Entire Access

Entire Access for UNIX and Windows

Version 9.3.1

October 2023

ADABAS & NATURAL

This document applies to Entire Access Version 9.3.1 and all subsequent releases.

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

Document ID: OSX-DOC-931-20231008

Table of Contents

Preface

This documentation describes the installation and use of Entire Access on UNIX and Windows. It applies to all UNIX and Windows servers and clients supported. It is organized under the following headings:

What is New with this Version?	Provides an overview of new features.
How Entire Access Works	Describes how Entire Access works.
Prerequisite for Installing the Entire Access Server	Before you install the Entire Access server.
Installing a Server under UNIX	Describes how to install Entire Access on a UNIX server.
Installing a Server under Windows	Describes how to install Entire Access on a Windows server.
Installing a Client under UNIX	Describes how to install the Entire Access client component on UNIX.
Installing a Client under Windows	Describes how to install the Entire Access client component on Windows.
Defining Data Sources to Natural	Describes how to define data sources (servers) to Natural on the client machines.
Start-up Procedures under Windows	Describes start-up procedures under Windows.
Using Entire Access as Service under Windows	Describes how to configure Entire Access to be used as service under Windows.
Supplying User ID and Password	Contains information about supplying user IDs and passwords for database systems which require them.
Using Natural with Entire Access	Describes the special uses of Natural with Entire Access.
Traces for Error Diagnosis	Describes the traces available for error diagnosis.
Entire Access and SSL	Describes how to enable SSL with Entire Access.

For information on using Entire Access on other server platforms, see the Software AG documentation about Entire Access on z/OS.

1 About this Documentation

Document Conventions

Convention	Description
Bold	Identifies elements on a screen.
`Monospace font`	Identifies service names and locations in the format *`folder.subfolder.service`*, APIs, Java classes, methods, properties.
`Italic`	Identifies: Variables for which you must supply values specific to your own situation or environment. New terms the first time they occur in the text. References to other documentation sources.
`Monospace font`	Identifies: Text you must type in. Messages displayed by the system. Program code.
{ }	Indicates a set of choices from which you must choose one. Type only the information inside the curly braces. Do not type the { } symbols.
|	Separates two mutually exclusive choices in a syntax line. Type one of these choices. Do not type the | symbol.
[ ]	Indicates one or more options. Type only the information inside the square brackets. Do not type the [ ] symbols.
...	Indicates that you can type multiple options of the same type. Type only the information. Do not type the ellipsis (...).

Online Information and Support

Product Documentation

You can find the product documentation on our documentation website at **https://documentation.softwareag.com**.

In addition, you can also access the cloud product documentation via **https://www.softwareag.cloud**. Navigate to the desired product and then, depending on your solution, go to “Developer Center”, “User Center” or “Documentation”.

Product Training

You can find helpful product training material on our Learning Portal at **https://knowledge.softwareag.com**.

Tech Community

You can collaborate with Software AG experts on our Tech Community website at **https://tech-community.softwareag.com**. From here you can, for example:

- Browse through our vast knowledge base.
- Ask questions and find answers in our discussion forums.
- Get the latest Software AG news and announcements.
- Explore our communities.
- Go to our public GitHub and Docker repositories at **https://github.com/softwareag** and **https://hub.docker.com/publishers/softwareag** and discover additional Software AG resources.

Product Support

Support for Software AG products is provided to licensed customers via our Empower Portal at **https://empower.softwareag.com**. Many services on this portal require that you have an account. If you do not yet have one, you can request it at **https://empower.softwareag.com/register**. Once you have an account, you can, for example:

- Download products, updates and fixes.
- Search the Knowledge Center for technical information and tips.
- Subscribe to early warnings and critical alerts.
- Open and update support incidents.
- Add product feature requests.

Data Protection

Software AG products provide functionality with respect to processing of personal data according to the EU General Data Protection Regulation (GDPR). Where applicable, appropriate steps are documented in the respective administration documentation.

2 What is New with this Version?

This section covers the following topics:

Strategy Regarding the Legacy-Unix Platforms

We would like to inform you that, after a detailed analysis & assessment, Software AG has decided to adjust its strategy regarding the Legacy-Unix platforms HP-UX®, AIX® and Solaris®. With many of our customers already departed from or soon planning to depart their Legacy-Unix platforms due to cost and technical reasons, Software AG has decided Linux x86 will be its strategic open systems platform for Adabas & Natural 2050+ going forward. This will allow Software AG to focus more resources on this platform and maximize the overall value to our customer base.

The end-of-maintenance date (EOM) for Software AG support of the Legacy-Unix platforms is December 31, 2024. For the period from December 31, 2024 to December 31, 2025 Software AG will offer options for non-standard sustained support on the Legacy-Unix platforms for customers who are unable to rehost by the regular EOM date. Both dates apply to all Software AG A&N products (excluding CONNX, which will still be available on Legacy-Unix).

This will provide you with more than five (5) years to rehost your Software AG applications from the legacy-Unix platform to your preferred Linux x86 platform(s). Software AG recommends one of the following rehosting options:

- RedHat Enterprise Linux®
- SUSE Linux Enterprise
- CentOS

Please be assured that Software AG is prepared to offer assistance in planning and executing your rehosting from the Legacy-Unix platform to an alternative platform.

Following the principles of our "A&N 2050+ Initiative", your rehosting project will be a high priority to Software AG. Our local Software AG teams will be happy to discuss any rehosting topic with you.

If you have any questions regarding the Adabas & Natural platform roadmap, please do not hesitate to contact Adabas & Natural Product Management (e-mail: *AskANProdMgt@softwareag.com*).

For the Adabas & Natural products on the Legacy-Unix platforms HP-UX®, AIX® and Solaris® we currently plan the following final versions:

	Final Version (GA)	EOM	EOSS
Adabas HP-UX®	6.7.0, October 2018	31.12.2024	31.12.2025
Adabas AIX® and Solaris®	7.0, October 2020	31.12.2024	31.12.2025
Natural HP-UX®	9.1.1, October 2018	31.12.2024	31.12.2025
Natural AIX® and Solaris®	9.1.4, October 2021	31.12.2024	31.12.2025

Supported Operating Systems and Databases

The operating systems and database servers supported by Version 9.3.1 of Entire Access differ from those supported by previous versions.

- Supported operating systems: see the sections *Hardware and Operating-System Requirements* for **UNIX** and **Windows** respectively.
- Supported database servers: see the sections *Database Servers Supported* for **UNIX** and **Windows** respectively.

Database Drivers

Not every database driver is available on any platform. Especially for z/Linux there is currently only a subset of drivers available.

New and Enhanced Functionality

The following functionality is available with Entire Access Version 9.3.1:

MariaDB Support

This version of Entire Access supports the database system MariaDB. Natural 9.3.1 or above is necessary to access MariaDB.

OpenSSL 3.0 Support

This version of Entire Access supports OpenSSL 3.0 on Linux operating systems.

Dropped Functionality

Adabas D Support

This is the last version of Entire Access which supports the Adabas D database. The support will be dropped with the next version of Entire Access.

3 How Entire Access Works

This section covers the following topics:

What is Entire Access?

