

Entire Access for TCP/IP Manual for UNIX

Manual Order Number: OSX511-030UNIX

This document applies to Entire Access for TCP/IP Version 5.1 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

© May 2002, 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	4
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	11
Before You Install	11
Client and Server Version Levels	11
Hardware and Operating System Requirements	12
Other Software Requirements	12
UNIX Compiler Support	13
Database Systems and Versions Overview	13
Database System Preparation	14
Environment Variables	15
Installation Procedure	17
Step 1 Log In at the UNIX System Prompt	17
Step 2 Generate the Environment File	17
Step 3 Select the Local Database Drivers (Local Access Only)	19
Step 4 Select the Server Database Drivers (Remote Access Only)	22
Step 5 Establish ADABAS SQL Server Run-time Parameters	26
The Entire Access Directory Structure Under UNIX	28

3	INSTALLING A CLIENT UNDER UNIX	29
	Before You Install	29
	Client and Server Version Levels	29
	UNIX Compiler Support	29
	Hardware and Operating System Requirements	30
	Other Software Requirements	30
	Installation Procedure	31
	Step 1 Log In at the UNIX System Prompt	31
	Step 2 Generate the Environment File	31
	Step 3 Select the Database Drivers (Natural for UNIX Only)	33
	Step 4 Relink Natural on UNIX Client Machines	36
4	INSTALLING A CLIENT UNDER WINDOWS	37
	Before You Install	37
	Client and Server Version Levels	37
	Client Hardware and Operating System Requirements	37
	Other Software Requirements for Clients	38
	Installation Procedure	38
	Installation CDs	38
	Installation Steps	38
5	INSTALLING A CLIENT UNDER OPENVMS	39
	Before You Install	39
	Operating System Requirements	39
	TCP/IP Connectivity	39
	Supported Application Environment	39
	SAGBASE Requirement	39
	The Installation Tape	40
	Installation Prerequisites	40
	Installation Kit Structure	41
	Example of an Initial Installation	42
	Installation Procedure	46
	Step 1 Run the VMSINSTAL Procedure	46
	Step 2 Check the Directory Structure	49

Step 3 Perform Startup and Verification	51
6 DEFINING DATA SOURCES TO NATURAL	53
Natural Global Configuration File	53
UNIX	54
Windows 98, Windows 2000, Windows NT, Windows Me, Windows XP	55
AXP OpenVMS	56
Local Client Connect Strings	57
Syntax for Local Client Connect Strings	57
Sample Local Client Connect Strings	58
Remote Client Connect Strings	59
Syntax for Remote Client Connect Strings	59
Sample Remote Client Connect Strings	60
Remote Access Commands	61
7 GENERATING AND LOADING NATURAL DDMS	63
Using the Natural DDM Editor	63
Generating and Loading DDMS On OpenVMS	63
Generating DDMS	64
Loading the Generated DDMS	65
8 SUPPLYING USER ID AND PASSWORD	67
UNIX Clients	68
Before Starting the Natural Session	68
Database Authentication	68
Operating System Authentication	68
Microsoft Windows Clients	69
Before Starting the Natural Session	69
Database Authentication	69
Operating System Authentication	69
AXP OpenVMS Clients	70
Before Starting the Natural Session	70
Database Authentication	70
Operating System Authentication	70

APPENDIX A – USING NATURAL WITH ENTIRE ACCESS FOR TCP/IP	71
Generating Natural DDMs	72
Setting Profile Parameters	72
The OPRB Parameter	72
Natural DML Statements	73
BACKOUT TRANSACTION	74
DELETE	74
END TRANSACTION	75
FIND	6
HISTOGRAM	77
READ	77
STORE	78
UPDATE	79
Natural SQL Statements	81
DELETE	81
INSERT	81
PROCESS SQL	82
SELECT	88
SELECT SINGLE	88
UPDATE	89
Flexible SQL	89
Examples:	90
RDBMS-Specific Requirements and Restrictions	91
Case-Sensitive Database Systems	91
SYBASE and Microsoft SQL Server	91
Data Type Conversion	94
ADABAS D	95
ADABAS SQL Server	96
DB2	97
INFORMIX	98
ORACLE	99
SYBASE and Microsoft SQL Server	100

Table of Contents

Date/Time Conversion	101
Conversion Tables	102
Obtaining Diagnostic Information	104
APPENDIX B – INTERSOLV ODBC	105
“chatr” Requirement for the ODBC Server	106
“chatr” Requirement for the ODBC Client	107
“odbc.ini” Examples for the ODBC Server	108
Sample SYBASE ODBC Variables	110
Sample INFORMIX ODBC Variables	111
IBM DB2 CLI	112
APPENDIX C — SERVER TRACES	113
Natural Client Trace	113
Entire Access for TCP/IP Client Trace	113
Entire Access for TCP/IP Server Trace	114
ODBC or CLI Trace	114
APPENDIX D — INSTALLING A CLIENT UNDER UNIX: AN ALTERNATE PROCEDURE	115
Installation Procedure	116
Step 1 Log In at the UNIX System Prompt	116
Step 2 Generate the Environment File	116
Step 3 Select the Database Drivers	118
Step 4 Relink Natural on UNIX Client Machines	120
INDEX	121

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 Platforms*
- *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 Me, Windows XP, Windows NT, and OpenVMS client platforms to access data sources on UNIX, Windows 2000 Server, Windows Me, Windows XP, 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:

Entire Access for TCP/IP Manual for UNIX

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

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	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	
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, Windows Me, Windows XP, 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 Platforms (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 describes the four types of traces that can be generated when using Natural with Entire Access for TCP/IP.

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.

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.

