFORMIX configuration parameter SHMBASE. For example, if SHMBASE is set to 536870912 (0x20000000), set INFORMIXSHMBASE to 524288 (0x20000000 / 0x400) as follows:

```
$ INFORMIXSHMBASE=524288
$ export INFORMIXSHMBASE
```

When you use Natural to access a local INFORMIX 7.3 or 9.x database on AIX 4.3.2, you may receive the following undocumented INFORMIX error:

```
-1829
Unable to load locale categories.
```

Setting any or all of the following variables is likely to resolve this error:

```
$ CLIENT_LOCALE=en_US
$ DB_LOCALE=en_US
$ SERVER_LOCALE=en_US
$ export CLIENT_LOCALE DB_LOCALE SERVER_LOCALE
```

For more information, refer to the manual *The Informix Guide to SQL: Reference* or *The Informix GLS Programmer's Manual*.

- 4 TWO_TASK must be set to “P:” if the database instance to be accessed is on the local machine. If TWO_TASK is set, it will override the database instance specified in the ORACLE_SID environment variable.

Installation Procedure

Use the following steps to install Entire Access for TCP/IP. Each step is valid for any supported UNIX platform. Actions specific to a particular product, such as DB2 or ADABAS SQL Server, are identified.

Step 1 Log In at the UNIX System Prompt

Log in as “sag”; do **not** log in as “root”.

Step 2 Generate the Environment File

The environment file “sagenv.new” must be modified to include the required environment variables before using Entire Access for TCP/IP.

Note:

If you have an existing “sagenv.old” environment file, be sure to rename it; otherwise, it will be overwritten later in this step when the “sagenv.new” file is automatically generated and the existing “sagenv.new” file is renamed to “sagenv.old”.

Execute the interactive SAGINST script to generate the “sagenv.new” file.

1. To start the script, enter the following commands:

```
cd $SAG  
./SAGINST
```

The script ensures that the SAG environment variable has been established; it then displays a list of all products in the supplied \$SAG directory.

2. Select each required product from the list by entering the corresponding number (from the left-hand column) after the prompt. Use spaces to separate the numbers. Be sure to select the correct version of Entire Access for TCP/IP.

Do not select more than one version of a product. In the following example, Natural version 4.1.2.6 and Entire Access for TCP/IP version 4.2.1 are selected:

```
INSTALL: ENVIRONMENT
```

```
Please choose products for which you want to
generate the environment file sagenv.new
```

```
1      ada/v31124
2      nat/v4126
3      osx/v421
```

```
PLEASE SELECT ITEMS : 2 3
```

3. Press ENTER to generate the “sagenv.new” file.
The generated “sagenv.new” file includes all environment variables required to use the selected products. If “sagenv.new” already exists, it is automatically renamed to “sagenv.old” and the previous “sagenv.old” is overwritten.
4. If you are performing an update installation (that is, you selected only the new products to be added to your existing “sagenv” file), use the concatenate command to append the “sagenv.new” to your existing “sagenv” file.
5. (Optional) Rename “sagenv.new” to another file name; the following steps assume that the environment file was renamed to “sagenv”.
6. To establish the modified environment variables, invoke the “sagenv” file with the following command:
. ./sagenv
7. To automatically establish this environment each time you log in, add the following command to your “.profile” file:

```
. <SAG-home-directory>/sagenv
```

The file will be executed automatically each time you log in.

Step 3 Select the Database Drivers (Natural for UNIX Only)

Use the interactive “osxlibs.sh” script to select the database drivers(s) to be used by Entire Access for TCP/IP.

1. To change your directory, enter the following command:

```
$ cd $OSXDIR/$OSXVERS/bin
```

2. To start the script, enter the following command:

```
$ osxlibs.sh
```

A list of database drivers appears.

3. Select each desired driver by entering the corresponding number (from the left-hand column) after the prompt and pressing ENTER.

To deselect a database driver, reenter the number for that driver at the prompt and press ENTER.

Each selected entry is indicated by an asterisk (*) to the left of the number column. In the following example, the selected entry (*) is local ADABAS SQL Server 1.4.3.

```
Entire Access for TCP/IP (HP-UX 11 32-bit)
=====

 1 - remote Entire Access Net          12 - local INGRES 2.0
* 2 - local ADABAS SQL Server 1.4.3    13 - local ORACLE 7.3
 3 - local ADABAS D 10.0               14 - local ORACLE 7.3.4
 4 - local ADABAS D 11.0               15 - local ORACLE 8.0.5
 5 - local ADABAS D ODBC               16 - local ORACLE 8.0.6
 6 - local DB2 v5 or v6.1              17 - local ORACLE 8.1.5
 7 - local DB2 v7.1                    18 - local ORACLE 8.1.6
 8 - local DB2 CLI V7                  19 - local SYBASE 11 DBLIB
 9 - local INFORMIX 7.3 (online)       20 - local SYBASE 11 CTLIB
10 - local INFORMIX IDS 9.1.3          21 - local SYBASE 12 CTLIB
11 - local INFORMIX IDS 9.2           22 - local INTERSOLV ODBC

g - Generate 'osxlibs.lst'
q - Exit

please select an entry:
g
```

4. After making your selections, enter “g” and press ENTER.

The following is an example of the confirmation screen that appears. It lists the values of the environment variables found for the drivers you selected.

```

Entire Access for TCP/IP
=====

You have selected the following parts to build Natural

- local adabas sql server version

$OSXDIR      = /usr/SAG/osx
$OSXVERS     = v421
$NATDIR      = /usr/SAG/nat
$NATVERS     = v4126
$ESQDIR      = /usr/SAG/esq
$ESQVERS     = v143

```

Press <enter> to see the file ' /usr/SAG/osx/v421/osxlibs.lst

5. When using shareable libraries, you may need to set an additional variable:

When using shareable libraries with:	Set the variable:
INFORMIX version 7.3 and 9.x under AIX version 4.3.2	LIBPATH
Solaris	LD_LIBRARY_PATH
HP-UX 32-bit	SHLIB_PATH
HP-UX 64-bit when using 32-bit utilities (SQL*Plus)	SHLIB_PATH
HP-UX 64-bit only	LD_LIBRARY_PATH
Tru64 (DEC UNIX)	LD_LIBRARY_PATH
SCO UNIXWARE	LD_LIBRARY_PATH
other UNIX	Consult the appropriate programmer's manual.

For example:

Set the LIBPATH variable as follows when using shareable libraries with INFORMIX version 7.3 and 9.x under AIX version 4.3.2:

```
$LIBPATH=/RDBMS/informix/lib/esql:/RDBMS/informix/lib:/usr/lib
$export LIBPATH
```

6. Verify that the environment variables are correct; then press ENTER to generate the “osxlibs.lst” file. The screen displays the contents of this file as it is being generated.

The “osxlibs.lst” file contains a list of all database libraries to be linked to the Natural prelinked object “natraw.o”. The “make” file uses the “osxlibs.lst” file to build the new Natural environment in step 4.

Step 4 Link the Server Daemons on Server Machines

1. Access the “bin” directory of the “osx” library by entering the following command:

```
$ cd $OSXDIR/$OSXVERS/bin
```

2. Enter the “make” command, specifying a valid “daemon-type” as shown in the table:

```
$ make [daemon-type]
```

Specify . . .	For . . .
esq143	ADABAS SQL Server 1.4.3
adabasd10	ADABAS D 10.0
adabasd11	ADABAS D 11.0
db261	DB2 v6.1 *
db271	DB2 v7.1
informix73	INFORMIX-OnLine version 7.3
informix913	INFORMIX IDS 9.1.3
informix92	INFORMIX IDS 9.2
ingres2	OpenIngres 2.0
oracle73	ORACLE version 7.3
oracle734	ORACLE version 7.3.4
oracle805	ORACLE version 8.0.5
oracle806	ORACLE version 8.0.6

Specify . . .	For . . .
oracle815	ORACLE version 8.1.5
oracle816	ORACLE version 8.1.6
sybase	SYBASE 11.5 DBLIB
sybase11	SYBASE 11.5 CTLIB
sybase12	SYBASE 12 CTLIB

- * DB v5.x users might use the DB2 v6.1 entry. However, in the event of a bug, an upgrade may be required to the highest DB2 version.

The following “daemon-types” are ODBC-specific:

Specify . . .	For . . .
adabasodbc	ADABAS D 11.0 ODBC (only)
db2cli	DB2 v7.1 CLI
intersolv	INTERSOLV ODBC

Note:

If necessary, modify the paths in “Makefile” according to your database system requirements. However, it is best to consult with your Software AG technical support representative to ensure that a change can be supported.

Step 5 Bind the Entire Access for TCP/IP Package

This step is required for DB2 users only.

Bind the Entire Access for TCP/IP package to each DB2 database to be accessed. The “db2bind.bnd” file contains the package. The “db2bind.sh” shell script executes the bind process; the default setting for the bind authorization is “public”.

Only an authorized user may issue the bind command.

1. Ensure that the DB2 database to be accessed by Entire Access for TCP/IP has been started.
2. Change to the “bin” directory where Entire Access for TCP/IP is installed by entering the following command at the system prompt:

```
$ cd $OSXDIR/$OSXVERS/bin
```

3. If necessary, edit “db2bind.sh” and change the bind authorization.
4. Run the following shell script for each DB2 database to be used by Entire Access for TCP/IP:

```
db2bind.sh db-name
```

—where “db-name” is the name of the DB2 database.

Step 6 Establish ADABAS SQL Server Run-time Parameters

This step is required for ADABAS SQL Server users only.

Refer to the ADABAS SQL Server version 1.4.3 documentation for more information.

The environment variable ESQPARMS specifies an ADABAS SQL Server run-time parameter file. Copy the file from \$ESQDIR/\$ESQVERS/files/esqrun.par to \$ESQPARMS=\$OSXDIR/\$OSXVERS/esqrun.par. **Do not modify the original ESQ file.**

The following example of this run-time parameter file illustrates the minimum requirements for Entire Access for TCP/IP:

```
GLOBAL
BEGIN
    CATALOG ( DBID = DBnr, FNR = Fnr )
    ERROR    ( DBID = DBnr, FNR = Fnr )
END

RUNTIME
BEGIN
    SERVER SESSION TIMEOUT = 120
    MAX DYNAMIC MPS       = 60
    MAX CURSORS            = 60
END
```

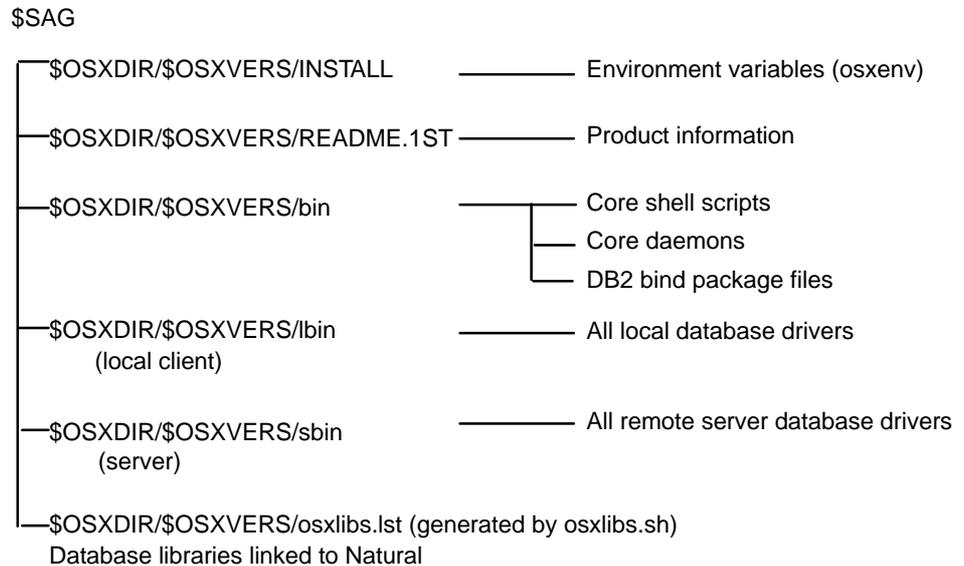
—where

CATALOG	Defines the ADABAS database and the ADABAS file in which the SQL Directory is located. With ADABAS SQL Server version 1.4, this is a CATALOG. With version 1.3, this is a DIRECTORY. Note that version 1.3 is no longer supported.
ERROR	Defines the ADABAS database and the ADABAS file in which the error messages are located.
MAX CURSORS	Specifies the maximum number of cursors expected to be opened. The default value is 15; the recommended value is 60.

MAX DYNAMIC MPS	Specifies the maximum number of dynamic metaprograms which are expected to be executed. Dynamic metaprograms are only in existence when the application is running. The default value is 15; the recommended value is 60.
SERVER SESSION TIMEOUT	Defines the server's timeout for a particular client. The recommended value is 120.

The Entire Access Directory Structure Under UNIX

The following is the Entire Access directory structure under UNIX:



INSTALLING A CLIENT UNDER UNIX

This chapter tells you how to install the Entire Access for TCP/IP client component under UNIX. It includes a discussion of hardware and software prerequisites and installation procedures.

An alternate installation procedure is provided, beginning on page 111. Refer to this information if the procedure outlined in this chapter is inappropriate for your site.

For more information about the various utilities and server capabilities for UNIX, refer to the *Entire Access for TCP/IP Manual for UNIX*.

Before You Install

Client and Server Version Levels

Beginning with version 4.1, the current version of Entire Access for TCP/IP can normally communicate with the prior version, as follows:

- Version 4.1 on the client can communicate with version 4.2 on the server.
- Version 4.2 on the client can communicate with version 4.1 on the server.

There are cases, however, where a complete upgrade to the current version is required on both the client and the server.

UNIX Compiler Support

Entire Access for TCP/IP supports the vendor compiler, including the Linker or Loader, for each supported UNIX platform. For example:

- On the SUN platform Entire Access for TCP/IP supports SUN Workshop Pro, the supported SUN compiler; it does not support the GNU compiler.
- On the HP 11.0 32-bit platform, Entire Access for TCP/IP supports GNU, the officially supported compiler for the platform; on other platforms, GNU and UCB are not officially supported compilers.

