VORTEX

Installation and Operations Manual

December 28, 2021

Trifox Inc.
www.trifox.com

Trademarks

TRIMapp, TRImpl, TRIMqmr, TRIMreport, TRIMtools, GENESISsql, DesignVision, DVapp, DVreport, VORTEX, VORTEXcli, VORTEXc, VORTEXcobol, VORTEXperl, VORTEXjdbc, VORTEX++, VORTEXJava Edition, LIST Manager, VORTEXodbc, VORTEXnet, VORTEXclient/server, VORTEXaccelerator, VORTEXreplicator are all trademarks of Trifox, Inc.

All other brand and product names are trademarks or registered trademarks of their respective owners.

Copyright

Contents

Preface

This guide explains how to install Trifox products, including VORTEX and DesignVision components.

This guide does not discuss installing or managing the database; for instructions and information about database management procedures. For database-specific information you must read your database vendor's documentation.

Background

Trifox Inc. has been serving the relational database market since 1984 through consulting and the development of software products. In 1987, Trifox created SQL*QMX for Oracle. This easy-to-use, powerful querying and report writing tool, which is based on IBM's QMF, continues to be used at thousands of sites. In 1989, Trifox created TRIMtools, a family of application and reportwriting tools now known as DesignVision. DesignVision was developed in response to the OLTP requirements of several large application vendors.

Database Access

VORTEX is an integrated family of products that allows nearly any production application to access SQL data:

- On any or all of the major relational databases.
- Across networks.
- Across platforms.
- With a dramatic increase in the number of concurrent users.
- Without any additional hardware.

In a client/server or multi-tier configuration, VORTEX makes it possible for your SQL applications to access data on different platforms over one or more network configurations. Currently it supports only TCP/IP.

Inherent in this approach are services that allow production applications originally written for one relational database (such as Oracle) can access the same data on another database (such as Informix), even if it is spread across different databases.

VORTEX Precompilers for C and COBOL, as well as a variety of program interfaces, allow existing SQL programs to take full advantage of VORTEX services such as performance enhancement, transaction monitoring, and flat-file database access.

With VORTEXaccelerator in your configuration, you dramatically increase the number of concurrent users who can log on to a specific SQL production application. Your users experience faster performance and you won't have to change any programs or add any hardware.

Application and Report Development

DesignVision DVapp lets you design, generate, and maintain forms-based applications. You can easily port the pop-up windows, customizable menus and submenus, and custom keyboard assignments, in fact the entire application, to Windows .NET, Unix, OpenVMS, or HTML5 with no extra effort.

The reportwriter, TRIMreport, lets you create simple reports quickly, or complex reports with absolute confidence in their power.

When you want to write stand-alone applications (including triggers) without a user interface, the TRIMpl 4GL language gives you the freedom you want. The procedural language has over 100 database-specific functions that help you write powerful applications in very little time.

Reaching Legacy Data

GENESISsql is a SQL processor that accesses low-level data sources such as ISAM, SDMS, ADABAS, RMS, and MicroFocus and makes the data accessible to VORTEX clients. You can add GENESIS data sources to a VORTEX system in a matter of days, simplifying what used to be an enormous task.

Conventions

Screen shots in this manual come from the Windows version of our software.

Trifox documentation uses the following conventions for communicating information:

Example	Describes
`CHOOSE REPORT > [F3] >`	Press [F3] on the CHOOSE REPORT menu and ...
Right-click	Clicking the right mouse button.
Left-click	Clicking the left mouse button.
`connect_string`	Replace italicized text with your own variable.
`vtxnetd`	Text in bold typewriter style represents strings that you type exactly as they appear in the manual.

Support

If you have a question about a TRIFOX product that is not answered in the documentation (paper or online), contact the Customer Support Services group at:

- support@trifox.com
- Trifox Customer Support Services
  2959 Winchester Boulevard
  Campbell, CA 95008
  U.S.A.
- 408-796-1590

Revisions

July 1999

First release.