Entire Access allows client applications running on Windows and UNIX clients to access data sources on Windows and UNIX. The following 64-bit Windows platforms are supported for client and server:

- Windows 10
- Windows 11
- Windows 2016 Server
- Windows 2019 Server
- Windows 2022 Server

Entire Access under Windows is a 32-bit application and can only access database servers in 32-bit mode. In case the database server is a 64-bit application the 32-bit version of the corresponding database client interface must be installed also.

Data Sources

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

Server Data Source	UNIX	z/OS	Windows
Adabas D	x		x
Db2	x	x	x
Microsoft SQL Server			x
MySQL	x		x
Oracle	x		x
PostgreSQL	x		x
MariaDB	x		x

Entire Access 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 Windows and UNIX server platforms. Whether such access is possible depends on the database system. As a minimum, data sources to be accessed using the Entire Access ODBC driver must comply with the ODBC level 1 API and must accept the SQL core grammar.

General Information on Entire Access

Entire Access supports local and remote databases. It consists of an application program interface (API) and each supported RDBMS driver. 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 forms the backbone for RDBMS access.

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

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.

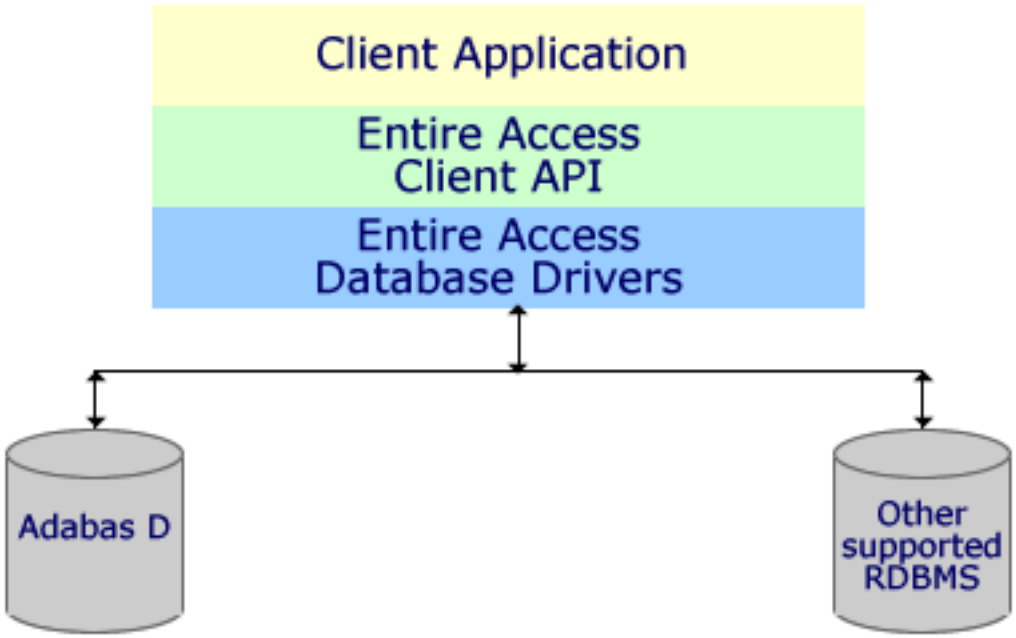
Accessing Data Sources from Clients

The same procedure is used to access data sources from all Windows and UNIX client platforms:

1. On the server machine, you start the database and then start Entire Access.
2. On the client machine, you set the connect string and then start the client application.

Local Data Access

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

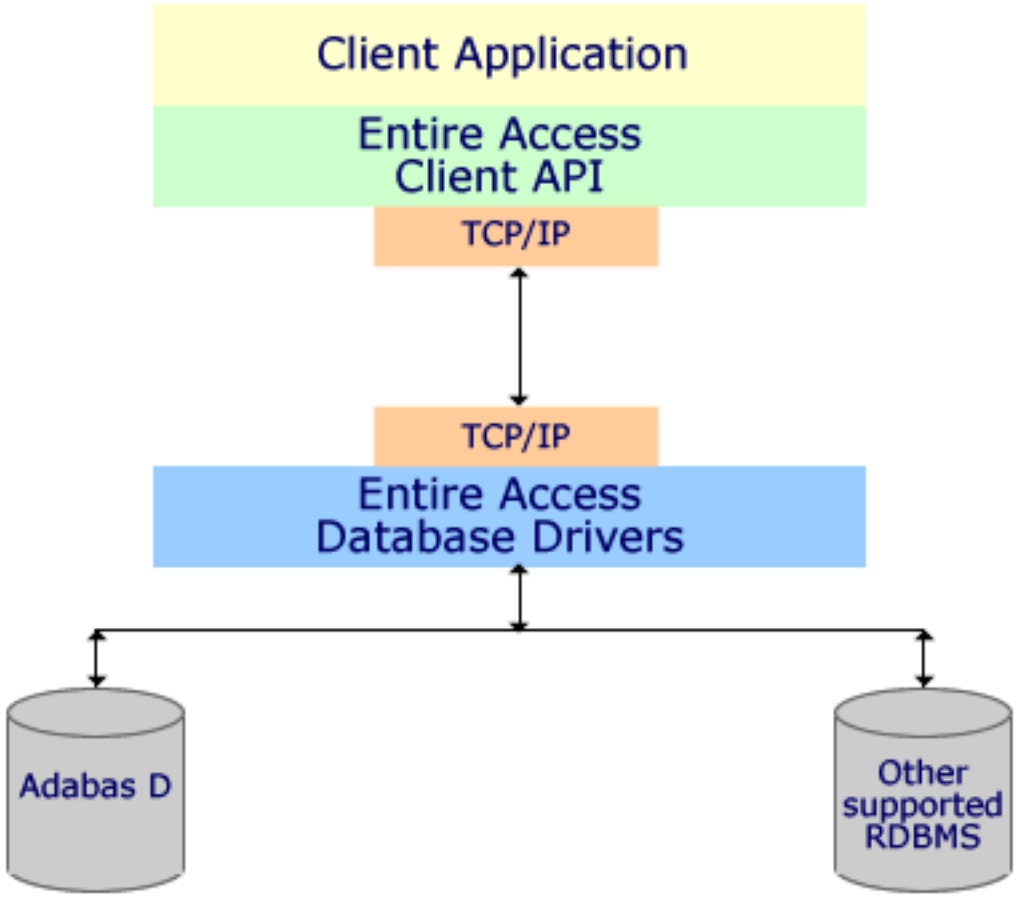


Note: Both local and remote access to data sources on the z/OS platform are supported by Entire Access. However, local Natural on z/OS does not utilize the local Entire Access API, and therefore z/OS may only be used as a RDBMS server.

Remote Data Access Using TCP/IP

With remote access, application clients communicate with Entire Access on the server side using the TCP/IP communications protocol. Entire Access 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.