Hardware and Operating System Requirements

The following table shows the minimum operating system versions and hardware requirements for selected UNIX platforms supported by Entire Access for TCP/IP. For a complete list of supported platforms, contact your Software AG representative.

Platform / Operating System	Hardware Requirements
AIX 4.3.3	IBM Series RS/6000
Digital UNIX 4.0D (Tru64), 5.0	Compaq (Digital ALPHA)
HP-UX 11.0 32-bit or 64-bit	HP9000 series S700 or S800
SCO UnixWare 7.01	Intel
SNI Reliant UNIX 5.44	SNI RMx00/R10000
Solaris 2.6	Sun SPARC, Sun Ultra SPARC
Solaris 7 32-bit	Sun Ultra SPARC
Solaris 8 64-bit	Sun Ultra SPARC

Note:

HP-UX 11.0 64-bit and Solaris 8 64-bit only support 64-bit databases and 64-bit Entire Access products.

Other Software Requirements

Entire Access for TCP/IP version 4.2 under UNIX requires TCP/IP on both the client and the server for remote server support. In addition:

- Natural client applications require Natural for UNIX version 4.1 or above.
- Bolero client applications require Bolero for UNIX and Entire Access for JDBC 4.1.1 or above.

For a brief discussion of the use of third-party network products, see the section **Remote Data Access Using Third-Party Network Products** on page 8.

Installation Procedure

Use the following steps to install the Entire Access for TCP/IP client component on any supported UNIX platform. Actions specific to a particular product, such as DB2 or ADABAS SQL Server, are identified.

Step 1 Log In at the UNIX System Prompt

Log in as “sag”; do **not** log in as “root”.

Step 2 Generate the Environment File

The environment file “sagenv.new” must be modified to include the environment variables required before using Entire Access for TCP/IP.

Note:

If you have an existing “sagenv.old” environment file, be sure to rename it; otherwise, it will be overwritten later in this step when the “sagenv.new” file is automatically generated and the existing “sagenv.new” file is renamed to “sagenv.old”.

Execute the interactive SAGINST script to generate the “sagenv.new” file.

1. To start the script, enter the following commands:

```
$ cd $SAG  
$/SAGINST
```

The script ensures that the SAG environment variable has been established; it then displays a list of all products in the supplied \$SAG directory.

2. Select each required product from the list by entering the corresponding number (from the left-hand column) after the prompt. Use spaces to separate the numbers.

Be sure to select the correct version of Entire Access for TCP/IP.

Do not select more than one version of a product. In the following example, Natural version 4.1.2 and Entire Access for TCP/IP version 4.2.1 are selected:

INSTALL: ENVIRONMENT

Please choose products for which you want to generate the environment file `sagenv.new`

```
1      ada/v31119
2      nat/v4126
3      osx/v421
```

PLEASE SELECT ITEMS : 2 3

3. Press ENTER to generate the “`sagenv.new`” file.
The generated “`sagenv.new`” file includes all environment variables required to use the selected products. If “`sagenv.new`” already exists, it is automatically renamed to “`sagenv.old`” and the previous “`sagenv.old`” is overwritten.
4. If you are performing an update installation (that is, you selected only the new products to be added to your existing “`sagenv`” file), use the concatenate command to append the “`sagenv.new`” to your existing “`sagenv`” file.
5. (Optional) Rename “`sagenv.new`” to another file name; the following steps assume that the environment file was renamed to “`sagenv`”.
6. To establish the modified environment variables, invoke the “`sagenv`” file with the following command:
`$. /sagenv`
7. To automatically establish this environment each time you log in, add the following command to your “`.profile`” file:
`$. <SAG-home-directory>/sagenv`
The file will be executed automatically each time you log in.

Step 3 Select the Database Drivers (Natural for UNIX Only)

Use the interactive “osxlibs.sh” script to select the database drivers(s) to be used by Entire Access for TCP/IP.

1. To change your directory, enter the following command:

```
$ cd $OSXDIR/$OSXVERS/bin
```

2. To start the script, enter the following command:

```
$ osxlibs.sh
```

A list of database drivers appears.

3. Select each desired driver by entering the corresponding number (from the left-hand column) after the prompt and pressing ENTER.

To deselect a database driver, reenter the number for that driver at the prompt and press ENTER.

Each selected entry is indicated by an asterisk (*) to the left of the number column. In the following example, the selected entry (*) is local ADABAS SQL Server 1.4.3:

```
Entire Access for TCP/IP (HP-UX 11 32-bit)
=====

  1 - remote Entire Access Net          12 - local INGRES 2.0
*  2 - local ADABAS SQL Server 1.4.3    13 - local ORACLE 7.3
  3 - local ADABAS D 10.0               14 - local ORACLE 7.3.4
  4 - local ADABAS D 11.0               15 - local ORACLE 8.0.5
  5 - local ADABAS D ODBC                16 - local ORACLE 8.0.6
  6 - local DB2 v5 or v6.1              17 - local ORACLE 8.1.5
  7 - local DB2 v7.1                    18 - local ORACLE 8.1.6
  8 - local DB2 CLI V7                   19 - local SYBASE 11 DBLIB
  9 - local INFORMIX 7.3 (online)        20 - local SYBASE 11 CTLIB
 10 - local INFORMIX IDS 9.1.3           21 - local SYBASE 12 CTLIB
 11 - local INFORMIX IDS 9.2            22 - local INTERSOLV ODBC

g - Generate 'osxlibs.lst'
q - Exit

please select an entry:
g
```

4. After making your selections, enter “g” and press ENTER.

The following is an example of the confirmation screen that appears. It lists the values of the environment variables found for the drivers you selected.

```
Entire Access for TCP/IP
=====
```

You have selected the following parts to build Natural

- local adabas sql server version

```
$OSXDIR    = /usr/SAG/osx
$OSXVERS   = v421
$NATDIR    = /usr/SAG/nat
$NATVERS   = v4126
$ESQDIR    = /usr/SAG/esq
$ESQVERS   = v143
```

Press <enter> to see the file ' /usr/SAG/osx/v411/osxlibs.lst

5. When using shareable libraries, you may need to set an additional variable:

When using shareable libraries with:	Set the variable:
INFORMIX version 7.3 and 9.x under AIX version 4.3.2	LIBPATH
Solaris	LD_LIBRARY_PATH
HP-UX 32-bit	SHLIB_PATH
HP-UX 64-bit when using 32-bit utilities (SQL*Plus)	SHLIB_PATH
HP-UX 64-bit only	LD_LIBRARY_PATH
Tru64 (DEC UNIX)	LD_LIBRARY_PATH
SCO UNIXWARE	LD_LIBRARY_PATH
other UNIX	Consult the appropriate programmer's manual.

For example:

Set the LIBPATH variable as follows when using shareable libraries with INFORMIX version 7.3 and 9.x under AIX version 4.3.2:

```
$LIBPATH=/RDBMS/informix/lib/esql:/RDBMS/informix/lib:/usr/lib
$export LIBPATH
```

6. Verify that the environment variables are correct; then press ENTER to generate the “osxlibs.lst” file. The screen displays the contents of these files as they are being generated.
The “osxlibs.lst” file contains a list of all database libraries to be linked to the Natural prelinked object “natraw.o”. The “make” file uses the “osxlibs.lst” file to build the new Natural environment in step 6.

Step 4 Relink Natural on UNIX Client Machines

Regenerate your Natural nucleus with the selected Entire Access for TCP/IP database drivers. This step is required for Natural 4.1.

1. Change to the Natural build directory by entering the following command:

```
cd $NATDIR/$NATVERS/bin/build
```

2. Enter a command to build a new Natural nucleus that includes support for the database drivers selected in step 5.

If you do not require Natural Security, Natural RPC, ADABAS, or ADABAS SQL Server, enter the command as follows:

```
make natural osx=yes
```

If you do require Natural Security, Natural RPC, ADABAS, or ADABAS SQL Server, enter the command with the “ada=[*adabas-library*]” specification, as follows:

```
make natural osx=yes ada= [adabas-library]
```

where adabas-library has one of the following values:

dyn	link in ADABAS shared libraries
stat	link in ADABAS static libraries
cscidyn	link in ADABAS and CSCI shared libraries
cscistat	link in ADABAS and CSCI static libraries

Whether you should link ADABAS as static or dynamic depends on the version of ADABAS you are using. Refer to your ADABAS installation instructions for more information.

Notes:

1. *By default, ADABAS is not included when Natural is relinked. If you fail to specify a value for the “ada=” parameter, ADABAS will not be linked into Natural and Natural Security will not function.*
2. *Users who require Natural RPC or ADABAS SQL Server must specify the ada= parameter for CSCI components (cscistat or cscidyn) to include a required CSCI stub module.*
3. To copy this new Natural file into the “bin” directory, enter the following command:
make install

INSTALLING A CLIENT UNDER WINDOWS 98, 2000, OR NT

This chapter tells you how to install the Entire Access for TCP/IP client component under Windows 98, Windows 2000, or Windows NT. It provides a discussion of

- hardware and software prerequisites; and
- installation procedures.

Before You Install

Client and Server Version Levels

Beginning with version 4.1, the current version of Entire Access for TCP/IP can normally communicate with the prior version, as follows:

- Version 4.2 on the server can communicate with version 4.1 on the client.
- Version 4.2 on the client can communicate with version 4.1 on the server.

There are cases, however, where a complete upgrade to the current version is required on both the client and the server.

Client Hardware and Operating System Requirements

The following are the minimum hardware requirements for using Entire Access for TCP/IP:

- an Intel 80586 CPU with a speed of 300 MHz
- 64 MB of RAM
- about 4 MB of available hard-disk space (if all options are installed)
- operating system:
 - Windows 98;
 - Windows 2000; or
 - Windows NT version 4.0 with Service Pack 6

Other Software Requirements for Clients

The following products and versions are required in order to use Entire Access for TCP/IP:

- For Natural clients, Natural version 4.1 is required.
- For remote access, TCP/IP is required on the client machine and on the server machine.

In some cases, you can use third-party network products in conjunction with the ODBC driver. See the section **Remote Data Access Using Third-Party Network Products** on page 8.

Installation Procedure

Installation CDs

The Entire Access for TCP/IP installation package contains two CDs:

- The server CD contains the Windows 2000 or Windows NT server component.
- The client CD contains the Windows 98, Windows 2000, and Windows NT client components only.

Installation Steps

To install the Entire Access for TCP/IP client component under Windows 98, Windows 2000, or Windows NT, copy the client API and driver files on the client CD to the BIN directory where Natural has been installed, as follows:

1. Insert the client CD.
2. Change to the Windows 98, Windows 2000, or Windows NT directory on the CD.
3. Copy the following files from the client CD to the BIN directory where Natural is installed:

VTXAPI32.DLL
VTX3.DLL
VTX11.DLL

Note:

This step is unnecessary if a complete server installation has been performed from the Entire Access for TCP/IP server CD and Natural version 4.1 or above is used.

INSTALLING A CLIENT UNDER AXP OPENVMS

This chapter tells you how to install the Entire Access for TCP/IP client component under AXP OpenVMS 7.1 or above.

The method used for installation is a combination of manual and automated functions that are executed using command files.

For more information about the various utilities and server capabilities for AXP OpenVMS, refer to the *Entire Access for TCP/IP Manual for OpenVMS*.

Before You Install

Operating System Requirements

To install Entire Access for TCP/IP, you must be running OpenVMS version 7.1 or above on an Alpha AXP platform.

TCP/IP Connectivity

Prior to installing Entire Access for TCP/IP, TCP/IP (UCX) connectivity must be installed for all configurations. TCP/IP must be configured and tested on the host server and on at least one remote OpenVMS client. TCP/IP connectivity can be tested by issuing a successful VTXPING command from the client using the address of the target host.

Supported Application Environment

Entire Access for TCP/IP under OpenVMS 7.1 or above must be installed in a Natural for OpenVMS version 4.1 or higher application environment. Support for other application environments is being considered for future releases.

Refer to the Software AG Natural documentation for more information.

SAGBASE Requirement

Before Entire Access for TCP/IP can be installed under OpenVMS 7.1 or above, the system must be prepared with SAGBASE, the prerequisite for installing Software AG products in an OpenVMS environment. For more information, refer to the SAGBASE documentation.

The Installation CD

The installation CD contains the Entire Access for TCP/IP save sets, which are extracted using the standard OpenVMS `VMSINSTAL` procedure. Save sets are described in the section **Installation Kit Structure** on page 37.

Installation Prerequisites

Before installing Entire Access for TCP/IP, ensure that the following prerequisites have been met:

Disk Space

To install Entire Access for TCP/IP, approximately 750 blocks of hard disk space are required.

PRCLM Parameter

The parameter `PRCLM` must be set to at least 10 for each account that needs the Entire Access server logical names.

WSDEFAULT Parameter

The parameter `WSDEFAULT` must be set to at least 2048 for each account that needs the Entire Access server logical names.

Process Privileges

The following process privileges are required in order to start a server on Alpha AXP:

`SYSPRV,SYSLCK,CMKRNL,SHARE,WORLD,GROUP,GRPNAM,NETMBX,TMPMBX`

Installation Kit Structure

Software products that layer on AXP OpenVMS are assembled into product kits that are physically distributed as one or more save sets (files created by the OpenVMS Backup utility).

The file name of each save set must be identical to the product name; it must also be assigned a unique file type that reflects the order in which the save sets are installed.