August 1999

- Updated VORTEX on NT installation instructions to specify that Global PATH statements include the directories that contact database DLL files.
- Updated VORTEX on OS/390 DB2 driver information.
- Updated VORTEX on NT installation instructions to specify that `service.ini` must include all necessary PATHs and environment variables.
- Updated Oracle driver information. Added new DLL for Oracle 8.1.5 and removed outdated information for drivers that are no longer supported.
- Added section on ADABAS C (VTX10) driver.

October 1999

- Added chapter on setting up VORTEXnet.
- Added chapters on connect string, logging, and messages to change the document from a simple installation manual to one that includes operational information.

November 1999

Updated connect and build information for VORTEX to reflect enhancement to shared libraries on Unix.

January 2000

- Corrected typographical errors.
- Incorporated all relevant documentation from ***Trifox Resource Manual*** including troubleshooting procedures, format masks, initialization file, and environment variables.

February 2000

Added chapter on new feature, VORTEXweb. See Chapter 7 for details.

March 2000

Added instructions for installing and usig the Unix System Services Extension on MVS.

Added new *db_connect_string* for Oracle 8 and updated BADCOV error message.

Added new *db_connect_string* for DB2 on MVS.

April 2000

- Added detailed instructions on how to APF authorize the TRIFOX.LOAD library on page 15.
- Updated file and process names to reflect change from WDD6* names to GDS6*.

May 2000

Made formatting changes.

Continued to update file and process names to reflect change from WDD6* to GDS6*.

Added driver information for “native” MVS platform connections.

September 2000

Add ADABAS C ddcard information for running under USS.

July 2001

Add OpenVMS installation instructions.

May 2009

Add PostgreSQL driver.

August 2009

Add AM/PM format mask keywords.

March 2010

Add DV_PREFIX environment variable

January 2013

Add unixODBC configuration parameters to VORTEXodbc section.

March 2015

Add -v6 option to vtxnetd.

August 2015

Add SSL options.

November 2015

Enhance SQL Server connection string definition.

March 2016

Update the Installing VORTEX on Windows chapter.

April 2017
Update the Windows Service chapter.

March 2019
Update the vtxnetd -f option to include preloading DLLs.

November 2019
Update the SSL section.

May 2020
Update the VORTEXjava and VORTEXjdbc section.

June 2020
Update the VORTEX_API_LOGOPTS and VORTEX_HOST_LOGOPTS definitions.

November 2020
Add tcp_keepalive keyword to net.ini.

December 2021
Update the SSL section to include SChannel.

Chapter 1 Installing VORTEX on Windows

This chapter documents the restore process and all modifications you must make for VORTEXserver and necessary database drivers on the Windows platform.

Install the VORTEXserver Files

After you have downloaded the Vortex installation file from the Trifox web site, start the process by running it from any Command Prompt or Explorer window.

Environment Variables

The installation program will set the following environment variables. You can check these using the **Control Panel** (Control Panel > System > Advanced System Settings > Environment tab). On 64bit systems, 64bit Vortex is installed in `C:\Program Files` and 32bit Vortex in `C:\Program Files (x86)`. On 32bit systems, Vortex is installed in `C:\Program Files`:

- ***VORTEX_HOME*** — points to the root directory where VORTEXserver is currently installed.

  `C:\Program Files\Trifox\`

- ***GENESIS_HOME*** — points to the root directory where GENESIS is currently installed.

  `C:\Program Files\Trifox\`

- ***PATH*** — includes the directory where the VORTEX Server programs and DLLs are currently installed.

  `C:\Program Files\Trifox\BIN`

Running VORTEX Server as a Windows Service

Vortex Server can be run either from a Command Prompt or as a Windows service. The Vortex Service method ensures that Vortex Server automatically starts up when the system is rebooted.

Install the VORTEX Service

To install the Vortex Service, open a Command Prompt and type:

```
vortex -install
```

Customize the Service

1. In the Windows **Control Panel**, open Administrative Tools, **Services** and highlight **Vortex Service.**
2. Click **Startup ….**
3. Set *Startup Type* to **Automatic**.
4. Set *Log On As* to a Windows account that has the User Right to log on as a batch job.
5. Click **OK** to close the Startup dialog box.
6. Click **Close** to close the Services Control Panel.
7. Open the **System** from the Control Panel window.
8. Select the **Environment** tab.
9. In System Variables, highlight **VORTEX_SERVICE_FILE** and ensure that its Value (shown at the bottom of the dialog) is the `%TRIM_HOME%\LIB\vortex.srv` file.