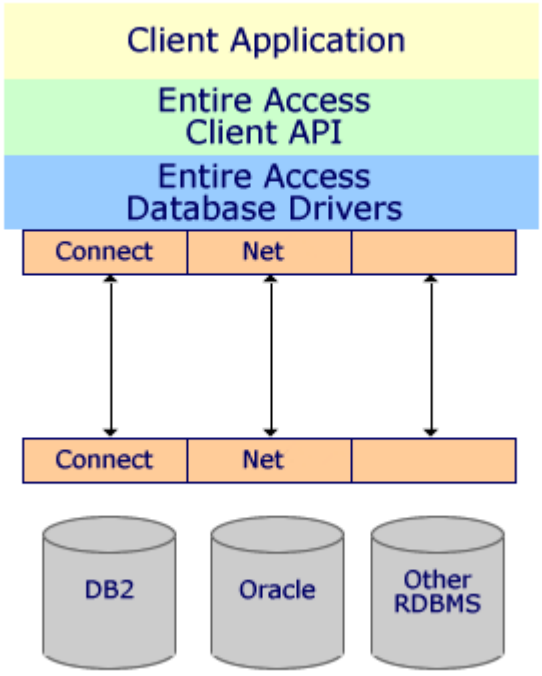
The following diagram shows remote access using TCP/IP:



Remote Data Access Using Third-Party Network Products

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

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



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

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

For example, if a local Windows 2016 Server PC is accessing a Db2 on VSE with IBM Connect networking, a local `DB2:databasename` connect string in Entire Access on the Windows 2016 Server PC is implied. Because Entire Access is completely unaware of the Windows 2016 Server to VSE rerouting, the VSE Db2 syntax must be identical to that of the Windows 2016 Server Db2, with no exceptions.

4 Prerequisite for Installing the Entire Access Server

Before you install the Entire Access server you need to install the Entire Access product.

This chapter describes the installation of Entire Access on Windows and Unix platforms.

Installing Entire Access Using the Software AG Installer GUI

To install Entire Access

1. Start the Software AG Installer GUI as described in *Using the Software AG Installer*.
2. When the first page of the Software AG Installer GUI (the so-called Welcome panel) is shown, choose **Next** repeatedly (and specify all required information on the shown panels as described in *Using the Software AG Installer*) until the panel containing the product selection tree appears. This tree lists the products you have licensed and which can be installed on the operating system of the machine on which you are installing.
3. To install Entire Access, expand the **Natural Products** node and select the Entire Access product node there. All required items are automatically selected.
4. Choose **Next**.
5. Read the license agreement, select the check box to agree to the terms of the license agreement, and choose **Next**.
6. On the last panel, review the items you have selected for installation. If the list is correct, choose **Install** to start the installation process.

Uninstalling Entire Access

You uninstall Entire Access using the Software AG Uninstaller. For information on how to use the uninstaller, see the *Using the Software AG Installer guide*.

5 Installing a Server under UNIX

This section describes how to install Entire Access on a UNIX server. It covers the following topics:

Before You Install

Please read the following information, before you proceed with the installation:

Client and Server Versions

This version of Entire Access is downward compatible to Entire Access Version 6, but not to any older version.

Hardware and Operating-System Requirements

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

Operating System	Hardware Requirement
SUSE Linux Enterprise Server 15	x86-64
Redhat Linux Advanced Server 8	x86-64
Redhat Linux Advanced Server 8	IBM z/Linux 64
Redhat Linux Advanced Server 9	x86-64

Other Software Requirements

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

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

For information about the use of third-party network products, see the section ***Remote Data Access Using Third-Party Network Products***.

UNIX Compiler Support

Entire Access supports the vendor compiler, including the Linker or Loader, for each supported UNIX platform.

Database Servers Supported

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

The following database server versions are supported:

- Adabas D 15
- Db2 10 and 11
- Oracle 12c, 19c, and 21c
- MySQL 8 (not on Linux IBM Z)
- MariaDB 10 and 11 (not on Linux IBM Z)
- PostgreSQL 14 and 15 (not on Linux IBM Z)
- ODBC-compliant servers (local access)

Oracle

With Oracle, problems may occur in conjunction with fields of types LONG and LONG RAW. These problems are due to the status of the OCI API in the 64-bit Oracle versions. Instead of LONG and LONG RAW, use types CLOB and BLOB respectively.

Database System Preparation

Before you begin to install Entire Access, perform the following steps:

- Install your DBMS software.
- Set the environment variables for the relevant RDBMS. See the table below for a list of the environment variables required for each RDBMS.
- Db2 users: Create the links for the Db2 libraries with the command `Db2ln`. See the *DATABASE 2 UNIX* installation guides for more information.

Environment Variables

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

RDBMS	Environment Variable	Build/Run Time
Adabas D	DBROOT	Build/Run
	DBNAME (optional) [1]	Run
	PATH=$PATH:DBROOT/bin	Run
Db2	DB2_HOME	Build
	DB2INSTANCE	Run
	PATH=$PATH:$DB2_HOME/sqllib/adm: $DB2_HOME/sqllib/bin: $DB2_HOME/sqllib/misc	Run
Oracle	ORACLE_HOME	Build/Run
	ORACLE_SID	Run
	TWO_TASK (optional) [3]	Run
MySQL	MYSQL_HOME	Build/Run
PostgreSQL	pg_config tool must be available	Build
MariaDB	MARIADB_HOME	Build/Run

[1] If DBNAME is set, it will override the database specified in the connect string (that is, Entire Access for TCP/IP will connect to the database specified in DBNAME and not to the database specified in the connect string); for further information on connect strings, see the section *Define the Data Sources*.

[3] TWO_TASK must be set to "P:" if the database instance to be accessed is on the local machine. If TWO_TASK is set, it will override the database instance specified in the ORACLE_SID environment variable.

Installation Procedure

Before you install the Entire Access servers you need to install the Entire Access product according to the installation description in *Prerequisite for Installing the Entire Access Server*. When the product has been successfully installed continue with the following steps.

Step 1 - Select the Local Database Drivers

Use the interactive `osxlibs.sh` script to select the database drivers(s) to be used by Entire Access. The selected database drivers for the local databases will be built; they can be used for local access, or for remote access via the Entire Access server (see Step 4).

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

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

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

```
$ osxlibs.sh
```

   A list of database drivers appears.

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

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

   Each selected entry is indicated by an asterisk (*) to the left of the number column. In the following example, the selected entry (*) is local Adabas D 15.

```
              ENTIRE ACCESS for TCP/IP X.X.X (Linux x86-64)
              ============================


    1  -  remote ENTIRE ACCESS NET
    2  -  local Oracle
    3  -  local Adabas D 15
    4  -  local DB2
    5  -  local DataDirect ODBC
    6  -  local MySQL
    7  -  local PostgreSQL
    8  -  local MariaDB

    g  -  Generate 'osxlibs.lst'          q  -  Exit

please select an entry:
```

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.

```
You have chosen to build the following environments

        - local Adabas D 15

$OSXDIR      = /FS/fsdb/products/osx
$OSXVERS     = .

$DBROOT      = /FS/fsdb/adabasd/aad/v1501

aad15
cpclnk tdb10_15 -o VTX10.so -shared tb2.a  -lcrypt -lnsl -lncurses
cc -o tdb10_15 -m64 tdb10_15.o -o VTX10.so -shared tb2.a -lcrypt -lnsl -lncurses
 /FS/fsdb/adabasd/aad/v1501/lib/pcrlib.a /FS/fsdb/adabasd/aad/v1501/lib/pcd1lib.
a /FS/fsdb/adabasd/aad/v1501/lib/pcd2lib.a /FS/fsdb/adabasd/aad/v1501/lib/pcd3li
b.a /FS/fsdb/adabasd/aad/v1501/lib/libsqlrte.a /FS/fsdb/adabasd/aad/v1501/lib/li
bsqlterm.a /FS/fsdb/adabasd/aad/v1501/lib/libsqlptc.a -lncurses -lm /FS/fsdb/ada
basd/aad/v1501/lib/sqlca.a
mv VTX10.so ../bin
mv ../bin/VTX10.so ../bin/AAD15.so
```

Note: Entire Access supports shareable libraries so that, for example, a generic Natural can be built from `OSXAPI.so`, and the target RDBMS for the client application is in fact determined by the connect string. This means that Natural can be built once for all RDBMS drivers without specifically loading any RDBMS driver code into the Natural nucleus. For example, from an Adabas D 15 object called `tds10 15.o` an object called `AAD15.so` will be created as the shareable library. AAD15 is the executable responsible for loading the `AAD15.so` shareable library.