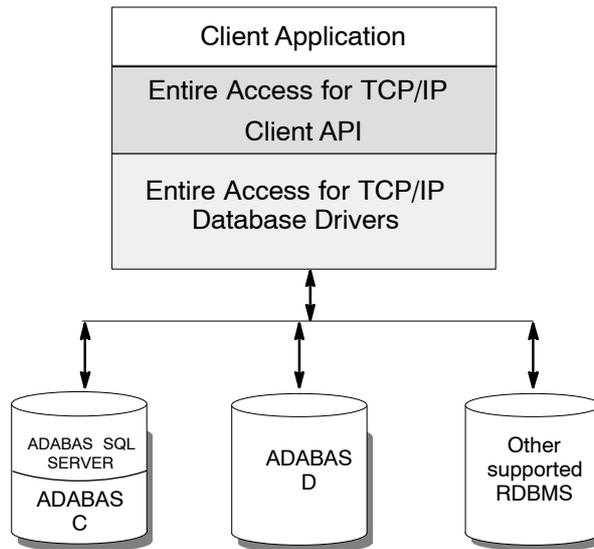
Accessing Data Sources from Clients

The same procedure is used to access data sources from all client platforms: UNIX, Windows 98, Windows 2000, Windows Me, Windows XP, 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:



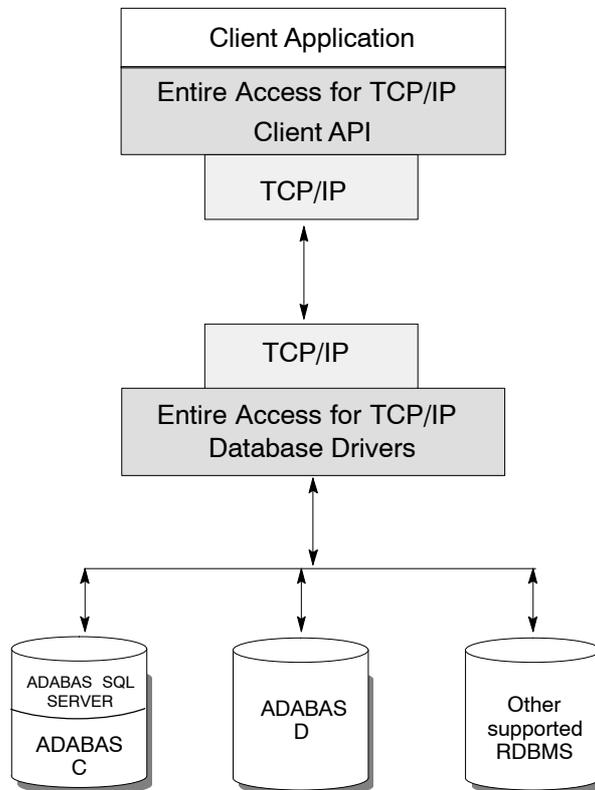
Note:

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.

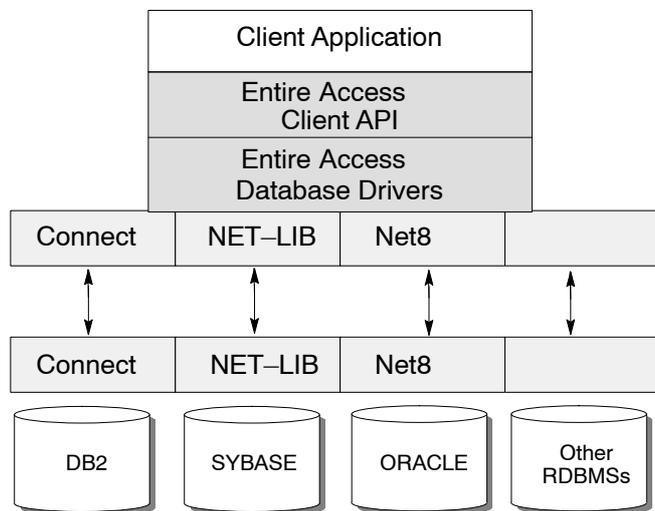
The following diagram shows remote access using TCP/IP:



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.

Note that when networking is performed for the client by a third party it is done transparently. It is also transparent to Entire Access for TCP/IP, which therefore uses only local connect strings. Because Entire Access for TCP/IP is completely unaware of the network rerouting, the local illusion must be completely implemented by the RDBMS vendor performing the networking.

For example, if a local Windows 2000 Server PC is accessing a DB2 on VSE with IBM Connect networking, a local DB2:databasename connect string in Entire Access for TCP/IP on the Windows 2000 Server PC is implied. Because Entire Access for TCP/IP is completely unaware of the Windows 2000 Server to VSE rerouting, the VSE DB2 syntax must be identical to that of the Windows 2000 Server DB2, with no exceptions.

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, Windows Me, Windows XP, 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 105, using HP-UX as an example.

Before You Install

Client and Server Version Levels

Entire Access for TCP/IP supports up to one previous version of the software. For example, version 4.2.1 and version 5.1.1 are interchangeable, as follows:

- The Entire Access for TCP/IP version 4.2.1 client works with the Entire Access for TCP/IP version 5.1.1 server.
- The Entire Access for TCP/IP version 5.1.1 client works with the Entire Access for TCP/IP version 4.2.1 server.

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
Tru64 4.0F	Compaq (Digital ALPHA)
HP-UX 11.0 32-bit or 64-bit	HP9000 series S700 or S800
SCO UnixWare 7.1.1	Intel
SNI Reliant UNIX 5.45B 64-bit	SNI RMx00/R10000
Solaris 7 32-bit	Sun SPARC, Sun Ultra SPARC
Solaris 8 64-bit	Sun SPARC, Sun Ultra SPARC

Note:

All 64-bit platforms support only 64-bit RDBMS. No 32-bit RDBMS are supported on 64-bit platforms.

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 (also known as “Forte”); 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 11.0 and 12.0

DB2 7.1 (uses DB2 CLI)

INFORMIX–IDS 9.2

ORACLE 7.3.4, 8.1.5 and above

SYBASE 11.5+ CTLIB and DBLIB

SYBASE 12 CTLIB

ODBC-compliant (local access)

Note:

OpenIngres is no longer supported.