The Entire Access for TCP/IP installation kit consists of the following save sets:

Save Set	Contents
OSX041.A	Installation procedures and files used by VMSINSTAL
OSX041.B	Patch level 0 specific files
OSX041.C	Patch level 0 OSX initialization procedures and files


```
%OSX-I-NOTEXIST, File SAG$ROOT:[OSX]VERSION.DAT does not exist,
                    it will be provided by this installation

%OSX-I-VERPL, Entire Access for TCP/IP(OpenVMS/AXP) V4.1.1 patch-level 0
                    initial installation

%OSX-I-NOEXA, No examples provided by this installation

%OSX-I-SPACEOK, This Entire Access for TCP/IP(OpenVMS/AXP) installation
                    requires 708 blocks

%OSX-I-DIRCREATED, Directory SAG$ROOT:[000000.OSX] created.

%OSX-I-DIRCREATED, Directory SAG$ROOT:[OSX.V411] created.

%OSX-I-DIRCREATED, Directory SAG$ROOT:[OSX.V411.VTX] created.

%OSX-I-DIRCREATED, Directory SAG$ROOT:[OSX.V411.LOG] created.

    This installation kit contains a READ_ME_FIRST file,
    named READ_ME_FIRST.411.
    It will be moved to directory SAG$ROOT:[OSX.V411]
    by this installation procedure.
    Please read this file after VMSINSTAL has finished.

%OSX-I-MOVE, Moving file READ_ME_FIRST.411 to SAG$ROOT:[OSX.V411]

* Print READ_ME_FIRST.411 (queue SYS$PRINT)?      [YES]: no

%VMSINSTAL-I-RESTORE, Restoring product save set B ...

%VMSINSTAL-I-RESTORE, Restoring product save set C ...

%OSX-I-MOVE, Moving files to their target directories ...

%OSX-I-MOVE, Moving login procedure LOGIN.COM to SAG$ROOT:[OSX]

* Move STARTUP_OSX.COM to SYS$STARTUP ?          [YES]:

%OSX-I-MOVE, Moving startup procedure STARTUP_OSX.COM
                    to SAG$ROOT:[OSX]

* Enable STARTUP_OSX.COM using SYSMAN ?          [YES]:

%OSX-I-INSTINFO, Remove file entry STARTUP_OSX.COM in system startup database
on node QAX
```

```

%OSX-I-INSTINFO, Add file entry STARTUP_OSX.COM in system startup database on
node QAX phase END mode DIRECT

%OSX-I-SETPROT, Setting protection on new files ...

%OSX-I-INSTINFO, Modify account DBA with settings required by
                    Entire Access for TCP/IP(OpenVMS/AXP)
%OSX-I-INSTINFO, required privilege TMPMBX
%OSX-I-INSTINFO, required default privilege TMPMBX
%OSX-I-INSTINFO, required privilege NETMBX
%OSX-I-INSTINFO, required default privilege NETMBX
%OSX-I-INSTINFO, required privilege GRPNAM
%OSX-I-INSTINFO, required default privilege GRPNAM
%OSX-I-INSTINFO, required privilege GROUP
%OSX-I-INSTINFO, required default privilege GROUP
%OSX-I-INSTINFO, required privilege WORLD
%OSX-I-INSTINFO, required default privilege WORLD
%OSX-I-INSTINFO, required privilege SHARE
%OSX-I-INSTINFO, required default privilege SHARE
%OSX-I-INSTINFO, required privilege CMKRNL
%OSX-I-INSTINFO, required default privilege CMKRNL
%OSX-I-INSTINFO, required privilege SYSLOCK
%OSX-I-INSTINFO, required default privilege SYSLOCK
%OSX-I-INSTINFO, required privilege SYSPRV
%OSX-I-INSTINFO, required default privilege SYSPRV
%OSX-I-INSTINFO, validating ASTLM >= 256
%OSX-I-INSTINFO, validating BIOLM >= 150
%OSX-I-INSTINFO, validating BYTLM >= 150000
%OSX-I-INSTINFO, validating DIOLM >= 150
%OSX-I-INSTINFO, validating ENQLM >= 2000
%OSX-I-INSTINFO, validating FILLM >= 100
%OSX-I-INSTINFO, validating PGFLQUOTA >= 50000
%OSX-I-INSTINFO, validating TQCNT >= 10
%OSX-I-INSTINFO, validating WSQUOTA >= 25000
%OSX-I-INSTINFO, validating WSEXTENT >= 25000

%OSX-I-EXECUTE, Executing startup procedure
                    SAG$ROOT:[OSX]STARTUP_OSX.COM

%OSX-I-STARTUP, OSX 411 startup finished

%OSX-I-CREATE, Creating patch level file SAG$ROOT:[OSX.V411]OSX_PL.DAT

%OSX-I-CREATE, Creating version file SAG$ROOT:[OSX]VERSION.DAT

```

Starting Verification Command Procedure for
Entire Access for TCP/IP(OpenVMS/AXP) 4.1.1

%OSX-I-VERIFY, IVP completed successfully.

The Entire Access for TCP/IP(OpenVMS/AXP) 4.1.1
initial installation completed successfully

For further installation steps, please refer to
installation notes of Entire Access for TCP/IP(OpenVMS/AXP) V 4.1.1.

Installation of OSX V4.2 completed at 20:17

Adding history entry in VMI\$ROOT:[SYSUPD]VMSINSTAL.HISTORY

Creating installation data file: VMI\$ROOT:[SYSUPD]OSX031.VMI_DATA

Enter the products to be processed from the next distribution volume set.

* Products:

VMSINSTAL procedure done at 20:17

\$

Installation Procedure

Entire Access for TCP/IP is installed using `VMSINSTAL`, the command procedure that is used to install software products in the OpenVMS environment. `VMSINSTAL` guides you through the installation procedure step by step.

Note:

For an update installation, `JTQUOTA` must be set to at least 8192.

Step 1 Run the `VMSINSTAL` Procedure

1. Log in to the system manager's account.

Establish the default directory `SYSS$UPDATE`:

```
$ set default sys$update
```

2. Enter the following command:

```
$ @vmsinstal
```

During the installation procedure, a number of general information messages are displayed. Read all messages carefully and follow any advice they may provide.

The following messages are displayed when the procedure is started:

```
OpenVMS AXP Software Product Installation Procedure V7.1
```

```
It is <dd-mmm-yyyy> at <hh:mm>.
```

```
Enter a question mark (?) at any time for help.
```

where <dd-mmm-yyyy> and <hh:mm> are the current date and time.

If DECnet is active on the system, the following message appears:

```
%VMSINSTAL-W-DECNET, Your DECnet network is up and running.
```

If other users are accessing the system, the following message appears:

```
%VMSINSTAL-W-ACTIVE, The following processes are still active:
<name>
.
.
.
* Do you want to continue anyway [NO]?
```

where <name> refers to the process name of a user logged into the system. Enter YES and continue processing; the installation of the Entire Access for TCP/IP is not affected if users are active.

The following message is then displayed:

```
* Are you satisfied with the backup of your system disk [YES]?
```

It is not necessary to back up the system disk because the files and directories that are installed by the Entire Access for TCP/IP can be removed easily. The installation of Entire Access does not affect any files on the system directories.

Press ENTER if you are satisfied.

3. The following prompt is displayed:

```
* Where will the distribution volumes be mounted?
```

Enter the device or directory where the distribution medium can be found.

4. Enter the products to be processed from the first distribution volume set.

```
* Products: osx
```

```
* Enter installation options you wish to use (none):
```

If the distribution medium is not already mounted on the specified device, VMSINSTAL asks for the distribution medium to be mounted on the device specified when VMSINSTAL was invoked or when the response to the device prompt was entered. If, for example, Entire Access for TCP/IP is to be installed from the device DISK:, VMSINSTAL will display the following:

```
Please mount the first volume of the set on DISK:.
```

VMSINSTAL then displays the following:

```
* Are you ready ?
```

5. Mount the first volume of the distribution medium.

Enter YES and press ENTER when the volume has been mounted. VMSINSTAL now attempts to mount the distribution medium. If it succeeds, a message is displayed, e.g.:

```
%MOUNT-I-MOUNTED, <label> mounted on _DISK:
```

```
The following products will be processed:
```

```
OSX V4.2
```

```
Beginning installation of OSX V4.2 at <hh:mm>
```

```
%VMSINSTAL-I-RESTORE, Restoring product saveset A ...
```

6. VMSINSTAL displays the following messages:

```
* Print READ_ME_FIRST.411 (queue SYS$PRINT) [YES]:
```

Enter YES or NO.

7. VMSINSTAL displays the following prompt:

```
* Move STARTUP_OSX.COM to SYS$STARTUP? [YES]:
```

Software AG recommends that you accept the default value (YES).

If you enter NO, the startup procedure STARTUP_OSX.COM is moved to the directory SAG\$ROOT:[OSX].

If you accept the default value, the startup procedure STARTUP_OSX.COM is moved to the SYS\$STARTUP directory and will be executed during system startup. The following prompt is displayed:

```
* Enable STARTUP_OSX.COM using SYSMAN? [YES]:
```

Entire Access for TCP/IP is now installed. You are asked for the next product to be installed.

Step 2 Check the Directory Structure

The upper portion of the Entire Access for TCP/IP directory structure is generated during the VMSINSTAL procedure and the logical names pointing to the specified directories are created automatically.

OSX\$MAIN

This directory contains the top level Entire Access for TCP/IP directory for each version. It also contains the STARTUP_OSX.COM file.

The STARTUP_OSX.COM file creates the System Logical Names for OSX\$MAIN, OSX\$VERSION, OSX\$LOG, and OSX\$VTX. It also creates the System Logical Name VTXAPI32 and installs VTXAPI.EXE as a Shareable Image.

Note:

If the System Logical Names do not exist after the OpenVMS system is rebooted, a privileged user can redefine them by invoking STARTUP_OSX.COM, which is located in the SAG\$ROOT:[OSX] directory or the SYS\$STARTUP: directory.

OSX\$VERSION

This directory separates the various releases of Entire Access for TCP/IP from each other. It also establishes the current version of Entire Access for TCP/IP.

OSX\$LOG

This directory contains the trace log files and is used in coordination with the logical names VORTEX_HOST_LOGFILE and VORTEX_HOST_LOGOPTS.

These two logical names are defined automatically during the installation procedure. If they do not exist after the OpenVMS system is rebooted, a privileged user can redefine them.

OSX\$VTX

This directory contains the VTXAPI.EXE executable, which provides the VTX API interface for Natural or other applications, such as C programs. VTXAPI.EXE is installed as a Shareable Image.

If you are using Natural, the logical name VTXAPI32 must point to VTXAPI.EXE. For non-Natural client applications, the logical name VTXAPI must point to VTXAPI.EXE.

Step 3 Perform Startup and Verification

Startup

The procedure LOGIN.COM provides the current version of Entire Access for TCP/IP.

Logical Names

The logical names that are used in the Entire Access environment and their definitions for version 4.2 are shown below:

(LNM\$SYSTEM_TABLE)

"OSX\$LOG"	=	"SAG\$ROOT:[OSX.V411.LOG]"
"OSX\$MAIN"	=	"SAG\$ROOT:[OSX]"
"OSX\$VERSION"	=	"SAG\$ROOT:[OSX.V411]"
"OSX\$VTX"	=	"SAG\$ROOT:[OSX.V411.VTX]"
"VTXAPI32"	=	"SAG\$ROOT:[OSX.V411.VTX]VTXAPI.EXE"

Also see the commented sections of LOGIN.COM in the OSX\$MAIN directory.

If the logical names are not already present in the group or system table, they should be added as shown. The following privileges are required:

- The GRPNAM privilege is required in order to enter a Logical Name into the Group Logical Name table.
- The SYSNAM privilege is required in order to enter a Logical Name into the System Logical Name table.

Warning:

To redefine the logical names OSX\$MAIN, OSX\$VERSION, OSX\$LOG, and OSX\$VTX, invoke the STARTUP_OSX.COM located in the SAG\$ROOT:[OSX] directory. Redefining these logical names manually is not recommended.

Natural Requirement

Natural users require an additional logical name. If it is missing, a privileged user can define it as follows:

```
$ASSIGN/SYSTEM/EXEC "NATBIN:NATOSQEnatvers.EXE" NATOSQE
```

where 'natvers' is the current Natural version. For example:

```
$ASSIGN/SYSTEM/EXEC "NATBIN:NATOSQE411.EXE" NATOSQE
```

Note:

Natural is linked to the Entire Access API, so no relink of Natural with Entire Access is necessary.

DEFINING DATA SOURCES TO NATURAL

You must define the data sources your application programs are to access. This chapter tells you how to define data sources to Natural clients on UNIX, Windows 98, Windows 2000, Windows NT, and AXP OpenVMS platforms.

Natural Global Configuration File

Each data source must be defined in the Natural global configuration file NATCONF.CFG. For more information about modifying the Natural configuration file, refer to the installation instructions for Natural.

The steps for defining the data sources are the same on all client platforms:

1. Access the Natural global configuration file (NATCONF.CFG).
2. Define each data source to Natural.
3. Save the updated Natural global configuration file.

These steps are described below in detail for each client platform: UNIX, Windows 98, Windows 2000, Windows NT, and AXP OpenVMS.

UNIX

Access the Natural Global Configuration File

1. Enter the command **natparm** at the system prompt to display the Natural Parameter Setting menu.
2. Select Configuration; if this option is not displayed on the menu, you do not have authorization to modify the configuration files.
3. Select the Global Configuration File option.
4. Select the DBMS Assignment option to display the options for defining the data source(s), as described in the following section.

Define the Data Source(s)

The DBMS assignment includes the “connect string” that Entire Access for TCP/IP uses to establish the connection with the data source.

Perform the following steps for each data source you want to define:

1. In the DBID entry field, specify a unique database ID.
2. In the DBMS Type entry field, specify **OSX or SQL**; use this value for each data source.
3. In the DBMS Parameter entry field, specify a connect string as described on the following pages.
4. In the Modify/Delete entry field, enter **M** (Modify) and press ENTER.

Save the Updated Natural Global Configuration File

1. When you have defined all the data sources, exit the DBMS Assignment window.
2. Select the “Save to Global Configuration File” option and press ENTER.
3. Exit the Natural Parameter Setting function.

Windows 98, Windows 2000, and Windows NT

Access the Natural Global Configuration File

1. Invoke the NATPARM utility by double-clicking on the NATPARM Utility icon in the Natural Program Group or by entering the following command at the command prompt:

natparm

The Natural Parameter Setting window appears. If the Configuration option is not displayed in the menu bar, you do not have permission to modify the configuration files.

2. Select Global Configuration File from the Configuration menu.
3. Select DBMS Assignments from the options menu; the Global DBMS Assignments dialog box appears.

Define the Data Source(s)

The global DBMS assignment includes the “connect string” that Entire Access for TCP/IP uses to establish the connection with the data source.