5. When using shareable libraries, you may need to set an additional environment variable `LD_LIBRARY_PATH` that points to the directories containing the shareable libraries. Refer to your operating-system's instructions.
6. Verify that the environment variables are correct; then press ENTER to generate the *osxlibs.lst* file. The screen displays the contents of this file as it is being generated.

   The *osxlibs.lst* file contains a list of all database libraries to be linked to the Natural prelinked object `natraw.o`. The Natural *make* file uses *osxlibs.lst* when a new Natural environment is built with *make natural osx=yes ....*
7. The build process for local database drivers can also be started without using the `osxlibs.sh` script.

   Enter the `makedb` command with a valid database identifier to build the desired database driver:

```
makedb identifier
```

identifier	Database Driver
aad15	Adabas D 15
cli	DataDirect
db2	Db2
ora	Oracle
mysql	MySQL
pgrs	PostgreSQL
mariadb	MariaDB

Note: If necessary, modify the paths in the corresponding *makesrv.???* file in order to match your specific system requirements.

Step 2 - Start the Entire Access Server (for Remote Access)

1. Access the *bin* directory of the *osx* installation by entering the following command:

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

2. Make sure that the database environments for all RDBMS are loaded correctly.
3. Use the interactive `osxopr.sh` script to start a server dispatcher:

```
$ osxopr.sh
```

```
------------------------------------------------------------------
     ENTIRE ACCESS for TCP/IP
      ========================

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

        q  -  exit


please select an entry:
2
```

```
------------------------------------------------------------------
      ENTIRE ACCESS for TCP/IP (start a server)
      =========================


please enter the server number
8888


starting server with number 8888
```

```
------------------------------------------------------------------
      ENTIRE ACCESS for TCP/IP
      ========================

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

        q  -  exit


please select an entry:
1
```

```
------------------------------------------------------------------
        ENTIRE ACCESS for TCP/IP (show status)
        ======================


ENTIRE ACCESS for TCP/IP directory:   /usr/natdev/OSX_TEST2/osx
ENTIRE ACCESS for TCP/IP version:     X.X.X
Available server:          ADABAS D server, ORACLE server
Available utilities:       serversingle, serverping, serverkill,

press <return> to continue
```

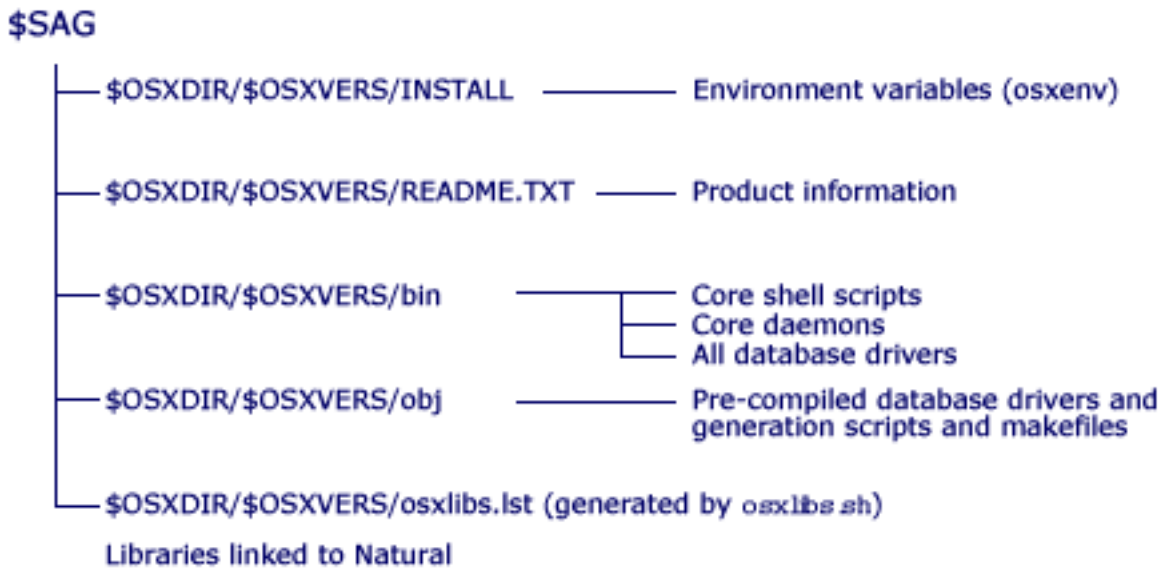
```
------------------------------------------------------------------
        ENTIRE ACCESS for TCP/IP (show status)
        ======================

Available Server Daemon ports:  8888

list of active client/server connections:

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

press <return> to continue
```

```
------------------------------------------------------------------
      ENTIRE ACCESS for TCP/IP
      ======================

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

        q  -  exit


please select an entry:
4
```

```
------------------------------------------------------------------
      ENTIRE ACCESS for TCP/IP (ping a server)
      ======================

please enter the server number:
8888



please enter the hostname (default: sunedb):

pinging server with number 8888 on sunedb ...

server #8888 is alive and kicking
```

Entire Access Directory Structure Under UNIX

The following is the Entire Access directory structure under UNIX:



6 Installing a Server under Windows

This section describes how to install Entire Access on a Windows server. It covers the following topics:

Supported Windows servers are:

- Windows 10
- Windows 11
- Windows 2016 Server
- Windows 2019 Server
- Windows 2022 Server

Except where indicated, the installation described in this section applies to all the above servers.

Before You Install

Please read the following information, before you proceed with the installation:

Client and Server Versions

See ***Client and Server Versions*** in the section *Installing a Server under UNIX*.

Hardware and Operating-System Requirements

The following are the minimum hardware and operating-system requirements for Entire Access servers:

- 512 MB of RAM,
- about 10 MB of available hard-disk space (if all options are installed).

Other Software Requirements

- Natural Version 8.4 or above is required for Natural clients.
- TCP/IP is required on both client and server machines for remote access.

For information about the use of third-party network products, see the section *Remote Data Access Using Third-Party Network Products*.

Database Servers Supported

Entire Access supports the data sources listed below. Except where noted, access can be either local or remote.

The following database server versions are supported on Windows servers:

- Adabas D 15
- Db2 10 and 11
- Oracle 12c, 19c, and 21c
- Microsoft SQL Server 2012, 2014, 2016, 2017, 2019, and 2022
- MySQL 8
- PostgreSQL 14 and 15
- MariaDB 10 and 11
- ODBC-compliant servers (local access)

The Entire Access server is available as a 32-bit and as a 64-bit application. Please note that database drivers for MySQL, MariaDB, or PostgreSQL are only available for 64-bit servers while drivers for Adabas D are only available for 32-bit servers.