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-IDS 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 v11	DBROOT	Build/Run
	DBNAME (optional) ¹	Run
	PATH=\$PATH:DBROOT/bin	Run
ADABAS D v12	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 9.2	INFORMIXDIR	Build
	PATH=\$PATH:\$INFORMIXDIR/bin	Run
	SQLEXEC=\$INFORMIXDIR/lib/sqlrm	Run
	INFORMIXSERVER=host-name	Run
	INFORMIXSHMBASE=shmbase / 1024 ³	Run
ORACLE	ORACLE_HOME	Build/Run
	ORACLE_SID	Run
	TWO_TASK (optional) ⁴	Run
SYBASE DBLIB and SYBASE CTLIB	SYBASE	Build
	DSQUERY	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 54.
- 2 DB2 version 7.1 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/v7.1. The DB2_HOME should be set to the **installation home** directory.
- 3 When you use Natural to access a local INFORMIX 9.2 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 9.2 database on AIX, 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 5.1.1 and Entire Access for TCP/IP version 5.1.1 are selected:

```
INSTALL: ENVIRONMENT
```

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

```
1      ada/v31124
2      nat/v511
3      osx/v511
```

```
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 Local Database Drivers (Local Access 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 D 12.0.

```

Entire Access for TCP/IP (Solaris 7 32-bit)
=====

 1 - remote ENTIRE ACCESS NET          8 - local ORACLE 7.3.4
 2 - local ADABAS SQL SERVER 1.4.3     9 - local ORACLE 8i (805+)
 3 - local ADABAS D 11.0               10 - local ORACLE 9i
* 4 - local ADABAS D 12.0              11 - local SYBASE 11 DBLIB
 5 - local ADABAS D 12.0 ODBC          12 - local SYBASE 11 CTLIB
 6 - local DB2 v7                      13 - local SYBASE 12 CTLIB
 7 - local INFORMIX 9.2                14 - 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 chosen to build the following environments

- local adabas d version 12.0

$OSXDIR      = /RDBMS/osxsag/sag/osx
$OSXVERS     = v511
$DBROOT      = /RDBMS/osxsag/sag/aad/v1200

ld -o ADA12.so ada12.o -G -B symbolic osx.a /RDBMS/osxsag/sag/aad/v1200/lib/
libs qlrte.a /RDBMS/osxsag/sag/aad/v1200/lib/pcrlib.a /RDBMS/osxsag/sag/aad/
v1200/lib/pcd2lib.a /RDBMS/osxsag/sag/aad/v1200/lib/pcd3lib.a /RDBMS/osxsag/sag/
aad/v1200/lib/libsqlrte.a /RDBMS/osxsag/sag/aad/v1200/lib/pcdlib.a /RDBMS/osx-
sag/sag/aad/v1200/lib/libsqlptc.a -lnsl -lsocket -lnsl -lsocket

Press <enter> to see the file '/RDBMS/osxsag/sag/osx/v511/osxlibs.1st'

/RDBMS/osxsag/sag/osx/v511/osxlibs.1st:
-----
/RDBMS/osxsag/sag/osx/v511/bin/OSXAPI.so
/RDBMS/osxsag/sag/osx/v511/bin/osx.a
-lnsl
-lsocket

```

Notes:

- 1 *Entire Access for TCP/IP 5.1.1 supports shareable libraries so that, for example, a generic Natural can be built from the files OSXAPI.so and osx.a and the target RDBMS for the client application is in fact determined by the connect string.*
- 2 *This means that Natural can be built once for all RDBMS drivers without specifically loading any RDBMS driver code into the Natural nucleus. For example, from an ADABAS D v12 object called ada12.o an uppercase variant called ADA12.so will be created as the shareable library. ADA12 is the executable responsible for loading the ADA12.so shareable library.*

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

Warning:

IF YOU DO NOT SET THE APPROPRIATE ENVIRONMENT VARIABLE, THE SHAREABLE LIBRARIES MIGHT NOT BE FOUND.

When using shareable libraries with:	Set the variable:
INFORMIX version 9.2	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 7.1.1 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 9.2 under AIX:

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

Note:

Some operating systems may use more than one variable or one that is not included in the above table. Please refer to the operating system vendor's documentation.

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 Select the Server Database Drivers (Remote Access Only)

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 . . .
oracle7	ORACLE 7.3.4
oracle8	ORACLE 8i (8.1.5 and above)
oracle9	ORACLE 9i
sybase11dblib	SYBASE 11 DBLIB
sybase12ctlib	SYBASE 11 CTLIB
sybase12ctlib	SYBASE 12 CTLIB
informix92	INFORMIX 9.2
db27	DB2
esq143	ADABAS SQL Server 1.4.3
adabas11	ADABAS D 11
adabas12	ADABAS D 12
adabas12odbc	ADABAS D 12 ODBC
intersolvodbc	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.

3. Use the interactive “osxopr.sh” script to start a server dispatcher for a specific RDBMS:

```
$ osxopr.sh
```

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

```
1 - show status
2 - start a server
3 - kill a server
4 - ping a server

q - exit
```

```
please select an entry:
1
```

```
Entire Access for TCP/IP (show status)
=====
```

```
Entire Access for TCP/IP directory: /RDBMS/osxsag/sag/osx
Entire Access for TCP/IP version: v511
Available server: ADABAS D 12,
Available utilities: serversingle, serverping, serverkill,
```

```
press <return> to continue
```

```
Entire Access for TCP/IP (show status)
=====
```

```
Available Server Daemon ports:
```

```
list of active client/server connections:
```

TYPE	INST.	OWNER
------	-------	-------

```
press <return> to continue
```

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

```
1 - show status
2 - start a server
3 - kill a server
4 - ping a server
```

```
q - exit
```

```
please select an entry:
```

```
2
```

```
Entire Access for TCP/IP (start a server)
=====
```

```
please enter the server number
```

```
7898
```

```
starting server with number 7898
```

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

```
1 - show status
2 - start a server
3 - kill a server
4 - ping a server
```

```
q - exit
```

```
please select an entry:
```

```
1
```

```
Entire Access for TCP/IP (show status)
=====
```

```
Entire Access for TCP/IP directory:  /RDBMS/osxsag/sag/osx
Entire Access for TCP/IP version:    v511
Available server:                    ADABAS D 12,
Available utilities:                 serversingle, serverping, serverkill,
```

press <return> to continue