► Repeat the following series of steps for each data source you want to define:

1. In the DBID box, specify a unique database ID.
2. In the DBMS Type box, specify **OSX** or **SQL**; use this value for each data source.
3. In the DBMS Parameter box, specify a connect string as described on the following pages.
4. Choose Update.

Save the Updated Natural Global Configuration File

1. When you have defined all the data sources, choose Close to end the Global DBMS Assignments dialog.
2. Select Global Configuration File from the Configuration menu.
3. Select Save to File.

AXP OpenVMS

Access the Natural Global Configuration File

1. Invoke the Natural Parameter utility NATPARM at the command prompt.
The Natural Parameter Setting window appears. If the Configuration option is not displayed in the menu bar, you do not have permission to modify the configuration files.
2. Select Global Configuration File from the Configuration menu.
3. Select DBMS Assignments from the options menu; the Global DBMS Assignments dialog box appears.

Define the Data Source(s)

The global DBMS assignment includes the “connect string” that Entire Access for TCP/IP uses to establish the connection with the data source.

Repeat the following series of steps for each data source you want to define:

1. In the DBID box, specify a unique database ID.
2. In the DBMS Type box, specify **OSX** or **SQL**; use this value for each data source.
3. In the DBMS Parameter box, specify a connect string as described on the following pages.
4. Choose Update.

Save the Updated Natural Global Configuration File

1. When you have defined all the data sources, choose Close to end the Global DBMS Assignments dialog.
2. Select Global Configuration File from the Configuration menu.
3. Select Save to File.

Local Client Connect Strings

A local connect string is used when the client application and the server are located on the same UNIX machine.

Syntax for Local Client Connect Strings

The syntax for a local database connect string is as follows:

dbms:db-name

—where

dbms specifies the Entire Access for TCP/IP database driver to be used and is required.

db-name must be the name that was specified when the database was created. It is required by most, but not all, databases and may or may not be case-sensitive. ORACLE, for example, does not use database names.

Note:

For ODBC connections, use the data source name instead of the database name.

Sample Local Client Connect Strings

The following table lists data sources and corresponding connect strings:

Data Source	Client Connect String
ORACLE 7.3, 7.3.4, 8.0.x, 8.1.x	ORACLE:
SYBASE DBLIB or CTLIB	SYBASE:pubs2
INFORMIX 7.3, 9.1.3, 9.2	INFORMIX:stores
DB2 V6.1, V7.1	DB25:SAMPLE
INGRES 2.0	INGRES:DB
ESQ 1.4.3	ADABAS:ESQSAS
ADABAS D v10, v11	ADABASD:DATABASE
ADABAS D v11 ODBC	ODBC:nodename:database
DB2 7.1 CLI/ODBC	ODBC:database
INTERSOLV ODBC	ODBC:datasource

Remote Client Connect Strings

The remote connect string is the same for all client platforms, i.e., UNIX, Windows 98, Windows 2000, Windows NT, and AXP OpenVMS.

Syntax for Remote Client Connect Strings

For remote access to a Windows 2000 or Windows NT server, connect the Entire Access network component by specifying “NET”:

NET:*[db-name]@server-number:host-name!driver*

—where

db-name	must be the name that was specified when the data source was created. It is required by most, but not all, data sources and may or may not be case-sensitive. ORACLE, for example, does not use database names.
server-number	is a 4-digit number greater than 1024 and less than or equal to 9999 that identifies the server daemon; it must match the server number you specify when you start the server daemon. See the section Default Server Numbers below.
host-name	identifies the host machine on which the server runs. Enter either the name (as specified in the “/etc/hosts” file) or the Internet address (in “nn.nn.nn.nn” format) of the host.
driver	specifies the database driver to be used.

Sample Remote Client Connect Strings

The following table lists data sources and corresponding connect strings:

Data Source	make	Client Connect String
ORACLE 7	make oracle73 make oracle734	NET:@5001:node!osxhost.ora7
ORACLE 8	make oracle805 make oracle806 make oracle815 make oracle816	NET:@5001:node!osxhost.ora8
SYBASE 11 DBLIB	make sybase	NET:sybdb@5002:node!osxhost.sybdb
SYBASE 11 or 12 CTLIB	make sybase11 make sybase12	NET:sybdb@5002:node!osxhost.sybct
INFORMIX 7.3 INFORMIX 9.13 INFORMIX 9.2	make informix73 make informix913 make informix92	NET:infdb@5005:node!osxhost.inf
DB2 v6.1 DB2 v7.1	make db261 make db271	NET:db2db@5007:node!osxhost.db2
INGRES 2.0	make ingres2	NET:ingdb@5008:node!osxhost.ing
ESQ 1.4.3	make esq143	NET:ESQDB@5009:node!osxhost.esq
ADABAS D 10	make adabasd10	NET:esddb@5010:node!osxhost.ada
ADABAS D 11	make adabasd11	NET:aaddb@5010:node!osxhost.ada
DB2 v7.1 ODBC	make db2cli	NET:db2db@5007:node!osxhost.odbcdb2
ADABAS D 11 ODBC	make adabasodbc	NET:node:aaddb@5010:node!osxhost.odbcسد
INTERSOLV	make intersolv	NET:ds@5011:node!osxhost.odbcint

GENERATING AND LOADING NATURAL DDMS

A Natural program can access a table or view in a relational database only if the structure has been defined to Natural. You can do this by creating a Natural data definition module (DDM) from the table or view.

Using the Natural DDM Editor

You can use the Natural DDM editor to generate Natural DDMS from SQL tables or views on UNIX, Windows 95, Windows 98, and Windows NT machines running Natural. For more information, see the *Natural User's Guide for UNIX* or the *Natural User's Guide for Windows*.

Generating and Loading DDMS On OpenVMS

On machines running Natural for OpenVMS, you can generate and load DDMS on the local machine, or you can generate the DDMS on another machine and port them to the local machine using the Natural SYSTRANS utility.

Generating DDMs

Since the DDM name must match the name of the database table or view, you can create only one DDM for each table or view. Software AG therefore recommends that the DDM name include the table creator.

Natural for OpenVMS incorporates a Natural DDM facility that enables you to create Natural DDMs from SQL tables.

Perform the following steps to create a DDM:

1. Ensure that you have defined the correct connect string for your data source.
2. Start Natural with the parameter module you specify.
3. Enter Natural Services.
4. Enter DDM Services.
5. Enter DDM Maintenance.
6. Position the cursor on <CREATE> and press `ENTER`.
7. Enter the DBID.
8. Define the Table Owner and Table Name (both case sensitive) or use the default value (*) to see a list of all possible tables by all possible users.
9. Enter the User ID and Password for the database.
10. Select the appropriate table.
11. Press <Esc> and choose Stow (with Exit).
12. Rename the DDM as necessary.

Loading the Generated DDMs

The Natural SYSTRANS utility is used to load the DDMs from the work file WRKF01.dat into the target library. For more information, refer to the installation instructions for Natural for OpenVMS.

Perform the steps below to load the DDMs:

1. Use the Edit function of the Natural NATPARM utility to define work file 1 using the Workfiles (WORK) profile parameter. Specify **WRKF01.dat** for the file name and specify the complete path to WRKF01.dat.
2. Start Natural with the parameter module you just modified.
3. Enter **LOGON SYSTRANS** on the command line of the screen.
4. Enter the command **MENU** to display the Natural Transfer Utility Main Menu.
5. Select the Load Transfer Objects function, set the User-Defined Conversion Table option to **N**, and press ENTER to display the Load Objects screen.
6. Select object type **D** (for DDMs) and press ENTER to display the Load DDMs window.
7. Specify an asterisk (*) for the DDM Name.
8. In the Library entry field, enter the name of the target Natural library that you specified in the DDM Library entry field during the DDM generation.
9. Press ENTER to begin loading the DDMs.

SUPPLYING USER ID AND PASSWORD

This describes the use of Natural variables, which are not part of the Entire Access product.

If your RDBMS requires a user ID and password, you can specify them either before starting the Natural session or during the Natural session.

When you submit your user ID and password using environment variables before starting the Natural session, the prompt window is suppressed and the program executes without interruption.

When you access a database for the first time during a session, a window appears and prompts you for your database user ID and password. The Natural program stops executing until you enter a valid ID and password.

Before starting a Natural session, **clients** using Natural 4.1 can specify database (DB, the default) authentication, operating system (OS) authentication, or both (DB_OS) using the environment variable `SQL_DATABASE_LOGIN`:

```
SQL_DATABASE_LOGIN={DB | OS | DB_OS}
```

Clients can then specify user ID and password using the environment variables for database authentication `SQL_DATABASE_USER` and `SQL_DATABASE_PASSWORD` or the environment variables for operating system authentication `SQL_OS_USER` and `SQL_OS_PASSWORD`, or both.

Once the Natural session starts, only database authentication (the default) is available for clients. The `SQLCONNECT` statement (see page 80) makes it possible to specify user IDs and passwords dynamically so that you can access different databases within a single Natural session. User ID and password can be specified either before or after the Natural session starts.

UNIX Clients

Before Starting the Natural Session

To set the authentication type (client only), enter the following statement in your AUTOEXEC.BAT file:

```
SQL_DATABASE_LOGIN=authentication-type
```

Database Authentication

For database authentication (both client and server), enter the following statements in your AUTOEXEC.BAT file:

```
SQL_DATABASE_USER=db-user-id  
SQL_DATABASE_PASSWORD=db-password
```

Note:

If you want to access multiple databases from a single user session, your user ID and password must be the same for each database.

Operating System Authentication

For operating system authentication (client only), enter the following statements in your AUTOEXEC.BAT file:

```
SQL_OS_USER=os-user-id  
SQL_OS_PASSWORD=os-password
```

Windows 98, Windows 2000, Windows NT Clients

Before Starting the Natural Session

To set the authentication type (client only), enter the following statement in your AUTOEXEC.BAT file:

```
SET SQL_DATABASE_LOGIN=authentication-type
```

Database Authentication

For database authentication (both client and server), enter the following statements in your AUTOEXEC.BAT file:

```
SET SQL_DATABASE_USER=db-user-id  
SET SQL_DATABASE_PASSWORD=db-password
```

Note:

If you want to access multiple databases from a single user session, your user ID and password must be the same for each database.

Operating System Authentication

For operating system authentication (client only), enter the following statements in your AUTOEXEC.BAT file:

```
SET SQL_OS_USER=os-user-id  
SET SQL_OS_PASSWORD=os-password
```

AXP OpenVMS Clients

Before Starting the Natural Session

To set the authentication type (client only), enter the following statement in your AUTOEXEC.BAT file:

```
SQL_DATABASE_LOGIN="authentication-type"
```

Database Authentication

For database authentication (both client and server), enter the following statements in your AUTOEXEC.BAT file:

```
SQL_DATABASE_USER="db-user-id"  
SQL_DATABASE_PASSWORD="db-password"
```

Note:

If you want to access multiple databases from a single user session, your user ID and password must be the same for each database.

Operating System Authentication

For operating system authentication (client only), enter the following statements in your AUTOEXEC.BAT file:

```
SQL_OS_USER="os-user-id"  
SQL_OS_PASSWORD="os-password"
```



APPENDIX A – USING NATURAL WITH ENTIRE ACCESS FOR TCP/IP

This chapter covers the following topics:

- Generating Natural DDMs.
- Natural OPRB parameter settings to control program commits.
- Considerations and restrictions when using Natural DML and SQL statements with Entire Access for TCP/IP.
- Using Flexible SQL for SQL syntax extensions.
- Restrictions and requirements when using Natural with certain RDBMSs.
- Conversion between Natural data formats and RDBMS-specific data types.
- How to obtain diagnostic information about database errors.

Entire Access for TCP/IP supports Natural SQL statements and most Natural DML statements. Natural DML and SQL statements can be used in the same Natural program. At compile time, if a DML statement references a DDM for a data source defined in NATCONF.CFG with DBMS type “OSQ”, Natural translates the DML statement into an SQL statement.

Natural converts DML and SQL statements into calls to Entire Access for TCP/IP. Entire Access for TCP/IP converts the requests to the data formats and SQL dialect required by the target RDBMS and passes the requests to the database driver.

For more information about Natural DML and SQL statements, refer to the *Natural Statements Manual*.



Generating Natural DDMs

A Natural program can access a table or view in a relational database only if the structure has been defined to Natural. You can do this by creating a Natural data definition module (DDM) from the table or view.

On machines running Natural version 4.1 under Windows 98, Windows 2000, Windows NT, or UNIX, and AXP OpenVMS, you can use the Natural DDM editor to generate Natural DDMs from SQL tables or views.

For more information about generating Natural DDMs, refer to the Natural documentation for the platform.

Setting Profile Parameters

The OPRB Parameter

This parameter can be set only by Natural administrators.

The Natural OPRB profile parameter controls transaction processing during a Natural session. It is required, for instance, if a single logical transaction is to span two or more Natural programs. In this case, Natural must **not** issue an END TRANSACTION command (that is **not** “commit”) at the termination of a given Natural program.

The OPRB parameter is set to

- “OFF” (the default), Natural issues an END TRANSACTION statement (that is, **automatically** “commits”) at the end of a Natural program if the Natural session is not at ET status.
- “NOOPEN”, Natural does **not** issue an END TRANSACTION command (that is, does **not** “commit”) at the end of a Natural program.

The NOOPEN setting thus enables a single logical transaction to span more than one Natural program.



Natural DML Statements

The following table shows how Natural translates DML statements into SQL statements:

DML Statement	SQL Statement
BACKOUT TRANSACTION	ROLLBACK
DELETE	DELETE WHERE CURRENT OF <i>cursor-name</i>
END TRANSACTION	COMMIT
EQUAL ... OR	IN (...)
EQUAL ... THRU ...	BETWEEN ... AND ...
FIND ALL	SELECT
FIND NUMBER	SELECT COUNT (*)
HISTOGRAM	SELECT COUNT (*)
READ LOGICAL	SELECT ... ORDER BY
READ PHYSICAL	SELECT ... ORDER BY
SORTED BY ... [DESCENDING]	ORDER BY ... [DESCENDING]
STORE	INSERT
UPDATE	UPDATE WHERE CURRENT of <i>cursor-name</i>
WITH	WHERE

Note:

Boolean and relational operators function the same way in DML and SQL statements.