Adabas D

There is one Adabas D DLL driver for Windows platforms:

- `AAD15.DLL` for Adabas D 15

Db2

There is one Db2 DLL driver for Windows platforms:

- `DB2.DLL` for all supported Db2 versions

Oracle

There is one Oracle DLL driver for Windows platforms:

- `ORA.DLL` for all supported Oracle versions

Microsoft SQL Server

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

- `MSSQLODBC.DLL` for MS SQL Server 2012, 2014, 2016, 2017, 2019, and 2022 using SQL Server's ODBC interface.

  The SQL Server database to be accessed has to be defined as a data source using the Microsoft ODBC Administrator. Natural 8.4 or above is a prerequisite for using the MSSQLODBC interface.
- `MSSQLODBCN.DLL` for MS SQL Server 2012, 2014, 2016, 2017, 2019, and 2022 using SQL Server's ODBC native client interface.

  The SQL Server database to be accessed has to be defined as a data source for the SQL Server native client interface using the Microsoft ODBC Administrator. Natural 8.4 or above is a prerequisite for using the MSSQLODBCN interface.

MySQL

There is one MySQL DLL driver for Windows platforms:

- `MYSQL.DLL` for all supported versions of MySQL

PostgreSQL

There is one PostgreSQL DLL driver for Windows platforms:

- `POSTGRESQL.DLL` for all supported versions of PostgreSQL

MariaDB

There is one MariaDB DLL driver for Windows platforms:

- `MARIADB.DLL` for all supported versions of MariaDB

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 (for example, DataDirect) will be installed and that the data source will be defined using the Microsoft ODBC Administrator.

Installation Procedure

To install the Entire Access product on your Windows machine please follow the installation instructions in *Prerequisite for Installing the Entire Access Server*.

Decide whether you will restart your computer now or later. Restarting now is recommended. Entire Access depends on Environment Variable updates that get established after restart.

After your computer is restarted, your Entire Access 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.

7 Installing a Client under UNIX

This section describes how to install the Entire Access client component under UNIX. It covers the following topics:

Before You Install

Please read the following information, before you proceed with the installation:

Client and Server Versions

See ***Client and Server Versions*** in the section *Installing a Server under UNIX*.

UNIX Compiler Support

Entire Access supports the vendor compiler, including the Linker or Loader, for each supported UNIX platform.

Hardware and Operating-System Requirements

The operating-system and hardware requirements are the same as described in **Hardware and Operating-System Requirements** in the section *Installing a Server under UNIX*.

Other Software Requirements

Entire Access requires TCP/IP on both the client and the server for remote server support.

In addition, Natural client applications require Natural Version 8.4 or above.

For the use of third-party network products, see also the section ***Remote Data Access Using Third-Party Network Products***.

Installation Procedure

Before you enable the Entire Access clients you need to install the Entire Access product according to the installation description in *Prerequisite for Installing the Entire Access Server*. When the product has been successfully installed continue with the following steps.

Perform the following steps to install the Entire Access client component on any supported UNIX platform.

Step 1 - Select the Database Drivers

This step is the same as for a server installation; see **Step 1** of the UNIX server installation procedure.

Step 2 - Relink Natural on UNIX Client Machines

The standard Natural image from the installation is already prelinked with all database drivers. In case you want to use this image you can skip this step. Otherwise, regenerate your Natural nucleus with the selected Entire Access database drivers.

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

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

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

   If you only require access to SQL databases with Natural, enter the command as follows:

   ```
   make natural osx=yes
   ```

   If you also require access to Adabas, enter the command as follows:

   ```
   make natural osx=yes ada=yes
   ```

   By default, Adabas is not included when Natural is relinked. If you do not specify a value for the `ada=` parameter, Adabas will not be linked into Natural and Natural Security will not function.

3. To copy this new Natural file into the *bin* directory, enter the following command:

   ```
   make install
   ```

8 Installing a Client under Windows

This section decribes how to install the Entire Access client component under Windows. It covers the following topics:

The client installation is not necessary if a complete server installation has been performed.

Before You Install

Please read the following information, before you proceed with the installation:

Client and Server Versions

See ***Client and Server Versions*** in the section *Installing a Server under UNIX*.

Client Hardware Requirements

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

- 512 MB of RAM,
- about 10 MB of available hard-disk space (if all options are installed).

Other Software Requirements for Clients

The following products and versions are required in order to use Entire Access:

- For Natural clients, Natural Version 8.4 is recommended.
- For remote access, TCP/IP is required on both the client machine and 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***.

Installation Procedure

Installation Prerequisites

Prerequisite is a full installation of Entire Access available on either a Unix or a Windows platform. A full installation of Entire Access includes an Entire Access Windows client.

Installation

Copy the file *OXCvnnnn.zip* from the directory <*EntireAccessInstallFolder*>/*OXC* of your Entire Access full installation to your target Windows machine. Extract the files from the zip archive into the /*BIN* directory where Natural has been installed.

9 Defining Data Sources to Natural

You have to define the data sources which your application programs are to access. This section describes how to define data sources to Natural clients. It covers the following topics:

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, see the installation instructions for Natural.

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

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

These steps are described below in detail for each client platform: UNIX and Windows.

Software AG recommends not to use a client machine as an RDBMS server.

UNIX Clients

Access the Natural Global Configuration File

To 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 Sources

The DBMS assignment includes the "connect string" that Entire Access uses to establish the connection with the data source.

Perform the following steps for each data source you wish 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 below.
4. In the **Modify/Delete** entry field, enter "M" (Modify) and press ENTER .

Save the Updated Natural Global Configuration File

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

Windows Clients

Access the Natural Global Configuration File

To access the Natural global configuration file:

1. Invoke the Natural Configuration utility either by double-clicking on the **Natural Configuration Utility** icon in the Natural Program Group, or by entering the command `natparm` at the command prompt.
2. Select **Natural Configuration Files > Global Configuration File > DBMS Assignments**.

Define the Data Sources

The global DBMS assignment includes the "connect string" that Entire Access uses to establish the connection with the data source.

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

1. In the **DBID** field, specify a unique database ID.
2. In the **DBMS Type** field, specify "SQL". Use this value for each data source.
3. In the **DBMS Parameter** field, specify a connect string as described in the following sections.

Save the Updated Natural Global Configuration File

When you have defined all the data sources, save the global configuration file.

Local Client Connect Strings

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

Syntax for Local Client Connect Strings

The syntax for a local database connect string is as follows: *dbms:db-name*.

dbms	Specifies the Entire Access 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, uses TNS names. For ODBC connections, use the data-source name instead of the database name.

Samples of Local Client Connect Strings for UNIX

The following table lists data sources and corresponding connect strings:

Data Source	Client Connect String
Adabas D 15	`AAD15:mydb`
DataDirect ODBC	`ODBCINT:mydsn`
Db2	`DB2:shand`
Oracle	`ORA:mytns`
MySQL	`MYSQL:mysqldb`
PostgreSQL	`POSTGRESQL:mypostdb`
MariaDB	`MARIADB:mymariadb`

Note: The driver names are case-sensitive. For example, the name of the Oracle RDBMS driver is *ORA.so*. It uses ORA in the connect string. ORA is an executable that is used by the RDBMS server program "serversingle" (dispatcher) to load the *ORA.so* shareable library. The individual shareable-library RDBMS drivers for local access are created via the `osxlibs.sh` utility. After running the `osxlibs.sh` script, the *ORA.so* shareable library exists, and the ORA: connect string states that the ORA executable should be loaded. The ORA executable then loads the *ORA.so* shareable library.