```
Entire Access for TCP/IP (show status)
=====
```

```
Available Server Daemon ports:      7898
```

```
list of active client/server connections:
```

TYPE	INST.	OWNER
------	-------	-------

press <return> to continue

Step 5 Establish ADABAS SQL Server Run-time Parameters

Note:

Entire Access for TCP/IP 5.1.1 supports shareable images.

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 as follows:

```
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
```

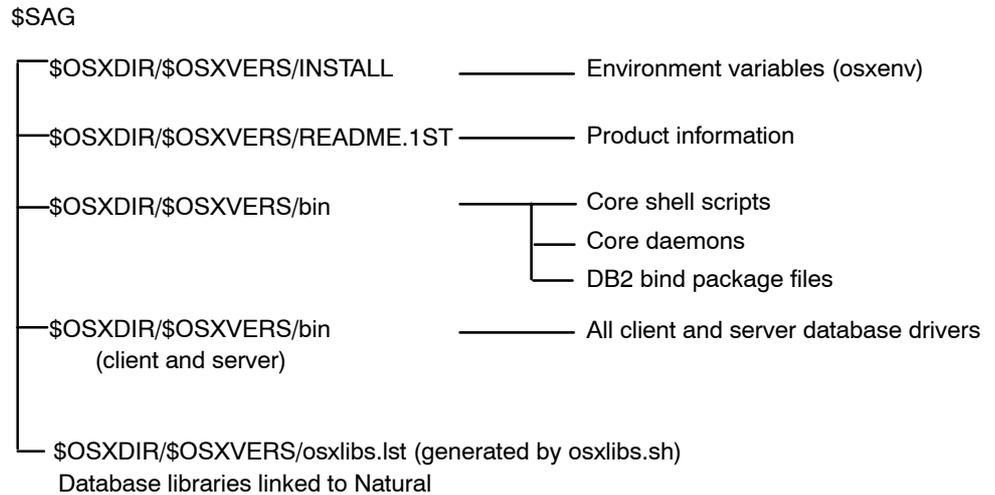
—where

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 115. 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

Entire Access for TCP/IP supports up to one previous version of the software. For example, version 4.2.1 and version 5.1.1 are interchangeable, as follows:

- The Entire Access for TCP/IP version 4.2.1 client works with the Entire Access for TCP/IP version 5.1.1 server.
- The Entire Access for TCP/IP version 5.1.1 client works with the Entire Access for TCP/IP version 4.2.1 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 (also known as “Forte”); 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
Tru64 4.0F	Compaq (Digital ALPHA)
HP-UX 11.0 32-bit or 64-bit 64-bit	HP9000 series S700 or S800
SCO UnixWare 7.1.1	Intel
SNI Reliant UNIX 5.45B	SNI RMx00/R10000
Solaris 7 32-bit	Sun SPARC, Sun Ultra SPARC
Solaris 8 64-bit	Sun SPARC, Sun Ultra SPARC

Note:

All 64-bit platforms support only 64-bit RDBMS. No 32-bit RDBMS are supported on 64-bit platforms.

Other Software Requirements

Entire Access for TCP/IP version 5.1.1 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 5.1.1 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 5.1.1 and Entire Access for TCP/IP version 5.1.1 are selected:

INSTALL: ENVIRONMENT

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

```
1      ada/v31119
2      nat/v511
3      osx/v511
```

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          8 - local ORACLE 7.3.4
 2 - local ADABAS SQL SERVER 1.4.3     9 - local ORACLE 8i (815+)
 3 - local ADABAS D 11.0              10 - local ORACLE 9i
* 4 - local ADABAS D 12.0             11 - local SYBASE 11 DBLIB
 5 - local ADABAS D 12.0 ODBC         12 - local SYBASE 11 CTLIB
 6 - local DB2 v7                    13 - local SYBASE 12 CTLIB
 7 - local INFORMIX 9.2              14 - 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 chosen to build the following environments

- local adabas d version 12.0

$OSXDIR      = /RDBMS/osxsag/sag/osx
$OSXVERS     = v511
$DBROOT      = /RDBMS/osxsag/sag/aad/v1200

ld -o ADA12.so ada12.o -G -B symbolic osx.a /RDBMS/osxsag/sag/aad/v1200/lib/
libs qlrte.a /RDBMS/osxsag/sag/aad/v1200/lib/pcrlib.a /RDBMS/osxsag/sag/aad/
v1200/lib/pcd2lib.a /RDBMS/osxsag/sag/aad/v1200/lib/pcd3lib.a /RDBMS/osxsag/sag/
aad/v1200/lib/libsqlrte.a /RDBMS/osxsag/sag/aad/v1200/lib/pcdlib.a /RDBMS/osx-
sag/sag/aad/v1200/lib/libsqlptc.a -lnsl -lsocket -lnsl -lsocket

Press <enter> to see the file '/RDBMS/osxsag/sag/osx/v511/osxlibs.1st'

/RDBMS/osxsag/sag/osx/v511/osxlibs.1st:
-----
/RDBMS/osxsag/sag/osx/v511/bin/OSXAPI.so
/RDBMS/osxsag/sag/osx/v511/bin/osx.a
-lnsl
-lsocket

```

Notes:

- 1 *Entire Access for TCP/IP 5.1.1 supports shareable libraries so that, for example, a generic Natural can be built from the files OSXAPI.so and osx.a and the target RDBMS for the client application is in fact determined by the connect string.*
- 2 *This means that Natural can be built once for all RDBMS drivers without specifically loading any RDBMS driver code into the Natural nucleus. For example, from an ADABAS D v12 object called ada12.o an uppercase variant called ADA12.so will be created as the shareable library. ADA12 is the executable responsible for loading the ADA12.so shareable library.*

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

Warning:

IF YOU DO NOT SET THE APPROPRIATE ENVIRONMENT VARIABLE, THE SHAREABLE LIBRARIES MIGHT NOT BE FOUND.