Entire Access for TCP/IP does not support the following DML statements and options:

- CIPHER
- COUPLED
- FIND FIRST, FIND UNIQUE, FIND ... RETAIN AS
- GET, GET SAME, GET TRANSACTION DATA, GET RECORD
- PASSWORD
- READ BY ISN
- STORE USING/GIVING NUMBER

BACKOUT TRANSACTION

Natural translates a BACKOUT TRANSACTION statement into an SQL ROLLBACK command. This statement reverses all database modifications made after the completion of the last recovery unit. A recovery unit may start at the beginning of a session or after the last END TRANSACTION (COMMIT) or BACKOUT TRANSACTION (ROLLBACK) statement.

Because all cursors are closed when a logical unit of work ends, do not place a BACKOUT TRANSACTION statement within a database loop; place it outside the loop or after the outermost loop of nested loops.

DELETE

The DELETE statement deletes a row from a database table that has been read with a preceding FIND, READ, or SELECT statement. It corresponds to the SQL statement “DELETE WHERE CURRENT OF cursor-name”, which means that only the last row that was read can be deleted.



Example:

```
FIND EMPLOYEES WITH NAME = 'SMITH'  
    AND FIRST_NAME = 'ROGER'  
DELETE
```

Natural translates the Natural statements above into the following SQL statements and assigns a cursor name (for example, CURSOR1). The SELECT statement and the DELETE statement refer to the same cursor.

```
SELECT FROM EMPLOYEES  
    WHERE NAME = 'SMITH' AND FIRST_NAME = 'ROGER'  
DELETE FROM EMPLOYEES  
    WHERE CURRENT OF CURSOR1
```

Natural translates a DELETE statement into an SQL DELETE statement the way it translates a FIND statement into an SQL SELECT statement. For details, see the FIND statement description on page 70.

You cannot delete a row read with a FIND SORTED BY statement. For an explanation, see the FIND statement description on page 70.

You cannot delete a row read with a READ LOGICAL statement. For an explanation, see the READ statement description on page 71.

END TRANSACTION

Natural translates an END TRANSACTION statement into an SQL COMMIT command. The END TRANSACTION statement indicates the end of a logical transaction, commits all modifications to the database, and releases data locked during the transaction.

Because all cursors are closed when a logical unit of work ends, do not place an END TRANSACTION statement within a database loop; place it outside the loop or after the outermost loop of nested loops.

The END TRANSACTION statement cannot be used to store transaction (ET) data when used with Entire Access for TCP/IP.

Note:

Entire Access for TCP/IP does not issue a COMMIT automatically when the Natural program terminates.



FIND

Natural translates a FIND statement into an SQL SELECT statement. The SELECT statement is executed by an OPEN CURSOR command followed by a FETCH command. The FETCH command is executed repeatedly until all records have been read or the program exits the FIND processing loop. A CLOSE CURSOR command ends the SELECT processing.

Example:

Natural statements:

```
FIND EMPLOYEES WITH NAME = 'BLACKMORE'  
    AND AGE EQ 20 THRU 40  
OBTAIN PERSONNEL_ID NAME AGE
```

Equivalent SQL statement:

```
SELECT PERSONNEL_ID, NAME, AGE  
FROM EMPLOYEES  
WHERE NAME = 'BLACKMORE'  
    AND AGE BETWEEN 20 AND 40
```

You can use any table column (field) designated as a descriptor to construct search criteria.

Natural translates the WITH clause of a FIND statement into the WHERE clause of an SQL SELECT statement. Natural evaluates the WHERE clause of the FIND statement *after* the rows have been selected using the WITH clause. View fields may be used in a WITH clause only if they are designated as descriptors.

Natural translates a FIND NUMBER statement into an SQL SELECT statement containing a COUNT(*) clause. When you want to determine whether a record exists for a specific search condition, the FIND NUMBER statement provides better performance than the IF NO RECORDS FOUND clause.

A row read with a FIND statement containing a SORTED BY clause cannot be updated or deleted. Natural translates the SORTED BY clause of a FIND statement into the ORDER BY clause of an SQL SELECT statement, which produces a read-only result table.



HISTOGRAM

Natural translates the HISTOGRAM statement into an SQL SELECT statement. The HISTOGRAM statement returns the number of rows in a table that have the same value in a specific column. The number of rows is returned in the Natural system variable *NUMBER.

Example:

Natural statements:

```
HISTOGRAM EMPLOYEES FOR AGE
OBTAIN AGE
```

Equivalent SQL statements:

```
SELECT AGE, COUNT(*) FROM EMPLOYEES
GROUP BY AGE
ORDER BY AGE
```

READ

Natural translates a READ statement into an SQL SELECT statement. Both READ PHYSICAL and READ LOGICAL statements can be used.

A row read with a READ LOGICAL statement (Example 1) cannot be updated or deleted. Natural translates a READ LOGICAL statement into the ORDER BY clause of an SQL SELECT statement, which produces a read-only result table.

A READ PHYSICAL statement (Example 2) can be updated or deleted. Natural translates it into a SELECT statement without an ORDER BY clause.

Example 1:

Natural statements:

```
READ PERSONNEL BY NAME
OBTAIN NAME FIRSTNAME DATEOFBIRTH
```

Equivalent SQL statement:

```
SELECT NAME, FIRSTNAME, DATEOFBIRTH FROM PERSONNEL
WHERE NAME >= ' '
ORDER BY NAME
```



Example 2:

Natural statements:

```
READ PERSONNEL PHYSICAL  
OBTAIN NAME
```

Equivalent SQL statement:

```
SELECT NAME FROM PERSONNEL
```

When a READ statement contains a WHERE clause, Natural evaluates the WHERE clause **after** the rows have been selected according to the search criterion.

STORE

The STORE statement adds a row to a database table. It corresponds to the SQL INSERT statement.

Example:

Natural statement:

```
STORE RECORD IN EMPLOYEES  
  WITH PERSONNEL_ID = '2112'  
      NAME          = 'LIFESON'  
      FIRST_NAME    = 'ALEX'
```

Equivalent SQL statement:

```
INSERT INTO EMPLOYEES (PERSONNEL_ID, NAME, FIRST_NAME)  
  VALUES ('2112', 'LIFESON', 'ALEX')
```



UPDATE

The DML UPDATE statement updates a table row that has been read with a preceding FIND, READ, or SELECT statement. Natural translates the DML UPDATE statement into the SQL statement “UPDATE WHERE CURRENT OF cursor-name” (a positioned UPDATE statement), which means that only the last row that was read can be updated. In the case of nested loops, the last row in each nested loop can be updated.

UPDATE with FIND/READ

When a DML UPDATE statement is used after a Natural FIND statement, Natural translates the FIND statement into an SQL SELECT statement with a FOR UPDATE OF clause, and translates the DML UPDATE statement into an “UPDATE WHERE CURRENT OF cursor-name” statement.

Example:

```
FIND EMPLOYEES WITH SALARY < 5000
  ASSIGN SALARY = 6000
  UPDATE
```

Natural translates the Natural statements above into the following SQL statements and assigns a cursor name (for example, CURSOR1). The SELECT and UPDATE statements refer to the same cursor.

```
SELECT SALARY FROM EMPLOYEES WHERE SALARY < 5000
  FOR UPDATE OF SALARY
UPDATE EMPLOYEES SET SALARY = 6000
  WHERE CURRENT OF CURSOR1
```

You cannot update a row read with a FIND SORTED BY statement. For an explanation, see the FIND statement description on page 70.

You cannot update a row read with a READ LOGICAL statement. For an explanation, see the READ statement description on page 71.

An END TRANSACTION or BACKOUT TRANSACTION statement releases data locked by an UPDATE statement.



UPDATE with SELECT

The DML UPDATE statement can be used after a SELECT statement only in the following case:

```
SELECT *  
  INTO VIEW view-name
```

Natural rejects any other form of the SELECT statement used with the DML UPDATE statement. Natural translates the DML UPDATE statement into a non-cursor or “searched” SQL UPDATE statement, which means that only an entire Natural view can be updated; individual columns cannot be updated.

In addition, the DML UPDATE statement can be used after a SELECT statement only in Natural structured mode, which has the following syntax:

```
UPDATE [RECORD] [[IN] [STATEMENT]] [(/)]
```

Example:

```
DEFINE DATA LOCAL  
01 PERS VIEW OF SQL-PERSONNEL  
  02 NAME  
  02 AGE  
END-DEFINE  
  
SELECT *  
  INTO VIEW PERS  
  FROM SQL-PERSONNEL  
  WHERE NAME LIKE 'S%'  
  OBTAIN NAME  
  
  IF NAME = 'SMITH'  
    ADD 1 TO AGE  
  UPDATE  
  END-IF  
  
END-SELECT
```

In other respects, the DML UPDATE statement works with the SELECT statement the way it works with the Natural FIND statement (see the section **UPDATE with FIND/READ** on page 73).



Natural SQL Statements

The SQL statements available within the Natural programming language comprise two different sets of statements: the **common set** and the **extended set**.

This section describes considerations and restrictions when using the common set of Natural SQL statements with Entire Access for TCP/IP.

The common set can be handled by each SQL-eligible database system supported by Natural. It basically corresponds to the standard SQL syntax definitions.

For a detailed description of the common set of Natural SQL statements, refer to the *Natural Statements Manual*.

For information about the extended set, refer to the documentation of the Natural interface for your RDBMS.

DELETE

The Natural SQL DELETE statement deletes rows in a table without using a cursor.

Whereas Natural translates the DML DELETE statement into a positioned DELETE statement (that is, an SQL “DELETE WHERE CURRENT OF cursor-name” statement), the Natural SQL DELETE statement is a non-cursor or searched DELETE statement. A searched DELETE statement is a stand-alone statement unrelated to any SELECT statement.

INSERT

The INSERT statement adds rows to a table; it corresponds to the Natural STORE statement.



PROCESS SQL

The PROCESS SQL statement issues SQL statements in a “statement-string” to the database identified by a **ddm-name**.

It is not possible to run database loops using the PROCESS SQL statement.

Refer to the *Natural Statements Manual* for more information.

Parameters

Natural version 4.1 supports the INDICATOR and LINDICATOR clauses. As an alternative, the statement-string may include parameters. The syntax item “parameter” is syntactically defined as follows:

$\left[\begin{array}{l} \text{:U} \\ \text{:G} \end{array} \right] \text{:host-variable}$
--

A **host variable** is a Natural program variable referenced in an SQL statement.

SET SQLOPTION option = value

With Entire Access for TCP/IP, you can also specify “SET SQLOPTION option=value” as statement-string. This can be used to specify various options for accessing SQL databases. The options apply only to the database referenced by the PROCESS SQL statement.

Supported options are:

- DATEFORMAT
- DBPROCESS (for Sybase only)
- TIMEOUT (for Sybase only)
- TRANSACTION (for Sybase only)



DATEFORMAT

This option specifies the format used to retrieve SQL Date and Datetime information into Natural 4.1 fields of type A. The option is obsolete if Natural 4.1 fields of type D or T are used. A subset of the Natural date and time edit masks can be used:

YYYY	Year (4 digits)
YY	Year (2 digits)
MM	Month
DD	Day
HH	Hour
II	Minute
SS	Second

If the date format contains blanks, it must be enclosed in apostrophes.

Examples:

To use ISO date format, specify

```
PROCESS SQL sql-ddm << SET SQLOPTION DATEFORMAT = YYYY-MM-DD >>
```

To obtain date and time components in ISO format, specify

```
PROCESS SQL sql-ddm << SET SQLOPTION DATEFORMAT = 'YYYY-MM-DD HH:II:SS' >>
```

Note:

The DATEFORMAT is evaluated only if data are retrieved from the database. If data are passed to the database, the conversion is done by the database system. Therefore, the format specified with DATEFORMAT should be a valid date format of the underlying database.

If no DATEFORMAT is specified for Natural 4.1 fields,

- the default date format DD-MON-YY is used (where “MON” is a 3-letter abbreviation of the English month name) and
- the following default datetime formats are used:

ADABAS D	YYYYMMDDHHIISS
DB2	YYYY-MM-DD-HH.II.SS
INFORMIX	YYYY-MM-DD HH:II:SS



OpenIngres	DD-MON-YYYY HH:II:SS
ODBC	YYYY-MM-DD HH:II:SS
ORACLE	YYYYMMDDHHIISS
SYBASE DBLIB	YYYYMMDD HH:II:SS
SYBASE CTLIB	YYYYMMDD HH:II:SS
other	DD-MON-YY

DBPROCESS

This option is valid for Sybase databases only.

This option is used to influence the allocation of SQL statements to Sybase DBPROCESSes. DBPROCESSes are used by Entire Access for TCP/IP to emulate database cursors, which are not provided by the Sybase DBlib interface. Two values are possible: MULTIPLE (default) and SINGLE. The specified value can only be changed if no database loop is active.

With DBPROCESS set to MULTIPLE, each SELECT statement uses its own secondary DBPROCESS, whereas all other SQL statements are executed within the primary DBPROCESS. The value MULTIPLE therefore enables your application to execute further SQL statements, even if a database loop is open. It also allows nested database loops.

With DBPROCESS set to SINGLE, all SQL statements use the same (that is, the primary) DBPROCESS. It is therefore not possible to execute a new database statement while a database loop is active, because one DBPROCESS can only execute one SQL batch at a time. Since all statements are executed in the same (primary) DBPROCESS, however, this setting enables SELECTIONs from non-shared temporary tables.

Note:

Since the DBPROCESS option only applies to the Sybase DBlib interface, your application should use a central CALLNAT statement to change the value (at least for SINGLE), so that you can easily remove these calls once Sybase client libraries are supported. Your application should also use a central error handling that establishes the default setting (MULTIPLE).

TIMEOUT

This option is valid for Sybase databases only.

With Sybase, Entire Access for TCP/IP uses a timeout technique to detect database-access deadlocks. The default timeout period is 8 seconds. With this option, you can change the duration of the timeout period (in seconds).

**Example:**

To set the timeout period to 30 seconds, specify

```
PROCESS SQL sql-ddm << SET SQLOPTION TIMEOUT = 30 >>
```

TRANSACTION

This option is valid for Sybase databases only.