Samples of Local Client Connect Strings for Windows

The following table lists data sources and corresponding connect strings:

Data Source	Client Connect String
Adabas D 15	`AAD15:DATABASE`
Db2	`DB2:SAMPLE`
MS SQL Server 2012, 2014, 2016, 2017, 2019, 2022	`MSSQLODBC:datasource` (see Note 1 below)
MS SQL Server 2012, 2014, 2016. 2017, 2019, 2022	`MSSQLODBCN:datasource` (see Note 2 below)
Oracle	`ORA:mytns`
ODBC	`ODBC:datasource`

Notes:

1. If this connect string is used, `datasource` has to be defined as an ODBC data source using the Microsoft ODBC Administrator. Natural 8.4 or above is a prerequisite.
2. If this connect string is used, `datasource` has to be defined as SQL Server native client data source using the Microsoft ODBC Administrator. Natural 8.4 or above is a prerequisite.
3. MySQL, MariaDB, and PostgreSQL cannot be used with local client connect strings on Windows.

Remote Client Connect Strings

The remote connect string is the same for all client platforms.

Syntax for Remote Client Connect Strings

For remote access to UNIX, Windows and z/OS RDBMS servers, connect the Entire Access network component by specifying "NET":

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

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, uses TNS names.
server-number	Is a 4-digit number from 1025 to 9999 which identifies the server daemon; it must match the server number you specify when you start the server daemon. See also *Default Server Numbers for Windows* below.
host-name	Identifies the host machine on which the server runs. Enter either the name (as specified in the */etc/hosts* file) or the Internet address (in *nn.nn.nn.nn* format) of the host.

driver	Specifies the database driver to be used.

Default Server Numbers for Windows

The following table shows the default server number for each Windows platform:

Platform	Server Number
Windows 2000 servers	2000
Entire Access Service server	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

Note: The Entire Access server number correlates directly with a TCP/IP socket port number.

Samples of Remote Client Connect Strings for UNIX

The following table lists data sources and corresponding connect strings:

Data Source	Client Connect String
Adabas D 15	`NET:mydb@2002:myhp!AAD15`
DataDirect ODBC	`NET:mydsn@2004:mysun!ODBCINT`
Db2 9	`NET:sas@1994:mywin!DB2`
Db2 10	`NET:sas@2000:myredhat!DB2`
Oracle	`NET:mytns@2006:mylinux!ORA`
MySQL	`NET:mysqldb@2022:myhost!MYSQL`
PostgreSQL	`NET:mypostdb@2022:myhost!POSTGRESQL`
MariaDB	`NET:mymariadb@2022:myhost!MARIADB`

Samples of Remote Client Connect Strings for Windows

The following table lists data sources and corresponding connect strings:

Data Source	Client Connect String
Adabas D 15.0	`NET:DATABASE@2050:dbhost!AAD15`
Db2	`NET:SAMPLE@2050:dbhost!DB2`
MS SQL Server 2012, 2014, 2016, 2017, 2019, 2022	`NET:datasource@2050:dbhost!MSSQLODBC` (see Note 1 below)
MS SQL Server 2012, 2014, 2016, 2017, 2019, 2022	`NET:datasource@2050:dbhost!MSSQLODBCN` (see Note 2 below)

Data Source	Client Connect String
Oracle	`NET:mytns@2050:dbhost!ORA`
UNIX ODBC Generic	`NET:datasource@7898:dbhost!ODBCINT`
Windows ODBC Generic	`NET:datasource@2050:dbhost!ODBCNET`
MySQL	`NET:mysqldb@2022:myhost!MYSQL`
PostgreSQL	`NET:mypostdb@2022!myhost!POSTGRESQL`
MariaDB	`NET:mymariadb@2022!myhost!MARIADB`

Notes:

1. If this connect string is used, `datasource` has to be defined as an ODBC data source on `dbhost` using the Microsoft ODBC Administrator. Natural 8.4 or above is a prerequisite.
2. If this connect string is used, `datasource` has to be defined as an SQL Server native client data source on `dbhost` using the Microsoft ODBC Administrator. Natural 8.4 or above is a prerequisite.

10 Start-Up Procedures under Windows

This section covers the following topics:

Accessing Data Sources from Clients

The connect strings used to access data sources on a Windows server are the same for all UNIX and Windows client platforms.

To access a local data source:

1 Start the database(s).

2 Start the client application.

To access a remote data source:

1 Start the database(s) on the remote Windows server.

2 Start the Windows server daemon.

3 Start the client application.

Server Daemons

This section describes how to start the server daemons using the icons installed with Entire Access or the Windows Start menu.

To start, ping, or stop the Windows server daemon or the multithreaded server, double-click on the appropriate icon (see the table below).

Alternatively, from the **Start** menu, you can choose **Programs > Entire Access** *n.n.n*, and Start, Ping or Stop it.

Server Type	Default Server Number	Icons Created During Installation
Singlethreaded Server	2000	Start Singlethreaded Server Ping Singlethreaded Server Stop Singlethreaded Server
Multithreaded Server	2022	Start Multithreaded Server Ping Multithreaded Server Stop Multithreaded Server
64-bit Server	2064	Start 64-bit Server Ping 64-bit Server Stop 64-bit Server

To select a server number other than the default value, you have to edit the batch file.

Note: Two additional icons are created during the installation procedure for pinging and stopping remote servers.

11 Using Entire Access as Service under Windows

Entire Access can be configured as service so that the Entire Access server can be automatically started during system startup time. The registration of the Entire Access service to the Windows service control manager is already done during the installation of Entire Access.

This section covers the following topics:

Installing an Entire Access Service under Windows

The Entire Access service consists of two files which can be found in the Entire Access *bin* directory after the installation procedure has finished:

- *sagosx.exe*: This is the service itself.
- *sagosx.cfg*: This is the service configuration file.

First the service has to be registered to the Windows service control manager.

This is done by typing the command

```
sagosx -install
```

at any command prompt. This step has usually already been performed by the installation procedure.

The service can be de-registered again with the command

```
sagosx -remove
```

This is usually done during the de-installation of Entire Access.

Starting the Service

The service is started from the **Start** menu as follows:

Start > Settings > Control Panel > Administrative Tools > Services > Entire Access Service.

1. Highlight the Entire Access Service.
2. Press the **Start** button of the service.

During the starting process the Entire Access service is executing the commands from the *sagosx.cfg* configuration file. Per default the relevant part of the configuration file is as follows:

```
# Start listener on port 2001
serversingle.exe -p2001
```

This command would start a serversingle listener process on port 2001. Generally, lines in the configuration file can either be commands, environment variable settings or comments. Therefore, a more complex example for a configuration file could look like the following example:

```
# First set environment variables
ORACLE_HOME=d:\db\ora12
ORACLE_SID=mysid
# Start listener on port 4711
serversingle.exe -p4711
# Start second listener on port 4712
serversingle.exe -p4712
```

The Entire Access service always searches for the file *sagosx.cfg* in the same directory where the corresponding *sagosx.exe* file is located. However, it is possible to specify a Windows environment variable `ENTIRE_ACCESS_SERVICE_FILE` that points directly to the configuration file. When using an environment variable, the file can be located in any desired location; e.g. setting `ENTIRE_ACCESS_SERVICE_FILE=d:\mydir\myfile.cfg` would force the Entire Access service to read the configuration from the file *myfile.cfg* in the directory *d:\mydir*.