When using shareable libraries with:	Set the variable:
INFORMIX version 9.2 under AIX	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 7.1.1	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 9.2 under AIX:

```
$LIBPATH=/RDBMS/informix/lib/esql:/RDBMS/informix/lib:/usrlib
$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 5.1.

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

This chapter tells you how to install the Entire Access for TCP/IP client component under Windows 98, Windows 2000, Windows NT, Windows Me, or Windows XP. 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 5.1.1 on the server can communicate with version 4.2.1 on the client.
- Version 5.1.1 on the client can communicate with version 4.2.1 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
- 128 MB of RAM
- about 4 MB of available hard-disk space (if all options are installed)
- operating system:
 - Windows 98;
 - Windows 2000;
 - Windows NT version 4.0 with Service Pack 6;
 - Windows Me; or
 - Windows XP

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

Installation Steps

To install the Entire Access for TCP/IP client component under Windows 98, Windows 2000, Windows NT, Windows Me, or Windows XP, 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, Windows NT, Windows Me, or Windows XP 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 5.1 or above is used.

INSTALLING A CLIENT UNDER OPENVMS

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

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

Before You Install

Operating System Requirements

To install Entire Access for TCP/IP, you must be running OpenVMS version 7.1 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 must be installed in a Natural for OpenVMS version 3.1 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, 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 Tape

The installation tape 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 41.

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 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
OSX411.A	Installation procedures and files used by VMSINSTAL
OSX411.B	Patch level 0 specific files
OSX411.C	Patch level 0 OSX initialization procedures and files

Example of an Initial Installation

```

$ set proc/priv=all

$ @sys$update:vmsinstal

      OpenVMS AXP Software Product Installation Procedure V7.1

It is 9-DEC-1999 at 20:16.

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

%VMSINSTAL-W-NOTSYSTEM, You are not logged in to the SYSTEM account.
%VMSINSTAL-W-ACTIVE, The following processes are still active:
    ...
    ...

* Do you want to continue anyway [NO]? yes

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

* Where will the distribution volumes be mounted: DISK:

Enter the products to be processed from the first distribution volume set.
* Products: osx

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

The following products will be processed:

    OSX V4.1

      Beginning installation of OSX V4.1 at 20:16

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

+-----+
|                                     |
|           Software AG Product Installation Procedure           |
|                                     |
+-----+

You are preparing a(n)
Entire Access for TCP/IP(OpenVMS/AXP) Version 4.1.1 initial installation
on a DEC 3000 Model 400 with the cpu-id %X80000000/00000000

```

```
%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 SYSLCK
%OSX-I-INSTINFO, required default privilege SYSLCK
%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 V4.1 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 V4.1
```

```
        Beginning installation of OSX V4.1 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].

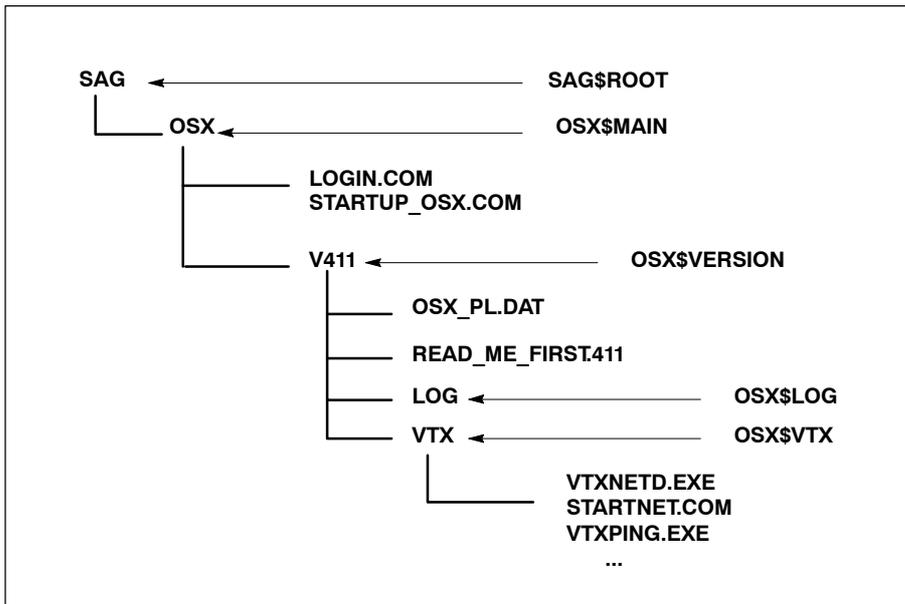
If you accept the default value, the startup procedure STARTUP_OSX.COM is moved to the SYS\$STARTUP directory and will be executed during system startup. The following prompt is displayed:

```
* 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 following files:

File Name	Description
LOGIN.COM	Initializes the VTXKILL, VTXNETD, and VTXPING Global Symbols.
STARTUP_OSX.COM	Creates the System Logical Names for OSX\$MAIN, OSX\$VERSION, OSX\$LOG, OSX\$VTX. 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 following executables:

File Name	Description
TVER.EXE	Provides product version tracking when executables are passed as an argument.
VTXPING.EXE	Allows basic checking of remote server functionality.
VTXKILL.EXE	Allows the VTXNETD.EXE process to be stopped in a clean and efficient manner.
VTXAPI.EXE	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 4.1 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:NATOSQE511.EXE" NATOSQE
```

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, Windows Me, Windows XP, 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, Windows Me, Windows XP, 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, Windows NT, Windows Me, Windows XP

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 7.3.4	ORA7:
ORACLE 8i (8.1.5 and above)	ORA8:
ORACLE 9i	ORA9:
SYBASE 11 DBLIB	SYBDB11:pubs2
SYBASE 11 CTLIB	SYBCT11:pubs2
SYBASE 12 CTLIB	SYBCT12:pubs2
INFORMIX 9.2	INF92:shand
DB2 v7.1	DB27:shand
ADABAS SQL Server 1.4.3	ESQ143:esq123
ADABAS D 11	ADA11:mydb
ADABAS D 12	ADA12:mydb
ADABAS D 12 ODBC	ODBCAAD:nodename:mydb
INTERSOLV ODBC	ODBCINT:mysn

