# Entire Access for TCP/IP

# Manual for Windows 2000 and Windows 2003 Servers

**Manual Order Number: OSX511-030WIN**

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

# TABLE OF CONTENTS

# PREFACE

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

- *Entire Access for TCP/IP Manual for OS/390*
- *Entire Access for TCP/IP Manual for UNIX*
- *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 Windows, UNIX, and OpenVMS client platforms to access data sources on Windows, UNIX, AXP OpenVMS, and OS390 or z/OS server platforms.

*Note:*
*UNIX includes Redhat Linux INTEL, SuSE Linux INTEL, and Linux S/390.*

Entire Access for TCP/IP supports the following Windows platforms:

*Note:*
*All server platforms can also function as client platforms.*

| Windows Client-Only Platforms | Windows Server / Client Platforms |
|---|---|
| Windows 2000 Professional | Windows 2000 Server Series (Windows 2000 Server, Windows 2000 Advanced Server) |
| Windows XP Professional | Windows 2003 Server Series |

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

| Application Type | Client Platform | Server Data Source | Requirements |
|---|---|---|---|
| Natural | All supported Windows platforms<br>AXP OpenVMS<br>UNIX | All supported RDBMS | Entire Access for TCP/IP |
| Bolero and Java | All supported Windows platforms<br>UNIX | All supported RDBMS | Entire Access for TCP/IP and Entire Access for JDBC |
| ODBC-compliant | All supported Windows platforms | 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 | OS390 | Windows 2000 and 2003 Server Series | UNIX | 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 Online | | 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 and 2003 Server Series 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 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 Windows server (Windows 2000 or 2003 Server Series).

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

- Windows 2000 Professional, Windows XP Professional (chapter 3).
- UNIX (chapter 4).
- AXP OpenVMS (chapter 5).

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

Chapter 7 describes Entire Access for TCP/IP start-up procedures.

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

Appendix A describes trace variables for Natural, Entire Access for TCP/IP, and ODBC.

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

Appendix C provides supplemental information about configuring Entire Access for TCP/IP as a Windows 2000 and 2003 Server Series service or user utility.

Appendix D provides an alternate method for installing the Entire Access client component on UNIX in case the procedure presented in chapter 4 is inappropriate for your site.

1

# 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.

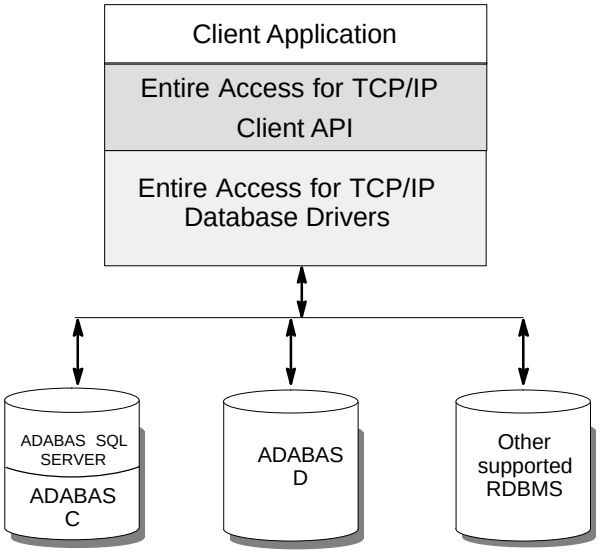
## Accessing Data Sources from Clients

The same procedure is used to access data sources from all client platforms: Windows 2000 Professional, Windows XP Professional, Windows 2000 and 2003 Server Series, UNIX, 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:

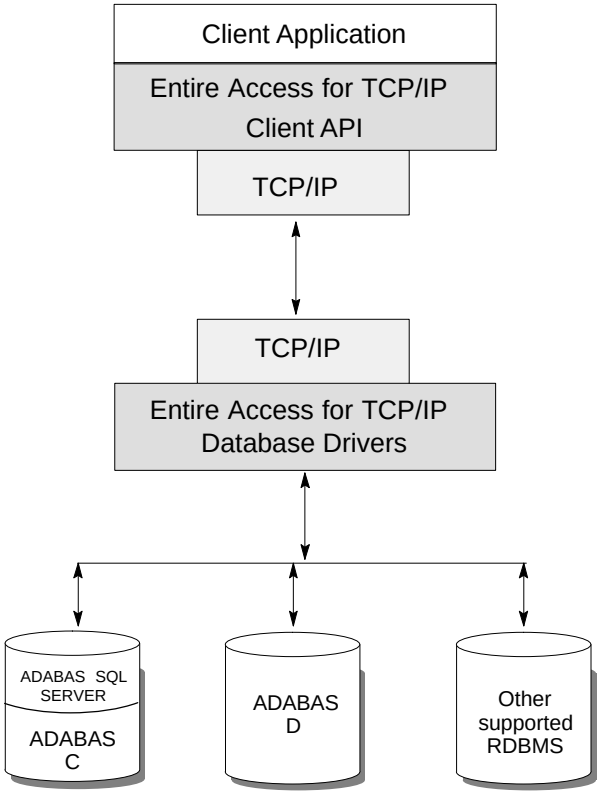



*Note:*
*Local access to data sources on the OS/390 and z/OS platform is supported only for Bolero application clients using Entire Access for JDBC for OS/390 and z/OS. Natural application clients are not supported under OS/390 and z/OS.*