   For example:

   ```
   VORTEX_SERVICE_FILE=c:\Program Files\Trifox\LIB\vortex.srv
   ```

10. Click **OK** to close the System dialog.
11. Close the Control Panel.
12. Make sure that the file `vortex.srv` exists in the named directory, all the necessary PATHs and environment variables for the databases you plan to access are set and that each line contains a valid end-of-line character. The `vortex.srv` file shipped with VORTEXserver will work in most cases, however some databases require specialized environment values and these must be set before VORTEXserver starts. For example:

    ```
    #
    # If c:\program files\trifox\bin is not part of the SYSTEM
    # PATH, then add it in.
    PATH=%PATH%;C:\Program Files\Trifox\Bin
    #
    # Set Vortex Environment Variables
    #
    TRIM_HOME=C:\Program Files\Trifox
    GENESIS_HOME=C:\program Files\Trifox
    TEMP=c:\TEMP
    #
    # Start vtxnet2.exe as a detached process
    #
    c:\Program Files\Trifox\Bin\vtxnet2.exe -p1958
    ```

13. Restart the Vortex Service to bring the VORTEX Server up as a service.

VORTEX Service Operations

The VORTEX Service starts the program as a DETACHED PROCESS, sleeps for 5 seconds (long enough to communicate to the service manager that everything is working), and then stops. This strategy allows you to startup multiple programs.

The **Stop Service** option has no effect on Vortex Service. Use VTXKILL to stop the VORTEX Server daemon.

To remove the Vortex Service from your Windows System, from any Command Prompt, type:

```
vortex -remove
```

VORTEX Service memory usage

VORTEX Service shares the same heap memory as the Windows Desktop. For very large installations, it is possible that VORTEX Service, in conjunction with other products on the system, will exhaust this memory resource. In this case, you may see an error message during connection, "Initialization of the dynamic library <system>\system32\user32.dll failed." It may alternatively refer to "kernel32.dll". Or you simply see a "recv: Connection reset by peer" error message. To fix this, decrease the memory allocated to each Windows Desktop. This is done by modifying the Windows Registry entry

HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Session Manager\Subsystems\Windows

This entry consists of a long string of values, of which the SharedSection item is of interest. The third value is the one that must be modified. Please refer to the Microsoft Knowledge Base Article 184802 for more information.

Driver-Specific Instructions

The following sections detail actions necessary to set up your environment for database-specific driver support. Only make the modifications necessary for the drivers you are going to use.

NOTE: *Be sure that your System PATH statement includes the directory that contains database DLL files. See each driver instruction section for specific examples.*

Sybase (VTX2)

Choose one of 2 versions of the Sybase driver:

- `VTX2.DLL` for Sybase (DB-LIB Interface)
- `VTX2_CT.DLL` for Sybase 11.9 (CT-LIB Interface)

To use the CT-LIB interface, rename `VTX2_CT.DLL` to `VTX2.DLL`. Be sure that your System PATH statement includes the directory that contains the necessary DLLs. For example,

```
C:\SYBASE\dll
```

Chapter 2 Installing VORTEX on OS/390

This chapter documents the restore process and all modifications you must make for VORTEXserver and necessary database drivers on the OS/390 (MVS) platform.

Restore MVS Datasets

After you have obtained and exploded the file `TRIFOX.tar.gz` from the Trifox web site, you are ready to restore the datasets, which are in TSO XMIT format.