Notes:

- 1 *The driver names are case sensitive. For example, the name of the ORACLE 8i RDBMS driver is ORA8.so. It uses ORA8 in the connect string and is built from the file ora8.o. ORA8 is an executable that is used by the RDBMS server program “serversingle” (dispatcher) to load the ORA8.so shareable library.*
- 2 *The individual shareable library RDBMS drivers for local access are created via the osxlibs.sh utility. If you build an ORACLE 8i RDBMS driver, for example, you will see an ora8.o object and an ORA8 executable prior to running the osxlibs.sh script. After running the osxlibs.sh script, the ORA8.so shareable library exists, and the ORA8: connect string states that the ORA8 executable should be loaded. The ORA8 executable then loads the ORA8.so shareable library.*

Remote Client Connect Strings

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

Syntax for Remote Client Connect Strings

For remote access to a Windows 2000, Windows NT, or Windows XP 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	Client Connect String
ORACLE 7.3.4	NET:@7898:myhp!ORA7
ORACLE 8i (8.1.5 and above)	NET:@6060:mysun!ORA8
ORACLE 9i	NET:@5555:myaix!ORA9
SYBASE 11 DBLIB	NET:pubs2@1234:mydec!SYBDB11
SYBASE 11 CTLIB	NET:pubs2@2345:mysco!SYBCT11
SYBASE 12 CTLIB	NET:pubs2@3456:mylinux!SYBCT12
INFORMIX 9.2	NET:shand@9899:mywin!INF92
DB2 v7.1	NET:shand@1960:myredhat!DB27
ADABAS SQL Server 1.4.3	NET:esq123@9876:mysinix!ESQ143
ADABAS D 11	NET:mydb@7788:myx86!ADA11
ADABAS D 12	NET:mydb@5566:mysol!ADA12
ADABAS D 12 ODBC	NET:nodename:mydb@1984:nodename!ODB - CAAD
INTERSOLV ODBC	NET:mysn@2002:mynode!ODBCINT

Note:

The individual shareable library RDBMS drivers for remote access are created via the make utility, which uses the Makefile by default. If you build a SYBASE 12 CTLIB RDBMS driver, for example, you will see a sybct12.o object and a SYBCT12 executable prior to running the make utility. After running the make utility, the shareable library SYBCT12.so exists. The NET:pubs2@3456:mylinux!SYBCT12 connect string notifies the remote serversingle dispatcher that the SYBCT12 executable should be loaded. The SYBCT12 executable then loads the SYBCT12.so shareable library

Remote Access Commands

The following table lists make commands used to generate the RDBMS drivers for remote access:

Command	Resultant RDBMS Driver
\$ make oracle7	ORA7.so
\$ make oracle8	ORA8.so
\$ make oracle9	ORA9.so
\$ make sybase11dblib	SYBDB11.so
\$ make sybase11ctlib	SYBCT11.so
\$ make sybase12ctlib	SYBCT12.so
\$ make informix92	INF92.so
\$ make db27	DB27.so
\$ make esq143	ESQ143.so
\$ make adabas11	ADA11.so
\$ make adabas12	ADA12.so
\$ make adabas12odbc	ODBCAAD.so
\$ make intersolvodbc	ODBCINT.so

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 98, Windows 2000, Windows NT, Windows Me, or Windows XP 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 5.1 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 86) 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
```

Microsoft Windows Clients

The supported Microsoft Windows client platforms are Windows 98, Windows 2000, Windows NT, Windows Me, and Windows XP.

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, Windows Me, Windows XP 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 76.

You cannot delete a row read with a FIND SORTED BY statement. For an explanation, see the FIND statement description on page 76.

You cannot delete a row read with a READ LOGICAL statement. For an explanation, see the READ statement description on page 77.

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 76.

You cannot update a row read with a READ LOGICAL statement. For an explanation, see the READ statement description on page 77.

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] [(r)]
```

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 79).



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:

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

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 5.1 fields of type A. The option is obsolete if Natural 5.1 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 5.1 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



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

Note:

OpenIngres is no longer supported.

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 86 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 5.1)
- OS_USERID (with Natural 5.1)
- 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 5.1 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
- ORACLE
- SYBASE and Microsoft SQL Server

Notes:

1. *Format/lengths in the tables apply to Natural version 4.1 and above.*
2. *OpenIngres is no longer supported.*

The following tables provide the format and length for data accessed using DDMs created with the Natural DDM editor. See the Natural 5.1 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 101 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

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 5.1 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 5.1 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

Note:

OpenIngres is no longer supported.

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*



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

Note:

OpenIngres is no longer supported

```
[ODBC Data Sources]
```

```
myds=
```

```
Oracle7=
```

```
dBase=
```

```
Sybase=
```

```
Informix=
```

```
DB2=
```

```
Text=
```

```
[dBase]
```

```
QEWSD=35691
```

```
Driver=/opt/odbc/lib/ivdbf12.sl
```

```
Description=dBase
```

```
Database=/opt/odbc/demo
```

```
[Sybase]
```

```
QEWSD=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]
QEWSID=35698
Driver=/opt/odbc/lib/ivinf12.sl
Description=informix92
Database=osqdev
HostName=usrshp2
LogonID=informix
Password=mypwd

[DB2]
Driver=/opt/odbc/lib/ivdb212.sl
Description=DB2
Database=waHgpo

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

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

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

[Infodbc]
QEWSID=35698
Driver=/opt/odbc/lib/ivinf12.sl
Description=informix92
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=mypwd
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:
    /users/root:/usr/lib:/usr/lib/acct:/usr/local/bin:./SAG/a
    da/v22313:/SAG/ada/v22313/tools:/SAG/nat/v41219/bin:/SAG/w
    cp/v2117:/SAG/wcp/v2117/bin:/SAG/osx/v511/vtx:/SAG/aad/v12
    12/bin:/SAG/esq/v143/tools:/SAG/esq/v143/bin:./SAG/wcp/
    ushpux1:/SAG/demoweb/sh:/RDBMS/oracle/app/oracle/orahome/p
    roduct/9.1/bin:/RDBMS/sybase/bin:/RDBMS/sybase/install:/
    RDBMS/sybase/include:/opt/odbc/lib:/opt/odbc/bin:/RDBMS/sy
    base/lib:/RDBMS/sybase/bin:/usr/lib
SHLIB_PATH=/opt/odbc/lib:/RDBMS/sybase/lib:/RDBMS/sybase/bin:/u
    sr/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/v41219/bin:/SAG/wcp/v2117:/SAG/wcp/v2117/bin:/SAG/osx/v511/vtx:/SAG/aad/v1212/bin:/SAG/esq/v143/tools:/SAG/esq/v143/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
```



IBM DB2 CLI

The current direction chosen for DB2 is to use the IBM DB2 CLI (Call Level Interface).