# Remote Data Access Using TCP/IP

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

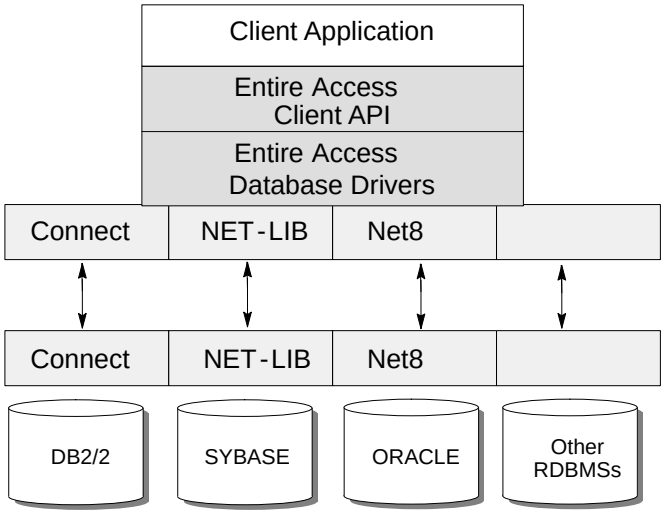
The following diagram shows remote access using TCP/IP:



# Remote Data Access Using Third-Party Network Products

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

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




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

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

2

# INSTALLING A SERVER

This chapter tells you how to install Entire Access for TCP/IP on the supported Windows server platforms:

- Windows 2000 Server Series (Windows 2000 Server and Windows 2000 Advanced Server)
- Windows 2003 Server Series

## Before You Install

### Client and Server Version Levels

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

- Version 5.1 on the server can communicate with version 4.2 on the client.
- Version 5.1 on the client can communicate with version 4.2 on the server.

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

### Server Hardware and Operating System Requirements

The following are the minimum hardware and operating system requirements for Entire Access for TCP/IP servers:

- an Intel 80586 CPU with a speed of 300 MHz
- 64 MB of RAM
- about 4 MB of available hard-disk space (if all options are installed)
- operating system for the server

## Other Software Requirements

- Natural version 5.1 or above is required for Natural clients.
- TCP/IP is required on both client and server machines for remote access.

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.

## Database Systems and Versions Overview

Entire Access for TCP/IP version 4.2 supports the data sources indicated in the following table. Except where noted, access can be either local or remote.

| Client Platforms | Database Servers on the Windows 2000 or Windows 2003 Servers |
|---|---|
| All supported Windows platforms (see page 1)<br>UNIX<br>AXP OpenVMS | Adabas SQL Server 1.4.4<br>Adabas D 12.0 and 13.0<br>DB2 7 and 8<br>INFORMIX-OnLine 9.2 and above<br>ORACLE 7.3.4<br>ORACLE 8i (all ORACLE 8.1 versions beginning with 8.1.5)<br>ORACLE 9i<br>SYBASE 11.9 and 12.0 CTLIB<br>SYBASE 11.1 and 11.9 DBLIB<br>ODBC-compliant<br>Microsoft SQL Server 6.5, 7.0, and 2000 |

## Adabas D

There are two Adabas D DLL drivers for Windows platforms:

- ADA12.DLL for Adabas D 12
- ADA13.DLL reserved for future use

## DB2

There are three DB2 DLL drivers for Windows platforms:

- DB26.DLL for DB2 v6
- DB27.DLL for DB2 v7
- DB28.DLL for DB2 v8

## Adabas C SQL Server (ESQ)

There is one ESQ DLL driver for Windows platforms:

- ESQ144.DLL for ESQ 1.4.4

## INFORMIX

There is one INFORMIX DLL drivers for Windows platforms:

- INF92.DLL for INFORMIX 9.2

Use the INFORMIX utility SETNET32.EXE to set the following INFORMIX variables for Entire Access: INFORMIXDIR, INFORMIXSERVER, INFORMIXSQLHOSTS.

If a conflict occurs with Natural shared memory, adjust the INFORMIX SHMBASE by changing the ONCONFIG.servername SHMBASE from ‘SHMBASE 0XC000000L’ to ‘SHMBASE 0X20000000L’.

## ORACLE

There are nine ORACLE DLL drivers for Windows platforms:

- ORA73.DLL for ORACLE 7.3.4
- ORA87.DLL for ORACLE 8 using version 7 features
- ORA97.DLL for ORACLE 9 using version 7 features
- ORA815.DLL for ORACLE 8.1.5
- ORA816.DLL for ORACLE 8.1.6
- ORA817.DLL for ORACLE 8.1.7
- ORA901.DLL for ORACLE 9.0.1
- ORA8.DLL for ORACLE 8 LOB support
- ORA9.DLL for ORACLE 9 LOB support

## SYBASE

If the environment variable DSQUERY is set, this value is used as the name of the SYBASE server. If DSQUERY is not set, the default “SYBASE” is used. Consult your SYBASE documentation for information about this variable.