Starting the Service Automatically

The Entire Access service can be configured to start automatically during a system startup. This can be achieved using the **Start** menu as follows:

Start > Settings > Control Panel > Administrative Tools > Services > Entire Access Service.

1. To open the **Properties** box of the service, double-click the Entire Access Service.
2. Select the Startup type **Automatic** on the tab **General**.

Stopping the Service

The service is stopped from the **Start** menu as follows:

Start > Settings > Control Panel > Administrative Tools > Services > Entire Access Service.

1. Highlight the Entire Access Service.
2. Press the **Stop** button of the service.

When the Entire Access service is stopped, all previously started listener processes as specified in the configuration file will also be stopped.

12 Supplying User ID and Password

This section describes the use of Natural variables, which are not part of Entire Access. It covers the following topics:

General Authentication Information

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.

Otherwise, 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 can specify the desired type of authentication, using the environment variable `SQL_DATABASE_LOGIN`:

```
SQL_DATABASE_LOGIN=authentication-type
```

where *authentication-type* can be one of the following:

- `DB` = database authentication (the default),
- `OS` = operating-system authentication,
- `DB_OS` = both database and operating-system authentication.

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 the Natural *Statements* documentation) 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.

Operating-system authentication is only possible for remote connect strings.

Note: The authentication details can also be set by the Natural Configuration Utility (NATPARM). Here it is possible to make DBID-specific settings. See the Natural *Configuration Utility* documentation for further details.

UNIX Clients

Authentication Type

To set the authentication type, set the following environment variable in your login procedure:

```
SQL_DATABASE_LOGIN=authentication-type
export SQL_DATABASE_LOGIN
```

Database Authentication

For database authentication, set the following environment variables in your login procedure:

```
SQL_DATABASE_USER=db-user-id
SQL_DATABASE_PASSWORD=db-password
export SQL_DATABASE_USER SQL_DATABASE_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, set the following environment variables in your login procedure:

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

Windows Clients

Authentication Type

To set the authentication type, set the following environment variable:

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

Database Authentication

For database authentication, set the following environment variables:

```
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, set the following environment variables:

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

Windows Server

Depending on how your Windows server 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 user ID of 8 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 server.
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
```

13 Using Natural with Entire Access

This section covers the following topics:

Entire Access supports Natural SQL statements and most Natural DML statements. Natural DML and SQL statements can be used in the same Natural program. At compilation, if a DML statement references a DDM for a data source defined in `NATCONF.CFG` with DBMS type "SQL", Natural translates the DML statement into an SQL statement.

Natural converts DML and SQL statements into calls to Entire Access. Entire Access 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, see the Natural documentation.

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. This is accomplished by creating a Natural data definition module (DDM) from the table or view.

To generate DDMs from SQL tables or views, you use the Natural DDM editor.

For details about generating DDMs and the DDM editor, see the Natural documentation.

Setting Natural Profile Parameters

ETEOP Parameter

This parameter can be set only by Natural administrators.

The Natural profile parameter `ETEOP` controls transaction processing during a Natural session. It is required, for example, 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 Natural program.

If the `ETEOP` parameter is set to:

- `ON` - 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;
- `OFF` (the default) - Natural does *not* issue an `END TRANSACTION` command (that is, does not "commit") at the end of a Natural program. This setting thus enables a single logical transaction to span more than one Natural program.

Note: The ETEOP parameter applies to Natural Version 6.1 and above. With previous Natural versions, the Natural profile parameter OPRB has to be used instead of ETEOP (ETEOP=ON corresponds to OPRB=OFF, ETEOP=OFF corresponds to ORPB=NOOPEN).

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 does not support the following DML statements and options:

DML Statement
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

The following DML statements are covered in detail below:

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 below.

You cannot delete a row read with a FIND SORTED BY or READ LOGICAL statement. For an explanation, see the FIND and READ statement descriptions below.

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.

Note: Entire Access 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 SQL 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` or `READ LOGICAL` statement. For an explanation, see the `FIND` and `READ` statement descriptions above.

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* above).

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*.

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

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

For a detailed description of the common set of Natural SQL statements, see the Natural documentation. For information about the extended set, see the documentation of the Natural interface for your database system.

The extended set is supported by Natural on mainframe computers only.

The following SQL statements are covered in detail below:

- DELETE
- INSERT
- PROCESS SQL
- SELECT
- SELECT SINGLE
- UPDATE

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.

See the Natural documentation for more information.

Parameters

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

```
[ :U ]
[ :G ] :host-variable
```

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

SET SQLOPTION option = value

With Entire Access, 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`
- `EXIO_RESULT_SETS` (for Db2 only)
- `RAW_DATETIME`
- `DYNAMIC_CURSOR`

DATEFORMAT

This option specifies the format used to retrieve SQL Date and Datetime information into Natural fields of type A. The option is obsolete if Natural 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' >>
```

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 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
MariaDB	YYYY-MM-DD HH:II:SS
MySQL	YYYY-MM-DD HH:II:SS
ODBC	YYYY-MM-DD HH:II:SS
ORACLE	YYYYMMDDHHIISS
PostgreSQL	YYYY-MM-DD HH:II:SS
other	DD-MON-YY

EXIO_RESULT_SETS

This option is valid for Db2 databases only.

Db2 allows the definition of stored procedures that have output parameters specified and that return a result set at the same time. If such a stored procedure is to be called via the `CALLDBPROC` statement, then this functionality has to be enabled by setting `EXIO_RESULT_SETS` to `YES`.

Examples:

To enable output parameters and result sets at the same time in stored procedure calls, issue the following statement:

```
PROCESS SQL sql-ddm << SET SQLOPTION EXIO_RESULT_SETS = YES >>
```

To disable output parameters and result sets at the same time in stored procedure calls, issue the following statement (this is the default):

```
PROCESS SQL sql-ddm << SET SQLOPTION EXIO_RESULT_SETS = NO >>
```

Note: Only one result set can be returned. Do not use stored procedures that return more than one result set.

RAW_DATETIME

This option is available as of Natural Version 8.3.4. It is used to enable/disable the date/time conversion in Entire Access and Natural when date and time values are read from a database. It should only be used if date/time values are read into alphanumeric fields, but not if they are read into date/time fields.

If this option is set to `NO` (this is the default), Entire Access converts date and time values according to the given date/time format masks. If it is set to `YES`, date/time values are provided in the format which is enabled in the database.

Examples:

To enable date/time conversion, specify:

```
PROCESS SQL sql-ddm << SET SQLOPTION RAW_DATETIME = NO >>
```

To disable date/time conversion, specify:

```
PROCESS SQL sql-ddm << SET SQLOPTION RAW_DATETIME = YES >>
```

DYNAMIC_CURSOR

This option (available as of Natural version 9.1.4) is used to switch between a dynamic and a non-dynamic cursor type only for SQL Server databases. By default, this switch is set to YES so that only dynamic cursors are used for retrieving data. Only in situations where you experience a big loss of performance for certain SELECT statements you may try to switch the cursor to non-dynamic before that statement and back to dynamic after. It is not recommended to run the complete application with cursors switched to non-dynamic.