The following is a sample db2cli.ini file:

```
[sample]
AUTOCOMMIT=0

[tstcli2x]
; Assuming dbalias2 is a database in DB2 for MVS.
SchemaList="OWNER1', 'OWNER2', CURRENT SQLID"

[MyVeryLongDBALIASName]
dbalias=dbalias3
SysSchema=MYSHEMA
```

Take special note of the **sample** database or alias name shown in the db2cli.ini file.

If you encounter the following error, you will need to add the AUTOCOMMIT=0 specification to the entries for your specific RDBMS.

```
[IBM] [CLI Driver] [DB2/SUN64] SQL0508N
The cursor specified in the UPDATE or DELETE statement is not positioned on a
row. SQLSTATE=24504.
```



APPENDIX C — SERVER TRACES

There are potentially four types of traces that can be generated when using Natural with Entire Access for TCP/IP.

Natural Client Trace

Natural specifically can invoke a trace as follows:

```
$ SQL_TRACE=3  
$ export SQL_TRACE
```

This trace may often be located in the \$NATDIR/\$NATVERS/tmp directory and would use your “login id” + trace iteration number. For example: JOHNDOE001.TRC

Note:

This trace is for Natural, not for Entire Access for TCP/IP. It might occasionally be requested by Software AG Customer Support. However, the standard trace is the Entire Access for TCP/IP client trace.

Entire Access for TCP/IP Client Trace

Entire Access for TCP/IP client can invoke a trace as follows:

```
$ VORTEX_API_LOGFILE=$OSXDIR/$OSXVERS/trace  
$ VORTEX_API_LOGOPTS=FULL  
$ export VORTEX_API_LOGFILE VORTEX_API_LOGOPTS
```

Note:

*This is the **standard** Entire Access for TCP/IP trace that may be requested by Software AG Customer Support.*



Entire Access for TCP/IP Server Trace

Entire Access for TCP/IP server can invoke a trace as follows:

```
$ VORTEX_HOST_LOGFILE=$OSXDIR/$OSXVERS/htrace  
$ VORTEX_HOST_LOGOPTS=FULL  
$ export VORTEX_HOST_LOGFILE VORTEX_HOST_LOGOPTS
```

Note:

*This is **not** the standard Entire Access for TCP/IP trace. This trace is only requested by Software AG Development under selective conditions.*

ODBC or CLI Trace

This trace is not documented here and would only be requested by Software AG Customer Support on behalf of Software AG Development.



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 11.



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 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 5.1.1 are selected:

```
INSTALL: ENVIRONMENT
```

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

```
1      ada/v31124
2      nat/v511
3      osx/v511
```

```
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 D 12.0.

```

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

  1 - remote ENTIRE ACCESS NET          8 - local ORACLE 7.3.4
  2 - local ADABAS SQL SERVER 1.4.3     9 - local ORACLE 8i (805+)
  3 - local ADABAS D 11.0              10 - local ORACLE 9i
* 4 - local ADABAS D 12.0              11 - local SYBASE 11 DBLIB
  5 - local ADABAS D 12.0 ODBC         12 - local SYBASE 11 CTLIB
  6 - local DB2 v7                    13 - local SYBASE 12 CTLIB
  7 - local INFORMIX 9.2              14 - 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 chosen to build the following environments

```
- local adabas d version 12.0
```

```
$OSXDIR    = /RDBMS/osxsag/sag/osx
$OSXVERS   = v511
$DBROOT    = /RDBMS/osxsag/sag/aad/v1200
```

```
ld -o ADA12.so ada12.o -G -B symbolic osx.a /RDBMS/osxsag/sag/aad/v1200/lib/
libs qlrte.a /RDBMS/osxsag/sag/aad/v1200/lib/pclib.a /RDBMS/osxsag/sag/aad/
v1200/lib/pcd2lib.a /RDBMS/osxsag/sag/aad/v1200/lib/pcd3lib.a /RDBMS/osxsag/sag/
aad/v1200/lib/libsqlrte.a /RDBMS/osxsag/sag/aad/v1200/lib/pcdlib.a /RDBMS/osx-
sag/sag/aad/v1200/lib/libsqlptc.a -lnsl -lsocket -lnsl -lsocket
```

Press <enter> to see the file '/RDBMS/osxsag/sag/osx/v511/osxlibs.lst'

```
/RDBMS/osxsag/sag/osx/v511/osxlibs.lst:
```

```
/RDBMS/osxsag/sag/osx/v511/bin/OSXAPI.so
/RDBMS/osxsag/sag/osx/v511/bin/osx.a
-lnsl
-lsocket
```

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**.