Unless you use the default server name (“SYBASE”), you must either define the server name in the INTERFACES file or pass the name in the connect string.

### SYBASE CTLIB DLL Drivers

There are two SYBASE CTLIB DLL drivers for Windows platforms:

- SYBCT119.DLL for SYBASE 11.9 CTLIB
- SYBCT12.DLL for SYBASE 12.0 CTLIB

### SYBASE DBLIB DLL Drivers

There are two SYBASE DBLIB DLL drivers for Windows platforms:

- SYBDB111.DLL for SYBASE 11.1 DBLIB
- SYBDB119.DLL for SYBASE 11.9 DBLIB

*Note:*
*Software AG recommends that new customers use SYBASE CTLIB, not SYBASE DBLIB, as their default driver choice.*

## Microsoft SQL Server

There are three Microsoft SQL Server DLL drivers for Windows platforms:

- SQL65.DLL for MS SQL Server 6.5
- SQL7.DLL for MS SQL Server 7.0
- SQL2K.DLL for MS SQL Server 2000

## ODBC

There are two ODBC (Open Database Connectivity) DLL drivers for Windows platforms:

- VTX11.DLL for local ODBC access
- ODBCNET.DLL for remote ODBC access

This implies that the actual ODBC driver from the vendor (Intersolv, for example) will be installed and that the data source will be defined using the Microsoft ODBC Administrator.

# Installation Procedure

## Installation CDs

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

- The server CD contains the server component for Windows 2000 and 2003 Server Series.
- The client CD contains the Windows 2000 Professional and Windows XP Professional client components only.

*Note:*
*Servers can also be clients, but clients cannot be servers.*

## Installation Steps

Use the following steps to install Entire Access for TCP/IP on the Windows server platform:

### Step 1 Start the Setup Program

1. Insert the server CD into a CD drive.
2. To start the installation program, invoke the CD autorun.inf file. It begins the installation program automatically when the CD drive is closed. Or, to start the installation program manually, issue the following command at the command prompt, where "x" is the selected CD drive:

   ***x*:setup**

   The InstallShield Wizard is invoked to perform the installation tasks. The Natural Entire Access for TCP/IP 5.1.1 "Welcome" message appears when the load is complete.
3. Follow the directions concerning the License Agreement.
4. Enter your customer information, including your selection regarding Administrator rights. Be sure to read the statement regarding Administrator privileges.
4. Select the appropriate Destination Folder designations.
5. Choose Next to proceed with the installation (or Cancel to terminate the setup program).

## Step 2 Review settings and copy files

1. Review the settings before copying the files.
2. Review the README file as necessary. (You may choose to review it at a later time.)
3. Decide whether you will restart your computer now or later. Restarting now is recommended. Entire Access for TCP/IP depends on both Registry updates and Environment Variable updates that need to be established without any impedence for future product usage.

## Step 3 Installation is completed

1. After your computer is restarted, the InstallShield Wizard continues. Although an InstallShield window appears, it does not take long to complete, depending on your processor speed and whether other applications are also being restart.
2. Your Entire Access for TCP/IP v5.1.1 installation is now ready to function as:
   - a local RDBMS server for a remote Natural application,
   - a local RDBMS server for a local Natural application,
   - a host for a local client application that will connect to a remote RDBMS elsewhere in the network.

   The Entire Access for TCP/IP Client API and Server components contain the files listed in the following tables:

**Database DLL Drivers**

AAD12.DLL
AAD12.EXE
AAD13.DLL
AAD13.EXE
DB26.DLL
DB26.EXE
DB27.DLL
DB28.EXE
DB28.DLL
DB28.EXE
ESQ144.DLL
ESQ144.EXE
INF92.DLL
INF92.EXE
ODBCNET.EXE
ODBCNET.EXE
ORA73.DLL
ORA73.EXE
ORA8.DLL
ORA8.EXE
ORA815.DLL
ORA815.EXE
ORA816.DLL
ORA816.EXE
ORA817.DLL
ORA817.EXE
ORA87.DLL
ORA87.EXE
ORA9.DLL
ORA9.EXE
ORA901.DLL
ORA901.EXE
ORA97.DLL
ORA97.EXE
SQL2K.DLL
SQL2K.EXE
SQL65.DLL
SQL65.EXE
SQL7.DLL
SQL7.EXE
SYBCT119.DLL
SYBCT119.EXE
SYBCT12.DLL
SYBCT12.EXE
SYBDB111.DLL
SYBDB111.EXE
SYBDB119.DLL
SYBDB119.EXE

**Core API DLLs**

VTX3.DLL
VTX3.EXE
VTX11.EXE
VTX11.DLL
VTXAPI32.DLL
VTXAPI.DLL
VTXAPITS.DLL

**Server Procedures**

wmultiping.bat
wmultistart.bat
wmultistop.bat
wntping.bat
wntstart.bat
wntstop.bat
serverkill.exe
servermulti.exe
serverping.exe
serversingle.exe
w2000ping.bat
w2000start.bat
w2000stop.bat
zremoteping.bat
zremotestop.bat

**Service Procedure**

OSXSRV.EXE

**Trouble Shooting Routines**

YCHECK.EXE

## INFORMIX

If the %INFORMIXSERVER% variable is not set for your environment, error -25560 will be issued when you attempt to connect to the database server.