These datasets are ***always*** required:

- TRIFOX.JOBS.XMIT
- TRIFOX.LOAD.XMIT OS/390 2.8 (OE Sockets)
- TRIFOX.LIB.XMIT

This dataset is required for ***DB2*** users only:

- TRIFOX.DBRM.XMIT

These datasets are required for ***DVrun*** users only:

- TRIFOX.MIR.XMIT
- TRIFOX.RUN.XMIT

These datasets are required for ***GENESIS*** users only:

- TRIFOX.GENESIS.XMIT
- TRIFOX.GENESIS.LIB.XMIT
- TRIFOX.GENESIS.UNLD.XMIT
- TRIFOX.MISC.XMIT

These datasets are required for GENESIS users who want to load the ***Trifox demo files*** into ADABAS:

- TRIFOX.ORG.UNLD.XMIT
- TRIFOX.STAFF.UNLD.XMIT
- TRIFOX.ZIPCODES.UNLD.XMIT

☞ ***1. Allocate the following empty datasets on your MVS system.***

1. Data Set Name . . . : TRIFOX.DBRM.XMIT

```
General Data                          Current Allocation
```

```
Volume serial . . . : TRIFOX          Allocated blocks  . : 52
Device type . . . . : 3380            Allocated extents . : 1
Organization  . . . : PS
Record format . . . : FB
Record length . . . : 80
Block size  . . . . : 3120            Current Utilization
1st extent blocks . : 52              Used blocks . . . . : 26
Secondary blocks  . : 15              Used extents  . . . : 1
```

2. Data Set Name . . . : TRIFOX.JOBS.XMIT

```
General Data                          Current Allocation
Volume serial . . . : TRIFOX          Allocated blocks  . : 52
Device type . . . . : 3380            Allocated extents . : 1
Organization  . . . : PS
Record format . . . : FB
Record length . . . : 80
Block size  . . . . : 3120            Current Utilization
1st extent blocks . : 52              Used blocks . . . . : 27
Secondary blocks  . : 15              Used extents  . . . : 1
```

3. Data Set Name . . . : TRIFOX.LOAD.XMIT

```
General Data                          Current Allocation
Volume serial . . . : TRIFOX          Allocated megabytes : 8
Device type . . . . : 3380            Allocated extents . : 1
Organization  . . . : PS
Record format . . . : FB
Record length . . . : 80
Block size  . . . . : 3120            Current Utilization
1st extent blocks . : 2002            Used blocks . . . . : 2,688
Secondary blocks  . : 250             Used extents  . . . : 1

Creation date . . . : 1999/06/25
Referenced date . . : 1999/06/25
Expiration date . . : ***None***
```

4. Data Set Name . . . : TRIFOX.LIB.XMIT

```
General Data                          Current Allocation
Volume serial . . . : TRIFOX          Allocated blocks  . : 26
Device type . . . . : 3380            Allocated extents . : 1
Organization  . . . : PS
Record format . . . : FB
Record length . . . : 80
Block size  . . . . : 3120            Current Utilization
1st extent blocks . : 26              Used blocks . . . . : 10
Secondary blocks  . : 8               Used extents  . . . : 1
```

5. Data Set Name . . . : TRIFOX.MISC.XMIT

```
General Data                          Current Allocation
Volume serial . . . : TRIFOX          Allocated blocks  . : 26
Device type . . . . : 3380            Allocated extents . : 1
Organization  . . . : PS
Record format . . . : FB
Record length . . . : 80
```

```
Block size  . . . . : 3120          Current Utilization
1st extent blocks . : 26            Used blocks . . . . : 6
Secondary blocks  . : 8             Used extents  . . . : 1
```

6. Data Set Name . . . : TRIFOX.MIR.XMIT

```
General Data                        Current Allocation
Volume serial . . . : TRIFOX        Allocated blocks  . : 78
Device type . . . . : 3380          Allocated extents . : 1
Organization  . . . : PS
Record format . . . : FB
Record length . . . : 80
Block size  . . . . : 3120          Current Utilization
1st extent blocks . : 78            Used blocks . . . . : 30
Secondary blocks  . : 22            Used extents  . . . : 1
```

7. Data Set Name . . . : TRIFOX.RUN.XMIT

```
General Data                        Current Allocation
Device type . . . . : 3380          Allocated extents . : 1
Organization  . . . : PS
Record format . . . : FB
Record length . . . : 80
Block size  . . . . : 3120          Current Utilization
1st extent blocks . : 26            Used blocks . . . . : 15
Secondary blocks  . : 8             Used extents  . . . : 1
```