INDEX

A

Adabas, open/close processing, OPRB profile parameter, 72
API, for database access, 5, 7
Application environment supported, 39

B

BACKOUT TRANSACTION statement, 74

C

Client-Server, processing, 3
Close, database, OPRB profile parameter, 72
CMOSQERR program, for trouble shooting, 104

D

Data
 access
 local, 6
 remote
 using TCP/IP, 7
 using third-party network products, 8
 conversion, 94
 format, 94
 source
 connection, 54, 55
 definition, 54, 55
 type, 94
 conversion
 Adabas D, 95
 Adabas SQL Server, 96
 DB2, 97

 INFORMIX, 98
 Microsoft SQL Server, 100
 ORACLE, 99
 SYBASE, 100
Data source definition, OpenVMS, 56
Data sources, accessing, 6
Database
 assignment, 53
 authentication
 OpenVMS, 70
 UNIX, 68
 Windows, 69
 driver, 5
 local, 5
 name
 for local connect string, 57
 for localconnect string, 57
 for remote connect string, 59
 open/close processing, OPRB profile parameter, 72
 password
 OpenVMS, 70
 UNIX, 68
 Windows, 69
 remote, 5
 user ID
 OpenVMS, 70
 UNIX, 68
 Windows, 69
DB2, library links, db2l command, 14
DBMS
 assignment, 54, 55
 parameter, 54, 55
 type, 54, 55
DBMS assignment, OpenVMS, 56
DDM
 generation, 63
 generator, creation of (UNIX), 21, 35, 119

Entire Access for TCP/IP Manual for UNIX

DDMs, creating, 63
DELETE statement
 positioned, 74
 searched, 81
Directory structure, 49
Disk space requirement, 40
Driver name, for remote connect string, 59

E

END TRANSACTION statement, 75
Environment file, under UNIX, 17, 31, 116
Environment variables for RDBMSs, 15

F

FIND statement, 76

G

GRPNAM privilege, 52

H

HISTOGRAM statement, 77
Host name, for remote connect string, 59

I

INFORMIX, special considerations, 14
INSERT statement, 81
Installation
 of client component, 37

prerequisites
 for client component
 under Windows 2000, 37
 under Windows 98, 37
 under Windows NT, 37
 under Windows XP, 37
 under UNIX, 11, 29
procedure, with TCP/IP, under UNIX, 17, 31
procedure under UNIX, alternate, 116
under UNIX, with TCP/IP, 11, 29
under Windows 2000, prerequisites for client component, 37
under Windows 98, prerequisites for client component, 37
under Windows Me, prerequisites for client component, 37
under Windows NT, prerequisites for client component, 37
under Windows XP, prerequisites for client component, 37
Installation example, 42
Installation Kit, Structure, 41
Installation prerequisites, SAGBASE, 39
Installation procedure, 46

J

JTQUOTA, for an update installation, 46

L

Local, data access, 6
LOGIN.COM, 51

M

Microsoft SQL Server, restrictions and limitations, 91

N

NATPARM utility, for data source definition, 53
 NATPARM utility, accessing global configuration file, 56
 Natural
 configuration file, under UNIX, 53
 DDMs, 63
 environment, building of (UNIX), 21, 35, 119
 for
 UNIX, 11, 29, 30
 Windows 98, 38
 Windows NT and Windows 2000, 38
 logical name requirement, 52
 NATPARM utility, 53
 relinking, on UNIX client machines, 36, 120
 restrictions
 with Microsoft SQL Server, 92
 with SYBASE, 92
 statements
 DML, 5, 73
 SQL, 5, 81
 Natural for OpenVMS Version 2.2.3.26, 39
 Natural Security, relinking Natural, on UNIX
 client machines, 36, 120
 Natural SYSTRANS utility, 63

O

Open/close processing, OPRB profile parameter,
 72
 Operating system requirements, 39
 OPRB profile parameter, 72
 ORACLE, special considerations, 14
 OSX directories, 50
 OSX231 save sets, 41

P

PRCLM parameter setting, 40
 Prerequisites, installation, 40
 Process privileges, for Alpha AXP servers, 40
 PROCESS SQL statement, 82

R

READ statement, 77
 Remote, data access
 using TCP/IP, 7
 using third-party network products, 8
 Remote access, using TCP/IP, 38
 Requirements, operating system, 39

S

SAGBASE, 39
 Save sets, 41
 SELECT SINGLE statement, 88
 SELECT statement
 cursor-oriented, 88
 non-cursor , 88
 Server
 database driver, remote access (UNIX), 22
 number, for remote connect string, 59
 Shell script, for database drivers (UNIX), 19, 33,
 118
 SQL
 flexible, 89
 statements, 81
 DELETE, 81
 INSERT, 81
 PROCESS SQL, 82
 SELECT, 88
 SELECT SINGLE, 88
 UPDATE, 89

Entire Access for TCP/IP Manual for UNIX

Statements

DML

- BACKOUT TRANSACTION, 74
- DELETE, 74
- END TRANSACTION, 75
- FIND, 76
- HISTOGRAM, 77
- READ, 77
- STORE, 78
- UPDATE, 79

SQL

- DELETE, 81
- INSERT, 81
- PROCESS SQL, 82
- SELECT, 88
- SELECT SINGLE, 88
- UPDATE, 89

STORE statement, 78

Supported, databases

- Adabas C, 3
- Adabas D, 3
- Adabas SQL Server, 3
- DB2 for AIX, 3
- DB2 for mainframe, 3
- INFORMIX-IDS, 3
- ODBC-compliant databases, 3
- ORACLE, 3
- SYBASE CTLIB, 3
- SYBASE DBLIB, 3

SYBASE, restrictions and limitations, 91

SYSNAM privilege, 52

System Logical Names, created in START-
UP_OSX.COM, 50

T

Tape, installation, 40

TCP/IP

- for remote access, 38
- for UNIX, 12, 30

TCP/IP (UCX), connectivity, 39

Top level directory, 50

Trace log files directory, 50

TVER.EXE, 51

U

UNIX

- environment file, 17, 31, 116

- installation procedure, 116

- installation under, procedure, 17, 31

UPDATE statement

- searched, 89

- with FIND/READ, positioned, 79

- with SELECT, searched, 80

V

Version directory, 50

VMSINSTAL, used to install Entire Access for
TCP/IP, 46

VTXAPI.EXE, 51

VTXKILL.EXE, 51

VTXPING.EXE, 51

W

WSDEFAULT parameter setting, 40

Entire Access for TCP/IP Manual for UNIX

Notes

Entire Access for TCP/IP Manual for UNIX