To set the %INFORMIXSERVER% variable:

1. Launch Setnet32 from the %INFORMIXDIR%\bin directory and select the Environment tab.
2. Find INFORMIXSERVER in the list of variables and select it. Type the server name into the entry box and then click the Set button.
3. Click the Apply button.

INFORMIX SHMBASE problems that are related to other products can often be resolved by adjusting the SHMBASE for INFORMIX, as follows:

1. Assume that your INFORMIX Server name as defined in the INFORMIXSERVER variable is “inf92”.
2. Copy and then edit the %INFORMIXDIR%\etc\ONCONFIG.inf92 and change the SHMBASE from:

```
SHMBASE         0xC000000L    #    Shared memory base address
```

to:

```
SHMBASE         0x2000000L    #    Shared memory base address
```

## Adabas SQL Server (ESQ)

Entire Net-Work 2.4.1 contains the required CSCI components but does not contain the required Entire Broker components. Support for the Broker stubs is provided by Software AG’s EntireX Broker product. For more information, refer to the EntireX product documentation.

3

# INSTALLING A CLIENT ON WINDOWS

This chapter tells you how to install Entire Access for TCP/IP on the supported Windows client platforms:

- Windows 2000 Professional
- Windows XP Professional

## Before You Install

### Client and Server Version Levels

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

- Version 5.1 on the server can communicate with version 4.2 on the client.
- Version 5.1 on the client can communicate with version 4.2 on the server.

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

### Client Hardware and Operating System Requirements

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

- an Intel 80586 CPU with a speed of 300 MHz
- 64 MB of RAM
- about 4 MB of available hard-disk space (if all options are installed)
- operating system for the client

## 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 server component for Windows 2000 and 2003 Server Series.
- The client CD contains the Windows 2000 Professional, and Windows XP Professional client components only.

*Note:*
*Servers can also be clients, but clients cannot be servers.*

## Installation Steps

To install the Entire Access for TCP/IP client component under Windows 2000 Professional or Windows XP Professional, 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 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.*

4

# INSTALLING A CLIENT ON UNIX

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

An alternate installation procedure is provided, beginning on page 103. 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

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

- Version 5.1 on the server can communicate with version 4.2 on the client.
- Version 5.1 on the client can communicate with version 4.2 on the server.

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

## Hardware and Operating System Requirements

The following tables show the minimum operating system versions and hardware requirements for selected 32-bit and 64-bit UNIX platforms supported by Entire Access for TCP/IP. For a complete list of supported platforms, contact your Software AG representative.

| 32-bit Platform / Operating System | Hardware Requirements |
|---|---|
| AIX 4.3.3 32-bit | IBM Series RS/6000 |
| AIX 5.1L 32-bit and above | IBM Series RS/6000 |
| HP-UX 11.0 32-bit | HP9000 series S700 or S800 |
| Linux SuSE 7.2 32-bit and above | Intel |
| Linux REDHAT 7.2 32-bit and above | Intel |
| Linux SuSE SLES-8 32-bit S/390 and above | Intel |
| SCO UnixWare 7.01 32-bit | SCO-certified Intel |
| Solaris 7, 8, and 9 32-bit | Sun Ultra SPARC |

| 64-bit Platform / Operating System | Hardware Requirements |
|---|---|
| AIX 5.1L 64-bit and above | IBM Series RS/6000 |
| Tru64 4.0F | Compaq ALPHA |
| Tru64 5.0 and above | Compaq ALPHA |
| HP-UX 11.0 and 11i 64-bit | HP9000 series S700 or S800 |
| Solaris 8 and 9 64-bit | Sun Ultra SPARC |

## Other Software Requirements

Entire Access for TCP/IP version 5.1 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 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 remote Entire Access Net.

   *Note:*
   *Only “remote Entire Access Net” is required on a UNIX client in order to access a Windows 2000 or Windows NT server.*

```
                ENTIRE ACCESS for TCP/IP (for 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                10  -  local ORACLE 9i
   4  -  local ADABAS D 12                11  -  local SYBASE 11 DBLIB
   5  -  local ADABAS D 12 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 selected the following parts to build Natural

   -   remote ENTIRE ACCESS NET


$OSXDIR    =  /usr/SAG/osx
$OSXVERS   =  v511



Press <enter> to see the file ' /usr/SAG/osx/v511/osxlibs.lst
```

5. 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.

   *Note:*
   *Repeat usage of “make” and “makefiles” is no longer necessary after the first “make” command when building a Natural nucleus. Instead of linking the RDBMS drivers into each Natural linking operation, Entire Access 5.1.1 uses two shareable libraries, OSXAPI.so and osx.a.*

## 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**

5

# INSTALLING A CLIENT UNDER AXP OPENVMS

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

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

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

## Before You Install

### Operating System Requirements

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

### TCP/IP Connectivity

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

### Supported Application Environment

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

Refer to the Software AG Natural documentation for more information.

### SAGBASE Requirement

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

# The Installation CD

The installation CD contains the Entire Access for TCP/IP save sets, which are extracted using the standard OpenVMS VMSINSTAL procedure. Save sets are described in the section **Installation Kit Structure** on page 31.