This option is used to enable or disable transaction mode. It becomes effective after the next END TRANSACTION or BACKOUT TRANSACTION statement.

If transaction mode is enabled (this is the default), Natural automatically issues all required statements to begin a transaction.

Examples:

To disable transaction mode, specify

```
PROCESS SQL sql-ddm << SET SQLOPTION TRANSACTION = NO >>
...
END TRANSACTION
```

To enable transaction mode, specify

```
PROCESS SQL sql-ddm << SET SQLOPTION TRANSACTION = YES >>
...
END TRANSACTION
```

SQLDISCONNECT

With Entire Access for TCP/IP, you can also specify “SQLDISCONNECT” as the statement-string. In combination with the SQLCONNECT statement (see page 80 below), this statement can be used to access different databases by one application within the same session, by simply connecting and disconnecting as required.

A successfully performed SQLDISCONNECT statement clears the information previously provided by the SQLCONNECT statement; that is, it disconnects your application from the currently connected SQL database determined by the DBID of the DDM used in the PROCESS SQL statement. If no connection is established, the SQLDISCONNECT statement is ignored. It will fail if a transaction is open.

Note:

If Natural reports an error in the SQLDISCONNECT statement, the connection status does not change. If the database reports an error, the connection status is undefined.



SQLCONNECT option = value

With Entire Access for TCP/IP, you can also specify “SQLCONNECT option=value” as the statement-string. This statement can be used to establish a connection to an SQL database according to the DBID specified in the DDM addressed by the PROCESS SQL statement. The SQLCONNECT statement will fail if the specified connection is already established.

Note:

If the SQLCONNECT statement fails, the connection status does not change.

Supported options are:

- USERID
- PASSWORD
- OS_PASSWORD (with Natural 4.1)
- OS_USERID (with Natural 4.1)
- DBMS_PARAMETER

If several options are specified, they must be separated by a comma. The options are evaluated as described below.

The specified value can be either a character literal or a Natural variable of format A. If Natural performs an implicit reconnect, because the connection to the database was lost, the values provided by the SQLCONNECT statement are used.

USERID and PASSWORD

Specifying USERID and PASSWORD for the database logon suppresses the default logon window and the evaluation of the environment variables SQL_DATABASE_USER and SQL_DATABASE_PASSWORD.

If only USERID is specified, PASSWORD is assumed to be blank, and vice versa. If neither USERID nor PASSWORD is specified, default logon processing applies.

Note:

With database systems that do not require user ID and password, a blank user ID and password can be specified to suppress the default logon processing.



OS_USERID and OS_PASSWORD

These options are valid for use with Natural 4.1 only.

Specifying OS_PASSWORD and OS_USERID for the operating system logon suppresses the logon window and the evaluation of the environment variables SQL_OS_USER and SQL_OS_PASSWORD.

If only OS_USERID is specified, OS_PASSWORD is assumed to be blank, and vice versa. If neither OS_USERID nor OS_PASSWORD is specified, default logon processing applies.

Note:

With operating systems that do not require user ID and password, a blank user ID and password can be specified to suppress the default logon processing.

DBMS_PARAMETER

Specifying DBMS_PARAMETER dynamically overwrites the DBMS Parameter definition in the Natural global configuration file.

Examples:

```
PROCESS SQL sql-ddm << SQLCONNECT USERID = 'DBA', PASSWORD = 'SECRET' >>
```

This example connects to the database specified in the Natural global configuration file with user ID “DBA” and password “SECRET”.

```
DEFINE DATA LOCAL
1 #UID (A20)
1 #PWD (A20)
end-define
```

```
INPUT 'Please enter ADABAS D user ID and password' / #UID / #PWD
```

```
PROCESS SQL sql-ddm << SQLCONNECT USERID = : #UID,
                          PASSWORD      = : #PWD,
                          DBMS_PARAMETER = 'ADABASD:mydb'
>>
```

This example connects to the ADABAS D database “mydb” with the user ID and password taken from the INPUT statement.

```
PROCESS SQL sql-ddm << SQLCONNECT USERID = ' ', PASSWORD = ' ',
                          DBMS_PARAMETER = 'DB2:EXAMPLE' >>
```



This example connects to the DB2 database “EXAMPLE” without specifying user ID and password (since these are not required by DB2, which uses the operating system user ID).

SELECT

The INTO clause and scalar operators for the SELECT statement either are RDBMS-specific and do not conform to the standard SQL syntax definitions (the Natural common set), or impose restrictions when used with Entire Access for TCP/IP.

Entire Access for TCP/IP does not support the INDICATOR and LINDICATOR clauses in the INTO clause. Thus, Entire Access for TCP/IP requires the following syntax for the INTO clause:

```
INTO { parameter, ...  
      VIEW {view-name},... }
```

The concatenation operator (//) does not belong to the common set and therefore is not supported by Entire Access for TCP/IP.

Refer to the *Natural Statements Manual* for more information.

SELECT SINGLE

The SELECT SINGLE statement provides the functionality of a non-cursor SELECT operation (singleton SELECT); that is, a SELECT statement that retrieves a maximum of one row without using a cursor.

This statement is similar to the Natural FIND UNIQUE statement. However, Natural automatically checks the number of rows returned. If more than one row is selected, Natural returns an error message.

If your RDBMS does not support dynamic execution of a non-cursor SELECT operation, the Natural SELECT SINGLE statement is executed like a set-level SELECT statement, which results in a cursor operation. However, Natural still checks the number of returned rows and issues an error message if more than one row is selected.



UPDATE

The Natural SQL UPDATE statement updates rows in a table without using a cursor.

Whereas Natural translates the DML UPDATE statement into a positioned UPDATE statement (that is, the SQL “UPDATE WHERE CURRENT OF cursor-name” statement), the Natural SQL UPDATE statement is a non-cursor or searched UPDATE statement. A searched UPDATE statement is a stand-alone statement unrelated to any SELECT statement.

Flexible SQL

Flexible SQL allows you to use arbitrary RDBMS-specific SQL syntax extensions. Flexible SQL can be used as a replacement for any of the following syntactical SQL items:

- atom
- column reference
- scalar expression
- condition

The Natural compiler does not recognize the SQL text used in flexible SQL; it simply copies the SQL text (after substituting values for the **host variables**, which are Natural program variables referenced in an SQL statement) into the SQL string that it passes to the RDBMS. Syntax errors in flexible SQL text are detected at runtime when the RDBMS executes the string.

Note the following characteristics of flexible SQL:

- It is enclosed in “<<” and “>>” characters and can include arbitrary SQL text and host variables.
- Host variables *must* be prefixed by a colon (:).
- The SQL string can cover several statement lines; comments are permitted.



Flexible SQL can also be used between the clauses of a select expression:

```
SELECT selection
<< ... >>
INTO ...
FROM ...
<< ... >>
WHERE ...
<< ... >>
GROUP BY ...
<< ... >>
HAVING ...
<< ... >>
ORDER BY ...
<< ... >>
```

Examples:

```
SELECT NAME
FROM EMPLOYEES
WHERE << MONTH (BIRTH) >> = << MONTH (CURRENT_DATE) >>
```

```
SELECT NAME
FROM EMPLOYEES
WHERE << MONTH (BIRTH) = MONTH (CURRENT_DATE) >>
```

```
SELECT NAME
FROM EMPLOYEES
WHERE SALARY > 50000
<< INTERSECT
  SELECT NAME
  FROM EMPLOYEES
  WHERE DEPT = 'DEPT10'
>>
```



RDBMS-Specific Requirements and Restrictions

This section discusses restrictions and special requirements for Natural and some RDBMSs used with Entire Access for TCP/IP.

Case-Sensitive Database Systems

In case-sensitive database systems, use lowercase characters for table and column names, since all names specified in a Natural program are automatically converted to lowercase.

Note:

This restriction does not apply when you use flexible SQL.

SYBASE and Microsoft SQL Server

To execute SQL statements against SYBASE and Microsoft SQL Server, you must use one or more DBPROCESS structures. A DBPROCESS can execute SQL command batches.

A command batch is a sequence of SQL statements. Statements must be executed in the sequence in which they are defined in the command batch. If a statement (for example, a SELECT statement) returns a result, you must execute the statement first and then fetch the rows one by one. Once you execute the next statement from the command batch, you can no longer fetch rows from the previous query.

With SYBASE and Microsoft SQL Server, an application can use more than one DBPROCESS structure; therefore, it is possible to have nested queries if you use a separate DBPROCESS for each query. Because SYBASE and Microsoft SQL Server lock data for each DBPROCESS, however, an application that uses more than one DBPROCESS can deadlock itself. Natural times out in case of a deadlock.



How Natural Statements are Converted to Database Calls

Natural uses one DBPROCESS for each open query and another DBPROCESS for all other SQL statements (UPDATE, DELETE, INSERT, ...).

If a query is referenced by a positioned UPDATE or DELETE statement, Natural automatically appends the FOR BROWSE clause to the generated SELECT statement to allow UPDATES while rows are being read.

For a positioned UPDATE or DELETE statement, the SYBASE “dbqual” function is used to generate the following search condition:

WHERE *unique-index = value AND tsequal (timestamp,old-timestamp)*

This search condition can be used to reselect the current row from the query. The “tsequal” function checks whether the row has been updated by another user.

Natural Restrictions with SYBASE and Microsoft SQL Server

The following restrictions apply when using Natural with SYBASE and Microsoft SQL Server.

Case-Sensitivity

SYBASE and Microsoft SQL Server are case-sensitive, and Natural passes parameters in lowercase. Thus, if your SYBASE and Microsoft SQL Server tables or fields are defined in uppercase or mixed case, you must use database SYNONYMS or Natural flexible SQL.

Positioned UPDATE and DELETE Statements

To support positioned UPDATE and DELETE statements, the table to be accessed must have a unique index and a timestamp column. In addition, the timestamp column must not be included in the select list of the query.



Querying Rows

SYBASE and Microsoft SQL Server lock pages, and locked pages are owned by DBPROCESS structures.

Pages locked by an active DBPROCESS cannot subsequently be read (by the same or another DBPROCESS) until the lock is released by an END TRANSACTION or BACKOUT TRANSACTION statement.

Therefore, if you have updated, inserted, or deleted a row in a table:

- Do not start a new SELECT (FIND, READ, ...) loop against the same table.
- Do not fetch additional rows from a query that references the same table if the SELECT statement has no FOR BROWSE clause.

Natural automatically appends the FOR BROWSE clause if the query is referenced by a positioned UPDATE or DELETE statement.

Transaction/Non-Transaction Mode

SYBASE and Microsoft SQL Server differentiate between transaction and non-transaction mode. In transaction mode, Natural connects to the database allowing INSERTs, UPDATEs and DELETEs to be issued; thus, commands that run in non-transaction mode, for example, CREATE TABLE, cannot be issued.

Stored Procedures

It is possible to use stored procedures in SYBASE and Microsoft SQL Server using the PROCESS SQL statement; however, the stored procedures must **not** contain

- commands that work only in non-transaction mode; or
- return values.



Data Type Conversion

When a Natural program accesses data in a relational database, Entire Access for TCP/IP converts RDBMS-specific data types to Natural data formats, and vice versa. The tables in this section show how Natural data formats correspond to data types in the following RDBMS:

- ADABAS D
- ADABAS SQL Server
- DB2
- INFORMIX
- OpenIngres
- ORACLE
- SYBASE and Microsoft SQL Server

Note:

Format/lengths in the tables apply to Natural version 4.1 and above.

The following tables provide the format and length for data accessed using DDMs created with the Natural DDM editor. See the Natural 4.1 documentation for more information.

The date/time or datetime format specific to a particular RDBMS can be converted into the Natural D and T formats. See the section **Date/Time Conversion** on page 96 for more information.



ADABAS D

RDBMS Data Type	Format/Length
boolean	L
char(<i>n</i>)	<i>An</i>
date	A10
long	A253 ¹
fixed(<i>n</i>)	<i>Nn</i>
fixed(<i>y,x</i>)	<i>Nx-y.y</i>
float(<i>n</i>)	<i>Nn</i> ²
time	A8
timestamp	A26

Notes:

1. *The maximum supported length of “long” fields is limited by 253 * (maximum length of occurrences of an MU field) = 48323.*
2. *If necessary (for example, to increase performance), fields of this format can be modified by the user so that the database field and Natural DDM variable match.*



ADABAS SQL Server

RDBMS Data Type	Format/Length
char(<i>n</i>)	<i>An</i>
char(5)	—
char(253)	—
decimal(5)	I2
decimal(10.4)	N6.4
double precision	N10.6
float(1...21)	N2.6
float(22...53)	N10.6
integer	I4
numeric(5)	I2
numeric(10.4)	N6.4
real	F4
smallint	I2

DB2

RDBMS Data Type	Format/Length
date	A10
decimal(5)	I2
decimal(10,4)	N6,4
fixed character(5)	A5
float	F8
large integer	I4
scientific notation	N10,6
small integer	I2
special data	A253
system date and time	A10
time	A10
variable character (250)	A250



INFORMIX

RDBMS Data Type	Format/Length
byte	A56
char(5)	A5
date	A10
datetime	A26
datetime year to year	—
decimal	N16
decimal(10.4)	N6.4
double precision	F8
float	F8
integer	I4
interval minute to second	A17
money	N14.2
numeric	—
real	F4
serial	I4
smallfloat	F4
smallint	I2
text	A56
varchar(5)	A5



OpenIngres

RDBMS Data Type	Format/Length
c	A1
char(<i>n</i>)	<i>An</i>
date	A20
float	F8
float4	F4
integer	I4
integer1	I1
money	N12.2
object_key	A16
smallint	I2
table_key	B8
text(<i>n</i>)	<i>An</i>
varchar(<i>n</i>)	<i>An</i>



ORACLE

RDBMS Data Type	Format/Length
char(<i>n</i>)	<i>An</i>
date	A14
decimal	N29 *
float	F8
integer	I4
long	A253
long raw(<i>n</i>)	<i>Bn</i>
number	N29 *
number(<i>n</i>)	<i>Nn</i>
number(<i>x,y</i>)	<i>Nx-y.y</i>
raw(<i>n</i>)	<i>Bn</i>
smallint	I4
varchar(<i>n</i>)	<i>An</i>
varchar2(<i>n</i>)	<i>An</i>

* If necessary (for example, to increase performance), fields of this format can be modified by the user so that the database field and Natural DDM variable match.