8. Data Set Name . . . : TRIFOX.GENESIS.XMIT

```
General Data                        Current Allocation
Volume serial . . . : TRIFOX        Allocated blocks  . : 26
Device type . . . . : 3380          Allocated extents . : 1
Organization  . . . : PS
Record format . . . : FB
Record length . . . : 80
Block size  . . . . : 3120          Current Utilization
1st extent blocks . : 26            Used blocks . . . . : 1
Secondary blocks  . : 8             Used extents  . . . : 1
```

9. Data Set Name . . . : TRIFOX.GENESIS.LIB.XMIT

```
General Data                        Current Allocation
Volume serial . . . : TRIFOX        Allocated blocks  . : 26
Device type . . . . : 3380          Allocated extents . : 1
Organization  . . . : PS
Record format . . . : FB
Record length . . . : 80
Block size  . . . . : 3120          Current Utilization
1st extent blocks . : 26            Used blocks . . . . : 2
Secondary blocks  . : 8             Used extents  . . . : 1
```

10. Data Set Name . . . : TRIFOX.GENESIS.UNLD.XMIT

```
General Data                        Current Allocation
Volume serial . . . : TRIFOX        Allocated blocks  . : 26
Device type . . . . : 3380          Allocated extents . : 1
Organization  . . . : PS
Record format . . . : FB
```

```
Record length . . . : 80
Block size  . . . . : 3120          Current Utilization
1st extent blocks . : 26            Used blocks . . . . : 1
Secondary blocks  . : 6             Used extents  . . . : 1
```

11. Data Set Name . . . : TRIFOX.ORG.UNLD.XMIT

```
General Data                        Current Allocation
Volume serial . . . : TRIFOX        Allocated blocks  . : 26
Device type . . . . : 3380          Allocated extents . : 1
Organization  . . . : PS
Record format . . . : FB
Record length . . . : 80
Block size  . . . . : 3120          Current Utilization
1st extent blocks . : 26            Used blocks . . . . : 1
Secondary blocks  . : 6             Used extents  . . . : 1
```

12. Data Set Name . . . : TRIFOX.STAFF.UNLD.XMIT

```
General Data                        Current Allocation
Volume serial . . . : TRIFOX        Allocated blocks  . : 26
Device type . . . . : 3380          Allocated extents . : 1
Organization  . . . : PS
Record format . . . : FB
Record length . . . : 80
Block size  . . . . : 3120          Current Utilization
1st extent blocks . : 26            Used blocks . . . . : 1
Secondary blocks  . : 6             Used extents  . . . : 1
```

13. Data Set Name . . . : TRIFOX.ZIPCODES.UNLD.XMIT

```
General Data                        Current Allocation
Volume serial . . . : TRIFOX        Allocated blocks  . : 754
Device type . . . . : 3380          Allocated extents . : 1
Organization  . . . : PS
Record format . . . : FB
Record length . . . : 80
Block size  . . . . : 3120          Current Utilization
1st extent blocks . : 754           Used blocks . . . . : 559
Secondary blocks  . : 223           Used extents  . . . : 1
```

2. FTP the exploded TRIFOX TSO XMIT datasets to your MVS system.

Start an FTP session on the machine from within the directory in which you exploded the `TRIFOX.tar.gz` files and establish a connection to your MVS system.

After you have established the FTP connection, type `bin` to ensure that the transport is binary.