# Installation Prerequisites

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

## Disk Space

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

### PRCLM Parameter

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

### WSDEFAULT Parameter

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

### Process Privileges

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

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

# Installation Kit Structure

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

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

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

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

# Example of an Initial Installation

```
$ set proc/priv=all

$ @sys$update:vmsinstal

        OpenVMS AXP Software Product Installation Procedure V7.1

It is 9-AUG-2000 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 SYS$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 STARTUP_OSX.COM file.

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

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

### OSX$VERSION

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

### OSX$LOG

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

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

### OSX$VTX

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

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

## Step 3 Perform Startup and Verification

### Startup

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

#### Logical Names

The logical names that are used in the Entire Access environment and their definitions for version 4.1 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:NATOSQE411.EXE" NATOSQE
```

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

6

# 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 Windows platforms (Windows 2000 Professional and Windows XP Professional), UNIX, and AXP OpenVMS.

## 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.

# Windows Platforms

## 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 **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.

# 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 **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.

# 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 **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 Windows 2000 or 2003 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.<br>*Note:*<br>*For most ODBC connections, use the data source name instead of the data base name.* |

## Sample Local Client Connect Strings

The following table lists data sources and corresponding connect strings:

| Data Source | Client Connect String |
|---|---|
| ADABAS D 12 | AAD12:DATABASE |
| ADABAS D 13 | AAD13:DATABASE (reserved for future use) |
| DB2 6 | DB26:SAMPLE |
| DB2 7 | DB27:SAMPLE |
| DB2 8 | DB28:SAMPLE |
| ESQ 1.4.4 | ESQ144:ESQSAS |
| INFORMIX 9.2 | INF92:stores9 |
| MS SQL Server 6.5 | SQL65:pubs |
| MSSQL Server 7.0 | SQL7:pubs |
| MS SQL Server 2000 | SQL2K:pubs |
| ORACLE 7.3.4 | ORA73: |
| ORACLE 8 (version 7) | ORA87: |
| ORACLE 9 (version 7) | ORA97: |
| ORACLE 8.1.5 | ORA815: |
| ORACLE 8.1.6 | ORA816: |
| ORACLE 8.1.7 | ORA817: |
| ORACLE 9.0.1 | ORA901: |
| ORACLE 8 LOBs | ORA8: |
| ORACLE 9 LOBs | ORA9: |
| SYBASE 11.9 CTLIB | SYBCT119:pubs2 |
| SYBASE 12.0 CTLIB | SYBCT12:pubs2 |
| SYBASE 11.1 DBLIB | SYBDB111:pubs2 |
| SYBASE 11.9 DBLIB | SYBDB119:pubs2 |
| ODBC | ODBC:datasource |

# Remote Client Connect Strings

The remote connect string is the same for all client platforms: Windows XP Professional, Windows 2000 Professional, UNIX, and AXP OpenVMS.

*Note:*
*All server platforms can also function as client platforms.*

## Syntax for Remote Client Connect Strings

For remote access to a Windows server (Windows 2000 or 2003 Server Series), 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 (the Windows 2000 or Windows 2003 DLL) to be used. |

### Default Server Numbers

The following table contains the default server number for each platform:

*Note:*
*The Entire Access for TCP/IP Server Number correlates directly with a TCP/IP Socket Port Number.*

| Platform | Server Number |
|---|---|
| Windows 2000 servers | 2000 |
| Entire Access for TCP/IP Service server (installed automatically) | 2001 |
| Windows 2000 or Windows 2003 multithreaded servers (support for multithreaded servers is limited; the standard Windows 2000 or Windows 2003 server may be required). | 2022 |

## Sample Remote Client Connect Strings

The following table lists data sources and corresponding connect strings:

| Data Source | Client Connect String |
|---|---|
| ADABAS D 12 | NET:DATABASE@2050:dbhost!AAD12 |
| ADABAS D 12 UNIX ODBC | NET:nodename:mydb@7898:dbhost!ODBCAAD |
| ADABAS D 12 WINDOWS ODBC | NET:nodename:mydb@2050:dbhost!ODBCNET |
| ADABAS D 13 | NET:DATABASE@2050:dbhost!AAD13 (reserved for future use) |
| DB2 6 | NET:SAMPLE@2050:dbhost!DB26 |
| DB2 7 | NET:SAMPLE@2050:dbhost!DB27 |
| DB2 8 | NET:SAMPLE@2050:dbhost!DB28 |
| ESQ 1.4.4 | NET:ESQSAS@2050:dbhost!ESQ144 |
| INFORMIX 9.2 | NET:stores9@2050:dbhost!INF92 |
| MS SQL Server 6.5 | NET:pubs@2050:dbhost!SQL65 |
| MS SQL Server 7.0 | NET:pubs@2050:dbhost!SQL7 |
| MS SQL Server 2000 | NET:pubs@2050:dbhost!SQL2K |
| ORACLE 7.3.4 | NET:@2050:dbhost!ORA73 |