SYBASE and Microsoft SQL Server

RDBMS Data Type	Format/Length
binary(<i>n</i>)	<i>Bn</i>
bit	N1
char(<i>n</i>)	<i>An</i>
datetime	A26
float	F8
image	B126
int	I4
money	N15.4
nchar(<i>n</i>)	<i>An</i>
nvarchar(<i>n</i>)	<i>An</i>
real	F4
smalldatetime	A26
smallint	I2
smallmoney	N6.4 *
text	A253
timestamp	B8
tinyint	I2
varbinary(<i>n</i>)	<i>Bn</i>
varchar(<i>n</i>)	<i>An</i>

* If necessary (for example, to increase performance), fields of this format can be modified by the user so that the database field and NATURAL DDM variable match.

Date/Time Conversion

Using Natural 4.1 and above, the RDBMS-specific date/time or datetime format can be converted into the Natural D and T formats.

To use this conversion, you must first edit the Natural DDM to change the date or time field formats from A(lphanumeric) to D(ate) or T(ime). The SQLOPTION DATEFORMAT is obsolete for fields with format D or T.

Note:

Date or time fields converted to Natural D(ate)/T(ime) format may not be mixed with those converted to Natural A(lphanumeric) format.

For update commands, Natural 4.1 and above converts the Natural Date and Time format to the database-dependent representation of DATE/TIME/DATETIME to a precision level of seconds.

For retrieval commands, Natural converts the returned database-dependent character representation to the internal Natural Date or Time format. (See detailed conversion table for more information.)

For Natural Date variables, the time portion is ignored and initialized to zero.

For Natural Time variables, tenth of seconds are ignored and initialized to zero.

Note:

*For retrieval commands, the date component of Natural Time is **not** ignored and is initialized to 0000-01-02 (YYYY-MM-DD) if the RDBMS's time format does not contain a date component.*



Conversion Tables

ADABAS D

RDBMS Formats	Natural Date	Natural Time
DATE	YYYYMMDD	
TIME		00HHIISS

DB2

RDBMS Formats	Natural Date	Natural Time
DATE	YYYY-MM-DD	
TIME		HH.II.SS

INFORMIX

RDBMS Formats	Natural Date	Natural Time
DATETIME, year to day	YYYY-MM-DD	
DATETIME, year to second (other formats are not supported)		YYYY-MM-DD-HH:II:SS*

OpenIngres

RDBMS Formats	Natural Date	Natural Time
DATE (only II_DATE_FORMAT US is supported)	DD-MON-YYYY	DD-MON-YYYYHH:II:SS*



ODBC

RDBMS Formats	Natural Date	Natural Time
DATE	YYYY-MM-DD	
TIME		HH:II:SS

ORACLE

RDBMS Formats	Natural Date	Natural Time
DATE (ORACLE session parameter NLS_DATE_FORMAT is set to YYYYMMDDHH24MISS)	YYYYMMDD000000 (ORACLE time component is set to null for update commands and ignored for retrieval commands.)	YYYYMMDDHHIISS *

SYBASE

RDBMS Formats	Natural Date	Natural Time
DATETIME	YYYYMMDD	YYYYMMDD HH:II:SS *

* When comparing two time values, remember that the date components may have different values.



Obtaining Diagnostic Information

If the database returns an error while being accessed, you can call the non-Natural program CMOSQERR to obtain diagnostic information about the error.

Call CMOSQERR using the following syntax:

```
CALL 'CMOSQERR' parm1 parm2
```

The parameters are described in the following table:

Parameter	Format/Length	Description
parm1	I4	The number of the error returned by the database.
parm2	A70	The text of the error returned by the database.

APPENDIX B – INTERSOLV ODBC

This appendix provides examples for using INTERSOLV ODBC on UNIX systems. The examples apply specifically to HP-UX and can be customized for other UNIX systems.

Note:

*Some versions of INTERSOLV ODBC may issue the following **warning** when connected:*

*[INTERSOLV][ODBC SQL Server driver] Error on input or output to a file. No such file or directory.
Additional Information /opt/odbc/odbc.ini.*

To bypass this error message, set the following environment variables on the server side before starting the INTERSOLV ODBC server:

```
VORTEX_HOST_LOGFILE=/tmp/host  
VORTEX_HOST_LOGOPTS=FULL  
export VORTEX_HOST_LOGFILE VORTEX_HOST_LOGOPTS
```

“chatr” Requirement for the ODBC Server

```
$ chatr +s enable osxhost.odbcint
osxhost.odbcint:
  current values:
    shared executable
    shared library dynamic path search:
      SHLIB_PATH      disabled  second
      embedded path   disabled  first  Not Defined
    internal name:
      osxhost.odbcint
    shared library list:
      dynamic  /opt/odbc/lib/libodbc.sl
      dynamic  /opt/odbc/lib/libodbcinst.sl
      dynamic  /usr/lib/libC.1
      dynamic  /usr/lib/libM.1
      dynamic  /usr/lib/libdld.1
      dynamic  /usr/lib/libc.1
    shared library binding:
      deferred
    static branch prediction disabled
    data page size: 4K
    instruction page size: 4K
  new values:
    shared executable
    shared library dynamic path search:
      SHLIB_PATH      enabled   second
      embedded path   disabled  first  Not Defined
    internal name:
      osxhost.odbcint
    shared library list:
      dynamic  /opt/odbc/lib/libodbc.sl
      dynamic  /opt/odbc/lib/libodbcinst.sl
      dynamic  /usr/lib/libC.1
      dynamic  /usr/lib/libM.1
      dynamic  /usr/lib/libdld.1
      dynamic  /usr/lib/libc.1
    shared library binding:
      deferred
    static branch prediction disabled
    data page size: 4K
    instruction page size: 4K
$
```

“chatr” Requirement for the ODBC Client

```
$ chatr +s enable natural
natural:
  current values:
    shared executable
    shared library dynamic path search:
      SHLIB_PATH      disabled  second
      embedded path  disabled  first  Not Defined
    internal name:
      natural
    shared library list:
      dynamic  /opt/odbc/lib/libodbc.sl
      dynamic  /opt/odbc/lib/libodbcinst.sl
      dynamic  /usr/lib/libC.1
      dynamic  /usr/lib/libM.1
      dynamic  /usr/lib/libdld.1
      dynamic  /usr/lib/libc.1
    shared library binding:
      deferred
    static branch prediction disabled
    data page size: 4K
    instruction page size: 4K
  new values:
    shared executable
    shared library dynamic path search:
      SHLIB_PATH      enabled   second
      embedded path  disabled  first  Not Defined
    internal name:
      natural
    shared library list:
      dynamic  /opt/odbc/lib/libodbc.sl
      dynamic  /opt/odbc/lib/libodbcinst.sl
      dynamic  /usr/lib/libC.1
      dynamic  /usr/lib/libM.1
      dynamic  /usr/lib/libdld.1
      dynamic  /usr/lib/libc.1
    shared library binding:
      deferred
    static branch prediction disabled
    data page size: 4K
    instruction page size: 4K
```

\$

“odbc.ini” Examples for the ODBC Server

```
[ODBC Data Sources]
myds=
Oracle7=
dBase=
Sybase=
Informix=
OpenIngres=
DB2=
Text=

[dBase]
QEWS=35691
Driver=/opt/odbc/lib/ivdbf12.sl
Description=dBase
Database=/opt/odbc/demo

[Sybase]
QEWS=35691
Driver=/opt/odbc/lib/ivsyb12.sl
Description=Sybase
Database=pubs2
ServerName=SYBASE
WorkstationID=id
LogonID=sa
Password=mypwd
OptimizePrepare=2
SelectMethod=1

[Oracle7]
Driver=/opt/odbc/lib/ivor712.sl
Description=Oracle7
ServerName=usrshp2
LogonID=scott
Password=tiger

[Informix]
QEWS=35698
Driver=/opt/odbc/lib/ivinf12.sl
Description=Informix7
Database=osqdev
HostName=usrshp2
LogonID=informix
Password=mypwd
```

```
[DB2]
Driver=/opt/odbc/lib/ivdb212.s1
Description=DB2
Database=waHgp0

[OpenIngres]
Driver=/opt/odbc/lib/ivoing12.s1
ServerName=ingreshost
Database=sysmaster
LogonID=odbc01
Password=odbc01
Workarounds=1

[Text]
Driver=/opt/odbc/lib/ivtxt12.s1
Description=Text driver
Database=/opt/odbc/demo

[ODBC]
Trace=0
TraceFile=odbctrace.out
TraceDll=/opt/odbc/lib/odbctrac.s1
InstallDir=/opt/odbc

[myds]
Driver=/opt/odbc/lib/ivor712.s1
Description=Oracle7
ServerName=usrshp2
LogonID=scott
Password=tiger

[Infodbc]
QEWSID=35698
Driver=/opt/odbc/lib/ivinf12.s1
Description=Informix7
Database=osqdev
HostName=usrshp2
LogonID=informix
Password=mypwd
```

```
[Sybodbc]
QEWS=35691
Driver=/opt/odbc/lib/ivs12.sl
Description=Sybase
Database=pubs2
ServerName=SYBASE
WorkstationID=id
LogonID=sa
Password=myspw
OptimizePrepare=2
SelectMethod=1
```

Sample SYBASE ODBC Variables

```
DSQUERY=SYBASE
INFORMIXDIR=/opt/odbc
IV_GLS_LCDIR=/opt/odbc/gls/lc11
IV_GLS_REGISTRY=/opt/odbc/gls/cm3/registry
ODBCHOME=/opt/odbc
ODBCINI=/opt/odbc/odbc.ini
ODBC_INI=/opt/odbc/odbc.ini
PATH=/usr/bin/X11:/bin:/usr/bin:/etc:/usr/etc:/usr/contrib/bin:
    /usr/root:/usr/lib:/usr/lib/acct:/usr/local/bin:./SAG/ada/v22313:/SAG/ada/v22313/tools:/SAG/nat/v21226/bin:/SAG/wcp/v2117:/SAG/wcp/v2117/bin:/SAG/osq/v211/vtx:/SAG/esd/v6112/bin:/SAG/esq/v1413/tools:/SAG/esq/v1413/bin:./SAG/wcp/ushpux1:/SAG/demoweb/sh:/RDBMS/oracle/app/oracle/orahome/product/7.3.2/bin:/RDBMS/sybase/bin:/RDBMS/sybase/install:/RDBMS/sybase/include:/opt/odbc/lib:/opt/odbc/bin:/RDBMS/sybase/lib:/RDBMS/sybase/bin:/usr/lib
SHLIB_PATH=/opt/odbc/lib:/RDBMS/sybase/lib:/RDBMS/sybase/bin:/usr/lib
SYBASE=/RDBMS/sybase
SYBASE_TERM=vt220
```

Sample INFORMIX ODBC Variables

```
INFORMIXDIR=/RDBMS/informix
INFORMIXSERVER=learn_online
IV_GLS_LCDIR=/opt/odbc/gls/lc11
IV_GLS_REGISTRY=/opt/odbc/gls/cm3/registry
ODBCHOME=/opt/odbc
ODBCINI=/opt/odbc/odbc.ini
ODBC_INI=/opt/odbc/odbc.ini
ONCONFIG=onconfig.learn
PATH=/usr/bin/X11:/bin:/usr/bin:/etc:/usr/etc:/usr/contrib/bin:
    /users/root:/usr/lib:/usr/lib/acct:/usr/local/bin:./SAG/ada/v22313:/SAG/ada/v22313/tools:/SAG/nat/v21226/bin:/SAG/wcp/v2117:/SAG/wcp/v2117/bin:/SAG/osq/v211/vtx:/SAG/esd/v6112/bin:/SAG/esq/v1413/tools:/SAG/esq/v1413/bin:./SAG/wcp/ushpux1:/SAG/demoweb/sh:/opt/odbc/lib:/opt/odbc/bin:/RDBMS/informix/bin
SHLIB_PATH=/opt/odbc/lib:/RDBMS/informix/lib:/RDBMS/informix/lib/esql:/RDBMS/informix/bin:/usr/lib
TERM=vt100
TERMCAP=/RDBMS/informix/etc/termcap
```




APPENDIX C – TRANSLATION TABLES

Entire Access for JDBC requires special ASCII/EBCDIC translations for code pages. Although many language code pages could be supplied, the following code pages are supplied in this release:

- A2E is the ASCII/EBCDIC Code Page Translation for English (Default)
- AUSGER is the ASCII/EBCDIC Code Page Translation for Austrian German
- A2EGER is the ASCII/EBCDIC Code Page Translation for German

You can set environment variable `VORTEX_CCMP_FILE` in the `LIB ENVFILE` to point to the translation table that you want to use.

The current default translation table is A2E for English. Unless you have a good reason for changing this default, do not use this variable.



APPENDIX D — INSTALLING A CLIENT UNDER UNIX: AN ALTERNATE PROCEDURE

This appendix provides an alternate procedure for installing the Entire Access for TCP/IP client component under UNIX.

For information about the client and server version levels, UNIX compiler support, and hardware and operating system requirements, see the section **Before You Install** on page 23.



Installation Procedure

Each of the following steps is valid for any supported UNIX platform. Where applicable, actions specific to a particular product, such as DB2 or ADABAS SQL Server, are identified.

Step 1 Log In at the UNIX System Prompt

Log in as “sag”; do **not** log in as “root”.

Step 2 Generate the Environment File

The environment file “sagenv.new” must be modified to include the environment variables required before using Entire Access for TCP/IP.

Note:

If you have an existing “sagenv.old” environment file, be sure to rename it; otherwise, it will be overwritten later in this step when the “sagenv.new” file is automatically generated and the existing “sagenv.new” file is renamed to “sagenv.old”.

Execute the interactive SAGAINST script to generate the “sagenv.new” file.