To enable dynamic cursors, specify:

```
PROCESS SQL sql-ddm << SET SQLOPTION DYNAMIC_CURSOR = YES >>
```

To disable dynamic cursors, specify:

```
PROCESS SQL sql-ddm << SET SQLOPTION DYNAMIC_CURSOR = NO >>
```

SQLDISCONNECT

With Entire Access, you can also specify `SQLDISCONNECT` as the statement-string. In combination with the `SQLCONNECT` statement (see 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, 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
- OS_USERID
- 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

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.

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

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

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

See the Natural documentation 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.

Case-Sensitive Database Systems

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

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

MySQL and MariaDB

MySQL and MariaDB do not implement updatable cursors. This means that the statements `UPDATE WHERE CURRENT OF cursor-name` and `DELETE WHERE CURRENT OF cursor-name` are not supported. However, when tables have a `PRIMARY KEY` or `UNIQUE NOT NULL` index that consists of a single column that has an integer type, Natural uses _rowid to simulate `WHERE CURRENT OF cursor-name` statements. Please refer to your MySQL or MariaDB documentation and especially the *Create Table* chapter for more information

Data-Type Conversion

When a Natural program accesses data in a relational database, Entire Access converts RDBMS-specific data types to Natural data formats, and vice versa. The RDBMS data types and their corresponding Natural data formats are described in the *Natural DDM Editor* documentation under *Data Conversion for RDBMS*.

The date/time or datetime format specific to a particular database can be converted into the Natural formats D and T; see the section *Date/Time Conversion* below.

Date/Time Conversion

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

To use this conversion, you first have to 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 must not be mixed with those converted to Natural A(lphanumeric) format.

For update commands, Natural 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 conversion tables below.

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

MariaDB

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

MySQL

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

ODBC

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

Oracle

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

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

PostgreSQL

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

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, using the following syntax:

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

The parameters are:

Parameter	Format/Length	Contents
parm1	I4	The number of the error returned by the database.
parm2	A255	The text of the error returned by the database.

14 Traces for Error Diagnosis

This section describes the traces available for error diagnosis when using Natural with Entire Access:

Use these traces only when Software AG Support requests you to do so.

Traces Under UNIX

Natural Client Trace

Natural can invoke a trace as follows:

```
$ SQL_TRACE=4
$ export SQL_TRACE
```

This trace is located in the Temporary Files Path directory. It uses your user ID and trace iteration number; for example: `JOHNDOE001.TRC`

The Natural client trace can also be enabled via the Natural Configuration Utility (Natural Version 6.2 or above).

Entire Access Client Trace

An Entire Access client can invoke a trace as follows:

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

Entire Access Server Trace

An Entire Access server can invoke a trace as follows:

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

Traces Under Windows

Natural Client Trace

Natural can invoke a trace as follows:

```
SQL_TRACE=4
```

This trace is located in the Temporary Files Path directory. It uses your user ID and trace iteration number; for example: JOHNDOE001.TRC

The Natural client trace can also be enabled via the Natural Configuration Utility (Natural Version 6.2 or above).

Entire Access Client Trace

An Entire Access client can invoke a trace as follows:

```
VORTEX_API_LOGFILE=file-name
VORTEX_API_LOGOPTS=FULL
```

Entire Access Server Trace

An Entire Access server can invoke a trace as follows:

```
VORTEX_HOST_LOGFILE=file-name
VORTEX_HOST_LOGOPTS=FULL
```

ODBC Trace

ODBC tracing is enabled using the ODBC Manager.

15 Entire Access and SSL

With Entire Access SSL can be used to secure the communication between the Entire Access client and the Entire Access server. In general, when an Entire Access server is enabled for SSL then the Entire Access client and the Entire Access server establish a secured communication channel and all data traffic is done encrypted on this channel. The most common Entire Access client in this context is Natural.

Prerequisites

When Entire Access in SSL mode is to be used, the following prerequisites must apply:

- On Linux platforms, OpenSSL 1.1.*x* or OpenSSL 3.0.*x* must be available on the client and on the server platform.
- On Windows platforms, the Windows built-in SChannel (Secure Channel) API is used for performing security related operations, so no other additional software is necessary.
- An Entire Access server must be available. Especially in cases where only an Entire Access client is used and where the network routing is done via database vendor specific software, the Entire Access SSL feature cannot be used.

Configure SSL

Configure SSL on Linux Platforms

When using the SSL mode of Entire Access on Linux platforms, an OpenSSL kit of version 1.1.*x* or 3.0.*x* must be available for the machines where the Entire Access server and the clients will run. The OpenSSL kit is not part of Entire Access and must be compiled and installed separately when required.

OpenSSL must be configured and compiled in shared mode, so that the `libssl.so` and `libcrypto.so` libraries are available for Entire Access during runtime and are present in the in the library search path.

Configure SSL on Windows Platforms

On Windows platforms, the built-in SChannel API is used for performing SSL related tasks, so that no additional software needs to be installed. SChannel will also interoperate with clients or servers on Linux platforms which use OpenSSL.

Certificates and Keys

To use SSL with Entire Access, a digital certificate and a private key are necessary. Please generate a certificate and a key in PEM format and put the certificate and the key into one single file which can then be used for starting the Entire Access server. For Entire Access servers on Windows platforms, the key must be an RSA key.

OpenSSL offers a command line utility `openssl` that is (among other functions) capable of generating self-signed certificates and private key files and can convert between the different formats. Refer to the **official OpenSSL** documentation for a detailed description.

Configure the Entire Access Server

To configure an Entire Access server in SSL mode, the following steps must be performed:

- Obtain a digital certificate and a private key in PEM format. Both are used by the Entire Access server to initiate a secured communication channel. Put the certificate and the private key into one single file, for instance `certificate_file`.
- Start the Entire Access server process `serversingle` with an additional parameter `-e certificate_file` which specifies the certificate file with full path. By providing this parameter the Entire Access server automatically starts in SSL mode and can only be connected by a client also capable of SSL mode.

  On Linux platforms the script *osxopr.sh* has been enhanced for specifying the certificate file parameter when starting a new Entire Access server.

  On Windows platforms the provided batch scripts and the service configuration file contain examples of starting a server in SSL mode.

Configure the Entire Access Client

To enable an Entire Access client, such as Natural, for connecting an Entire Access server in SSL mode, you must first do the following:

- On Linux platforms, make sure the two OpenSSL libraries mentioned above are available in the search path of the client.
- On Windows platforms, no additional software is necessary.

The Entire Access client automatically starts an encrypted communication with the Entire Access server.

Server Validation

In cases where it is required to enable the client to validate the server certificate before initiating the communication, the following steps must be perfomed:

1. Add a new environment variable `TRIM_HOME` pointing to the Entire Access installation base directory, which is *<InstallDir>/EntireAccess*. This variable is already pre-defined on Linux systems but must be added manually for Windows installations.
2. Edit the *net.ini* file delivered with each Entire Access installation inside the directory *${TRIM_HOME}/lib* (or *%TRIM_HOME%\lib* on Windows) and add a new line specifying the location of a certificate trust store file

   ssl_certfile <mypath>/<mycertstore>

   The certificate trust store file will be used to validate the server certificate. The certificates must be in PEM format and must not be pass-phrase protected. The following example specifies that the client will use a certificate trust store stored in the *crt.txt* file:

   ```
   ssl_certfile /etc/certs/crt.txt
   ```

Note: The server validation is only available with the full installation of the Entire Access product and not with the OXC client kit.