| Data Source | Client Connect String |
|---|---|
| ORACLE 8 (version 7) | NET:@2050:dbhost!ORA87 |
| ORACLE 9 (version 7) | NET:@2050:dbhost!ORA97 |
| ORACLE 8.1.5 | NET:@2050:dbhost!ORA815 |
| ORACLE 8.1.6 | NET:@2050:dbhost!ORA816 |
| ORACLE 8.1.7 | NET:@2050:dbhost!ORA817 |
| ORACLE 9.0.1 | NET:@2050:dbhost!ORA901 |
| ORACLE 8 LOBs | NET:@2050:dbhost!ORA8 |
| ORACLE 9 LOBs | NET:@2050:dbhost!ORA9 |
| SYBASE 11.9 CTLIB | NET:pubs2@2050!SYBCT119 |
| SYBASE 12.0 CTLIB | NET:pubs2@2050!SYBCT12 |
| SYBASE 11.1 DBLIB | NET:pubs2@2050!SYBDB111 |
| SYBASE 11.9 DBLIB | NET:pubs2@2050!SYBDB119 |
| UNIX ODBC Generic | NET:datasource@7898:dbhost!ODBCINT |
| WINDOWS ODBC Generic | NET:datasource@2050:dbhost!ODBCNET |

7

# START-UP PROCEDURES

## Accessing Data Sources from Clients

The connect strings used to access data sources on Windows servers (Windows 2000 or 2003 Server Series) are the same for all client platforms: Windows 2000 Professional, Windows XP Professional, UNIX, and AXP OpenVMS.

Perform the steps below to access a **local** data source:

1. Start the database(s).
2. Start the client application.

Perform the steps below to access a **remote** data source:

1. Start the database(s) on the remote Windows server (Windows 2000 or 2003 Server Series).
2. Start the Windows server daemon.
3. Start the client application.

# Server Daemons

This section explains how to start the server daemons using the icons installed with Entire Access for TCP/IP or using the Windows 2000 or Windows 2003 Start menu.

For a "service" method of starting and stopping the server daemon, see Appendix B starting on page 101.

## Starting, Pinging, and Stopping the Server Daemon

To start, ping, or stop the Windows 2000 or Windows 2003 server daemon or the multithreaded server, double click on the appropriate icon (see the table below) or use the Start menu as follows:

**Start**
**Programs**
**Entire Access for TCP-IP 5.1.1**
**Start, Ping, or Stop, as appropriate**

*Note:*
*To select a server number other than the default value, you must edit the batch file.*

| Server Platform or Type | Default Server Number | Icons Created During Installation |
|---|---|---|
| Windows 2000 | 2000 | Start 2000 Server<br>Ping 2000 Server<br>Stop 2000 Server |
| Multithreaded | 2022 | Start Threaded Server<br>Ping Threaded Server<br>Stop Threaded Server |

*Notes:*

1. *Two additional icons are created during the installation procedure for pinging and stopping remote servers.*
2. *Support for multithreaded servers is limited; the standard Windows 2000 or Windows 2003 server may be required.*

# Entire Access for TCP/IP Service

The default Entire Access for TCP/IP service is installed on port 2001 during the installation procedure.

## Starting the Default Service

To start the default Entire Access for TCP/IP service:

1. Use the Start menu, as follows:

   **Start**
   **Settings**
   **Control Panel**
   **Services**
   **Entire Access for TCP/IP Boot Service**

2. Highlight the service.

3. Press the Start button to start the service on Socket Port number 2001.

## Stopping the Default Service

To stop the default Entire Access for TCP/IP service:

1. Use the Start menu, as follows:

   **Start**
   **Settings**
   **Control Panel**
   **Services**
   **Entire Access for TCP/IP Boot Service**

2. Highlight the service.

3. Press the Stop button.

4. Double click the Service Kill icon.

8

# SUPPLYING USER ID AND PASSWORD

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 80) makes it possible to specify user IDs and passwords dynamically so that you can access different databases within a single Natural session. User ID and password can be specified either before or after the Natural session starts.

# Windows Clients

This section applies to the Windows client platforms: Windows 2000 Professional and Windows XP Professional.

## 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*

# 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*

# 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"*

# Windows Server

Depending on how your Windows server (Windows 2000 or 2003 Server Series) utilizes Domain Name Services (DNS), operating system authentication may or may not be possible. The system administrator should proceed as follows:

1. Use a valid Windows 2000 or Windows 2003 user ID of eight characters or fewer.

2. Before starting serversingle, set the VORTEX_AUTH_DOMAIN=<domain_name_server> where <domain_name_server> is the Domain Name Server used by your Windows 2000 or Windows 2003 server.