1. To start the script, enter the following commands:

```
cd $SAG  
./SAGAINST
```

The script ensures that the SAG environment variable has been established; it then displays a list of all products in the supplied \$SAG directory.

2. Select each required product from the list by entering the corresponding number (from the left-hand column) after the prompt. Use spaces to separate the numbers.

Be sure to select the correct version of Entire Access for TCP/IP.

Do not select more than one version of a product. In the following example, Natural version 4.1.2.6 and Entire Access for TCP/IP version 4.2.1 are selected:



```
INSTALL: ENVIRONMENT
```

```
Please choose products for which you want to
generate the environment file sagenv.new
```

```
1      ada/v31119
2      nat/v4126
3      osx/v421
```

```
PLEASE SELECT ITEMS : 2 3
```

3. Press ENTER to generate the “sagenv.new” file.

The generated “sagenv.new” file includes all environment variables required to use the selected products. If “sagenv.new” already exists, it is automatically renamed to “sagenv.old” and the previous “sagenv.old” is overwritten.

4. If you are performing an update installation (that is, you selected only the new products to be added to your existing “sagenv” file), use the concatenate command to append the “sagenv.new” to your existing “sagenv” file.
5. (Optional) Rename “sagenv.new” to another file name; the following steps assume that the environment file was renamed to “sagenv”.
6. To establish the modified environment variables, invoke the “sagenv” file with the following command:
`./sagenv`
7. To automatically establish this environment each time you log in, add the following command to your “.profile” file:
`./SAG-home-directory/sagenv`

The file will be executed automatically each time you log in.



Step 3 Select the Database Drivers

This step is required for Natural for UNIX users only.

Use the interactive “osxlibs.sh” script to select the database drivers(s) to be used by Entire Access for TCP/IP.

1. To change your directory, enter the following command:

cd \$OSXDIR/\$OSXVERS/bin

2. To start the script, enter the following command:

osxlibs.sh

A list of database drivers appears.

3. Select each desired driver by entering the corresponding number (from the left-hand column) after the prompt and pressing ENTER.

To deselect a database driver, reenter the number for that driver at the prompt and press ENTER.

Each selected entry is indicated by an asterisk (*) to the left of the number column. In the following example, the selected entry (*) is local ADABAS SQL Server 1.4.3.

```

Entire Access for TCP/IP (HP-UX 11 32-bit)
=====

  1 - remote Entire Access Net          12 - local INGRES 2.0
*  2 - local ADABAS SQL Server 1.4.3    13 - local ORACLE 7.3
  3 - local ADABAS D 10.0               14 - local ORACLE 7.3.4
  4 - local ADABAS D 11.0               15 - local ORACLE 8.0.5
  5 - local ADABAS D ODBC               16 - local ORACLE 8.0.6
  6 - local DB2 v5 or v6.1              17 - local ORACLE 8.1.5
  7 - local DB2 v7.1                    18 - local ORACLE 8.1.6
  8 - local DB2 CLI V7                  19 - local SYBASE 11 DBLIB
  9 - local INFORMIX 7.3 (online)       20 - local SYBASE 11 CTLIB
 10 - local INFORMIX IDS 9.1.3          21 - local SYBASE 12 CTLIB
 11 - local INFORMIX IDS 9.2            22 - local INTERSOLV ODBC

g - Generate 'osxlibs.lst'
q - Exit

please select an entry:
g

```

If you want to use ADABAS, ADABAS SQL Server, Natural Security, Natural, or Remote Procedure Call (RPC), you must specify the “ada=” parameter when relinking Natural (see step 8).



*Note for Natural Security users:
ADABAS must be selected for Natural Security to function.*

4. After making your selections, enter “g” and press ENTER.

A confirmation screen appears that lists the values of the environment variables found for the drivers you selected; see the following example.

```
Entire Access for TCP/IP
=====
```

You have selected the following parts to build Natural

```
- local adabas sql server version
```

```
$OSXDIR    = /usr/SAG/osx
$OSXVERS   = v421
$NATDIR    = /usr/SAG/nat
$NATVERS   = v4126
$ESQDIR    = /usr/SAG/esq
$ESQVERS   = v143
```

Press <enter> to see the file ' /usr/SAG/osx/v421/osxlibs.lst

5. Verify that the environment variables are correct; then press ENTER to generate the “osxlibs.lst” and “ddmlibs.lst” files. The screen displays the contents of these files as they are being generated.

The “osxlibs.lst” file contains a list of all database libraries to be linked to the Natural prelinked object “natraw.o”. The “make” file uses the “osxlibs.lst” file to build the new Natural environment in step 7.



Step 4 Relink Natural on UNIX Client Machines

This step is required for Natural 4.1 and above.

Regenerate your Natural nucleus with the selected Entire Access for TCP/IP database drivers.

1. Change to the Natural build directory by entering the following command:

```
cd $NATDIR/$NATVERS/bin/build
```

2. Enter a command to build a new Natural nucleus that includes support for the database drivers selected in step 3.

If you do not require Natural Security, Natural RPC, ADABAS, or ADABAS SQL Server, enter the command as follows:

```
make natural osx=yes
```

If you do require Natural Security, Natural RPC, ADABAS, or ADABAS SQL Server, enter the command with the “ada=[*adabas-library*]” specification, as follows:

```
make natural osx=yes ada= [adabas-library]
```

where *adabas-library* has one of the following values:

dyn	link in ADABAS shared libraries
stat	link in ADABAS static libraries
cscidyn	link in ADABAS and CSCI shared libraries
cscistat	link in ADABAS and CSCI static libraries

Whether you should link ADABAS as static or dynamic depends on the version of ADABAS you are using. Refer to your ADABAS installation instructions for more information.

Notes:

1. *By default, ADABAS is not included when Natural is relinked. If you fail to specify a value for the “ada=” parameter, ADABAS will not be linked into Natural and Natural Security will not function.*
2. *Users who require Natural RPC or ADABAS SQL Server must specify the ada= parameter for CSCI components (cscistat or cscidyn) to include a required CSCI stub module.*
3. To copy this new Natural file into the “bin” directory, enter the command **make install**.



APPENDIX E — CLIENT AND SERVER TRACES

Client Tracing

This section contains examples that illustrate how to perform tracing on the available client platforms.

UNIX Client

```
$ VORTEX_API_LOGFILE=C:\trace  
$ VORTEX_API_LOGOPTS=FULL  
$ export VORTEX_API_LOGFILE VORTEX_API_LOGOPTS
```

Windows NT client

Control Panel → System → Environment

Variable: VORTEX_API_LOGFILE
Value: C:\trace
Variable: VORTEX_API_LOGOPTS
Value: FULL

AXP OpenVMS Client

```
$ ASSIGN SYS$LOGIN:TRACE.LOG VORTEX_API_LOGFILE  
$ ASSIGN FULL VORTEX_API_LOGOPTS
```

OS/390 Client

Tracing is not supported on the OS/390 client platform.



Server Tracing

This section contains examples that illustrate how to perform tracing on the available server platforms.

UNIX Server

```
$ VORTEX_HOST_LOGFILE=C:\htrace
$ VORTEX_HOST_LOGOPTS=FULL
$ export VORTEX_HOST_LOGFILE VORTEX_HOST_LOGOPTS
```

Windows NT Server

Control Panel → System → Environment

```
Variable:  VORTEX_HOST_LOGFILE
Value:     C:\htrace
Variable:  VORTEX_HOST_LOGOPTS
Value:     FULL
```

AXP OpenVMS Server

```
$ ASSIGN SYS$LOGIN:HTRACE.LOG VORTEX_HOST_LOGFILE
$ ASSIGN FULL VORTEX_HOST_LOGOPTS
```

OS/390 Server

For Started Tasks only set the following in HLQ.OSX411.LIB(ENVFILE):

```
VORTEX_HOST_LOGFILE=DUMMY
VORTEX_HOST_LOGOPTS=FULL
```

Check the Started Task log for VTX9STC. This log would then also contain the trace.

INDEX

A

Adabas, open/close processing, OPRB profile parameter, 66
API, for database access, 5, 7
Application environment supported, 35

B

BACKOUT TRANSACTION statement, 68
Bind command, for DB2 database, under UNIX, 20

C

CD, installation, 36
Client-Server, processing, 2
Close, database, OPRB profile parameter, 66
CMOSQERR program, for trouble shooting, 99

D

Data
 access
 local, 6
 remote
 using TCP/IP, 7
 using third-party network products, 8
 conversion, 88
 format, 88
 source
 connection, 50, 51
 definition, 50, 51

 type, 88
 conversion
 Adabas D, 89
 Adabas SQL Server, 90
 DB2, 91
 INFORMIX, 92
 INGRES, 93
 Microsoft SQL Server, 95
 ORACLE, 94
 SYBASE, 95
 Data source definition, OpenVMS, 52
 Data sources, accessing, 6
 Database
 assignment, 49
 authentication
 OpenVMS, 64
 UNIX, 62
 Windows, 63
 driver, 5
 local, 5
 name
 for local connect string, 53
 for localconnect string, 53
 for remote connect string, 55
 open/close processing, OPRB profile parameter, 66
 password
 OpenVMS, 64
 UNIX, 62
 Windows, 63
 remote, 5
 user ID
 OpenVMS, 64
 UNIX, 62
 Windows, 63
 DB2
 bind, under UNIX, 20
 library links, db2l command, 12

Entire Access for TCP/IP Manual for UNIX

DBMS

- assignment, 50, 51
- parameter, 50, 51
- type, 50, 51

DBMS assignment, OpenVMS, 52

DDM

- generation, 57
- generator, creation of (UNIX), 19, 29, 115

DDMs, creating, 57

DELETE statement

- positioned, 68
- searched, 75

Disk space requirement, 36

Driver name, for remote connect string, 55

E

END TRANSACTION statement, 69

Environment file, under UNIX, 15, 25, 112

Environment variables for RDBMSs, 13

F

FIND statement, 70

G

GRPNAM privilege, 47

H

HISTOGRAM statement, 71

Host name, for remote connect string, 55

I

INFORMIX, special considerations, 12

INSERT statement, 75

Installation

of client component, 31

prerequisites

for client component

under Windows 2000, 32

under Windows 98, 32

under Windows NT, 32

under UNIX, 9, 23

procedure, with TCP/IP, under UNIX, 15, 25

procedure under UNIX, alternate, 112

under UNIX, with TCP/IP, 9, 23

under Windows 2000, prerequisites for client component, 32

under Windows 98, prerequisites for client component, 32

under Windows NT, prerequisites for client component, 32

Installation example, 38

Installation Kit, Structure, 37

Installation prerequisites, SAGBASE, 35

Installation procedure, 42

J

JTQUOTA, for an update installation, 42

L

Local, data access, 6

LOGIN.COM, 46

M

Microsoft SQL Server, restrictions and limitations, 85

N

NATPARM utility, for data source definition, 49
 NATPARM utility, accessing global configuration file, 52
 Natural
 configuration file, under UNIX, 49
 DDMs, 57
 environment, building of (UNIX), 19, 29, 115
 for
 UNIX, 9, 23, 24
 Windows 98, 32
 Windows NT and Windows 2000, 32
 logical name requirement, 47
 NATPARM utility, 49
 relinking, on UNIX client machines, 29, 116
 restrictions
 with Microsoft SQL Server, 86
 with SYBASE, 86
 statements
 DML, 5, 67
 SQL, 5, 75
 Natural for OpenVMS Version 2.2.3.26, 35
 Natural Security, relinking Natural, on UNIX
 client machines, 29, 116
 Natural SYSTRANS utility, 57

O

Open/close processing, OPRB profile parameter, 66
 Operating system requirements, 35
 OPRB profile parameter, 66
 ORACLE, special considerations, 12
 OSX directories, 45
 OSX231 save sets, 37

P

PRCLM parameter setting, 36
 Prerequisites, installation, 36
 Process privileges, for Alpha AXP servers, 36
 PROCESS SQL statement, 76

R

READ statement, 71
 Remote, data access
 using TCP/IP, 7
 using third-party network products, 8
 Remote access, using TCP/IP, 32
 Requirements, operating system, 35

S

SAGBASE, 35
 Save sets, 37
 SELECT SINGLE statement, 82
 SELECT statement
 cursor-oriented, 82
 non-cursor , 82
 Server
 daemon, linkage of (UNIX), 19
 number, for remote connect string, 55
 Shell script, for database drivers (UNIX), 17, 27, 114
 SQL
 flexible, 83
 statements, 75
 DELETE, 75
 INSERT, 75
 PROCESS SQL, 76
 SELECT, 82
 SELECT SINGLE, 82
 UPDATE, 83

Entire Access for TCP/IP Manual for UNIX

Statements

DML

- BACKOUT TRANSACTION, 68
- DELETE, 68
- END TRANSACTION, 69
- FIND, 70
- HISTOGRAM, 71
- READ, 71
- STORE, 72
- UPDATE, 73

SQL

- DELETE, 75
- INSERT, 75
- PROCESS SQL, 76
- SELECT, 82
- SELECT SINGLE, 82
- UPDATE, 83

STORE statement, 72

Supported, databases

- Adabas C, 2
- Adabas D, 2
- Adabas SQL Server, 2
- DB2 for AIX, 2
- DB2 for mainframe, 2
- INFORMIX OnLine, 2
- ODBC-compliant databases, 2
- OpenIngres, 2
- ORACLE, 2
- SYBASE CTLIB, 2
- SYBASE DBLIB, 2

SYBASE, restrictions and limitations, 85

SYSNAM privilege, 47

System Logical Names, created in START-UP_OSX.COM, 45

T

TCP/IP

- for remote access, 32
- for UNIX, 11, 24

TCP/IP (UCX), connectivity, 35

Top level directory, 45

Trace log files directory, 45

Tracing

- client, 117
- server, 118

U

UNIX

- environment file, 15, 25, 112
- installation procedure, 112
- installation under, procedure, 15, 25

UPDATE statement

- searched, 83
- with FIND/READ, positioned, 73
- with SELECT, searched, 74

V

Version directory, 45

VMSINSTAL, used to install Entire Access for TCP/IP, 42

W

WSDEFAULT parameter setting, 36

Notes

Entire Access for TCP/IP Manual for UNIX

Entire Access for TCP/IP Manual for UNIX