   *Note:*
   *Natural 5.1 requires an SQL keyword instead of "OSQ".*

3. Enable access to the Domain Name Server for the connecting user by logging on as a batch job.

4. To enable the attribute "Log on as a batch job", check the option "Show Advanced User Rights".

5. Ensure that both the user and the Administrator are added to the Administrators and any Domain groups. For example:

   **Administrative Tools (Common)**
   **User Manager for Domains**
   **Policies**
   **User Rights . . .**

   **User Rights Policy**
   **Domain: OSXDEV**
   **Right: Log on as a batch job**
   **Grant To: Administrator**
   **Administrators**
   **Domain Admins**
   **Domain Users**
   **Everyone**
   **NETWORK**
   **your user-name**
   **SQLExecutiveCmdExec(SQLExecutiveCmdExec)**
   **Users**

   ✓ **Show Advanced User Rights <checked>**

**Administrative Tools (Common)**
**User Manager for Domains**
**Administrator <double click>**
**Groups**
**Member of:**
**Administrators**
**Domain Admins**
**Domain Guests**
**Domain Users**

**Administrative Tools (Common)**
**User Manager for Domains**
**your user-name <double click>**
**Groups**
**Member of:**
**Administrators**
**Domain Admins**
**Domain Guests**
**Domain Users**
**Users**

A

# APPENDIX A - USING TRACE VARIABLES

## Natural Trace Variables

Natural supports a client trace variable called SQL_TRACE. For example:

**SQL_TRACE=3**

## Entire Access for TCP/IP Trace Variables

Entire Access for TCP/IP supports two sets of trace variables, one for the client and one for the server.

Example client trace variables:

**VORTEX_API_LOGFILE=C:\trace.txt**
**VORTEX_API_LOGOPTS=FULL**

Example server trace variables:

**VORTEX_HOST_LOGFILE=C:\htrace.txt**
**VORTEX_HOST_LOGOPTS=FULL**

## ODBC Trace Variables

ODBC tracing is enabled using the ODBC Manager and should be set to something like the following:

**C:\odbctrace.txt**

B

# APPENDIX B - 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 5.1 under Windows 2000, 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 *cursor-name* |
| 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 *cursor-name* |
| WITH | WHERE |

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

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

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

## BACKOUT TRANSACTION

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

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

## DELETE

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

### Example:

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

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

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

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

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

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

## END TRANSACTION

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

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

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

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

# FIND

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

## Example:

Natural statements:

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

Equivalent SQL statement:

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

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

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

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

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

# HISTOGRAM

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

## Example:

Natural statements:

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

Equivalent SQL statements:

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

# READ

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

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

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

## Example 1:

Natural statements:

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

Equivalent SQL statement:

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

### Example 2:

Natural statements:

```
READ PERSONNEL PHYSICAL
OBTAIN NAME
```

Equivalent SQL statement:

```
SELECT NAME FROM PERSONNEL
```

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

## STORE

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

### Example:

Natural statement:

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

Equivalent SQL statement:

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

# UPDATE

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

## UPDATE with FIND/READ

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

## Example:

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

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

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

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

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

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

## UPDATE with SELECT

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

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

Natural rejects any other form of the SELECT statement used with the DML UPDATE statement. Natural translates the DML UPDATE statement into a non-cursor or "searched" SQL UPDATE statement, which means than 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 73).

# Natural SQL Statements

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

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

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

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

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

## DELETE

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

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

## INSERT

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

# PROCESS SQL

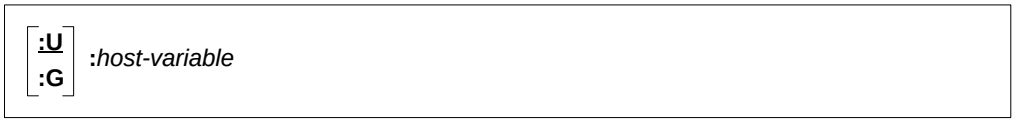
The PROCESS SQL statement issues SQL statements in a “statement-string” to the database identified by a **ddm-name**.

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

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

## Parameters

Natural version 5.1 supports the INDICATOR and LINDICATOR clauses. As an alternative, the statement-string may include parameters. The syntax item “parameter” is syntactically defined as follows:




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 |

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

**DBPROCESS**

**This option is valid for Sybase databases only.**

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

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

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

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

**TIMEOUT**

**This option is valid for Sybase databases only.**

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

**Example:**

To set the timeout period to 30 seconds, specify

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

**TRANSACTION**

**This option is valid for Sybase databases only.**

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

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

**Examples:**

To disable transaction mode, specify

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

To enable transaction mode, specify

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

## SQLDISCONNECT

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

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

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

## **SQLCONNECT** option = value

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

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

Supported options are:

- USERID
- PASSWORD
- OS_PASSWORD (with Natural 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:


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

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

## SELECT SINGLE

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

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

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

## UPDATE

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

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

# Flexible SQL

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

- atom
- column reference
- scalar expression
- condition

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

Note the following characteristics of flexible SQL:

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

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

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

## Examples:

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

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

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

# RDBMS-Specific Requirements and Restrictions

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

## Case-Sensitive Database Systems

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

*Note:*
*This restriction does not apply when you use flexible SQL.*

## SYBASE and Microsoft SQL Server

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

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

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

## How Natural Statements are Converted to Database Calls

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

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

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

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

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

## Natural Restrictions with SYBASE and Microsoft SQL Server

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

### Case-Sensitivity

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

### Positioned UPDATE and DELETE Statements

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

### Querying Rows

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

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

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

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

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

### Transaction/Non-Transaction Mode

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

### Stored Procedures

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

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

# Data Type Conversion

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

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

*Note:*
*Format/lengths in the tables apply to Natural version 5.1 and above.*

The following tables provide the format and length for data accessed using DDMs created with the Natural DDM editor. See the Natural 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 96 for more information.

# Adabas D

| RDBMS Data Type | Format/Length |
|---|---|
| boolean | L |
| char(*n*) | A*n* |
| date | A10 |
| long | A253 [1] |
| fixed(*n*) | N*n* |
| fixed(*y*,*x*) | N*x*-*y*.*y* |
| float(*n*) | N*n* [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(*n*) | A*n* |
| char(5) | -- |
| char(253) | -- |
| decimal(5) | I2 |
| decimal(10.4) | N6.4 |
| double precision | N10.6 |
| float(1...21) | N2.6 |
| float(22...53) | N10.6 |
| integer | I4 |
| numeric(5) | I2 |
| numeric(10.4) | N6.4 |
| real | F4 |
| smallint | I2 |

# DB2

| RDBMS Data Type | Format/Length |
|---|---|
| date | A10 |
| decimal(5) | I2 |
| decimal(10.4) | N6.4 |
| fixed character(5) | A5 |
| float | F8 |
| large integer | I4 |
| scientific notation | N10.6 |
| small integer | I2 |
| special data | A253 |
| system date and time | A10 |
| time | A10 |
| variable character (250) | A250 |

# INFORMIX

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

# OpenIngres

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

# ORACLE

| RDBMS Data Type | Format/Length |
|---|---|
| char(*n*) | A*n* |
| date | A14 |
| decimal | N29 * |
| float | F8 |
| integer | I4 |
| long | A253 |
| long raw(*n*) | B*n* |
| number | N29 * |
| number(*n*) | N*n* |
| number(*x*,*y*) | N*x*-*y*.*y* |
| raw(*n*) | B*n* |
| smallint | I4 |
| varchar(*n*) | A*n* |
| varchar2(*n*) | A*n* |

***** 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(*n*) | B*n* |
| bit | N1 |
| char(*n*) | A*n* |
| datetime | A26 |
| float | F8 |
| image | B126 |
| int | I4 |
| money | N15.4 |
| nchar(*n*) | A*n* |
| nvarchar(*n*) | A*n* |
| real | F4 |
| smalldatetime | A26 |
| smallint | I2 |
| smallmoney | N6.4 * |
| text | A253 |
| timestamp | B8 |
| tinyint | I2 |
| varbinary(*n*) | B*n* |
| varchar(*n*) | A*n* |

***** 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

## 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 | | YYYY-MM-DD-HH:II:SS* |
| (other formats are not supported) | | |

## OpenIngres

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

## ODBC

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

## ORACLE

| RDBMS Formats | Natural Date | Natural Time |
|---|---|---|
| DATE<br>(ORACLE session parameter NLS_DATE_FORMAT is set to YYYYMMDDHH24MISS) | YYYYMMDD000000<br>(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. |

C

# APPENDIX C - ENTIRE ACCESS FOR TCP/IP AS A SERVICE UNDER WINDOWS 2000 OR 2003

Entire Access for TCP/IP supports the Windows 2000 and Windows 2003 service control program with a single, executable program file:

**OSXSRV.EXE**

User logons and logoffs do not affect the execution of the program. The program is aware of Entire Access for TCP/IP through the Windows 2000 or Windows 2003 registry. You can define "instances" of Entire Access for TCP/IP, and the service will be able to start and stop these instances for you.

***Warning:***
*Although the registry can be modified, extreme care must be taken when doing so.*

## Defining the Configuration in the Registry

Before using the OSXSRV service, you can verify the configuration in the registry.

The Entire Access for TCP/IP installation program automatically adds the following **single entry** to the Windows 2000 or Windows 2003 registry:

```
HKEY_LOCAL_MACHINE
  SOFTWARE
    Software AG
     Entire Access for TCP/IP
       5.1.1
       Port2001
         SAGOSX=C:\Program Files\Software AG\
           Entire Access for TCP-IP\5.1.1\Bin\serversingle.exe -p2001
         START=NO
```

D

# 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.

The client and server version levels and the hardware and operating system requirements are described in the section **Before You Instal**l on page 21.

## 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 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

**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 remote Entire Access Net.

```
                ENTIRE ACCESS for TCP/IP (for 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                10  -  local ORACLE 9i
   4  -  local ADABAS D 12                11  -  local SYBASE 11 DBLIB
   5  -  local ADABAS D 12 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 selected the following parts to build Natural

   -   remote Entire Access NET


$OSXDIR    =  /usr/SAG/osx
$OSXVERS   =  v421


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

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

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

   The "ddmlibs.lst" file is used to create the DDM generator ("make ddmgen") in step 9.

   *Note:*
   *Repeat usage of "make" and "makefiles" is no longer necessary after the first "make" command when building a Natural nucleus. Instead of linking the RDBMS drivers into each Natural linking operation, Entire Access 5.1.1 uses two shareable libraries, OSXAPI.SO and OSX.A.*

## Step 4 Relink Natural on UNIX Client Machines

**This step is required for Natural 5.1.**

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 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**

# INDEX

## E



## F



## G



## H



## I



## J



## L



## M



## N

# O



# P



# R



# S

## T



## U



## V



## W