Then, issue the follow `put` commands:

```
put TRIFOX.DBRM.XMIT 'TRIFOX.DBRM.XMIT'
put TRIFOX.JOBS.XMIT 'TRIFOX.JOBS.XMIT'
put TRIFOX.LOAD.XMIT 'TRIFOX.LOAD.XMIT'
put TRIFOX.LIB.XMIT 'TRIFOX.LIB.XMIT'
```

```
put TRIFOX.MISC.XMIT 'TRIFOX.MISC.XMIT'
put TRIFOX.MIR.XMIT 'TRIFOX.MIR.XMIT'
put TRIFOX.RUN.XMIT 'TRIFOX.RUN.XMIT'
put TRIFOX.GENESIS.XMIT 'TRIFOX.GENESIS.XMIT'
put TRIFOX.GENESIS.LIB.XMIT 'TRIFOX.GENESIS.LIB.XMIT'
put TRIFOX.GENESIS.UNLD.XMIT 'TRIFOX.GENESIS.UNLD.XMIT'
put TRIFOX.ORG.UNLD.XMIT 'TRIFOX.ORG.UNLD.XMIT'
put TRIFOX.STAFF.UNLD.XMIT 'TRIFOX.STAFF.UNLD.XMIT'
put TRIFOX.ZIPCODES.UNLD.XMIT 'TRIFOX.ZIPCODES.UNLD.XMIT'
```

3. Restore the TRIFOX TSO XMIT datasets to your MVS system.

Issue the following receive command from the ISPF Command shell:

```
receive indsname('dataset_XMIT')
```

Receive the following prompt:

```
INMR901I Dataset dataset from P390 on NODENAME
INMR906A Enter restore parameters or 'DELETE' or 'END' +
```

Issue the following reply:

```
dataset('dataset') volume(TRIFOX)
```

Where *dataset* is the name of the dataset you require for your installation (from the list on page 10) and *volume* is the name of the volume in which you want the data set allocated.

Customize Files

Rules for Modification

The following job streams are used as templates by the VORTEXserver daemon. Do ***not*** submit them directly to MVS.

```
VTX0
VTX1
VTX3
VTX4
VTX7
VTX9
```

In addition, do not remove the "/" character at the end of the JOB card. The VORTEXserver needs it to parse the JOB card correctly. You can remove the text "<= MUST HAVE THIS" to make more room on the JOB card line for JOB Accounting information.

Finally, do not remove or modify the PARM='?' or PARM=OPER cards.

☞ *Customize the following jobs for all installations*

- ***VTXNETD*** — The VORTEX daemon is the process that listens for requests to fulfill.
- ***VTXPING*** — The process you use to test the connection.
- ***VTXKILL*** — The process you use to stop the daemon.

1. Modify the STEPLIB to point to the TRIFOX.LOAD library.
2. APF authorize the TRIFOX.LOAD library.

   Any module such as VTXNETD that is linked with an AC code of 1 can issue privileged instructions. There are two ways to APF authorize a load library:

   - Through an IPL of the system
   - Dynamically via an Operator command

   To permantly APF authorize a library you must add an entry to the current `PROGxx` member in `SYS1.PARMLIB`. The default member is normally 00.

   The entry would look somethink like this:

   ```
   APF ADD
   DSNAME(TRIFOX.LOAD)                    VOLUME(TRIFOX)
   ```

   To dynamically APF authorize a library, you can issue the following operator command on the console:

   ```
   SETPROG APF,,ADD,DSNAME=TRIFOX.LOAD,VOLUME=TRIFOX
   ```

   NOTE: *You must add a forward slash (/) in front of the SETPROG command when issuing it from the SDSF LOG panel.*

3. Modify the SYSTCPD DD statement to point to your system's TCP/IP data file.
4. Modify the EV DD statement in VTXNETD to point to the environment variables file ENVFILE in the TRIFOX.LIB library.
5. Modify the following DD statements in VTXNETD to point to the TRIFOX.JOBS library.

   ```
   VTX0
   VTX1
   VTX2
   VTX3
   VTX4
   VTX5
   VTX6
   VTX7
   VTX9
   ```

6. If you plan to use Started Tasks for a specific database driver, then you must customize the JOBINI file in TRIFOX.LIB library.

   In the example below:

- DB2 is running with batch jobs from DD statement VTX7
- ESQ is running with started tasks VTX9STC

```
rem ---------- VORTEXnet Daemon specifics
rem job_prefix     V
drv7_job       DD:VTX7
drv7_batch     yes
drv9_job       VTX9STC
drv9_batch     no
```

Each Started Task name that you specify in the JOBINI file must also have an OMVS Segment defined in the RACF profile.

For example, a RACF profile for Started Task VTX9STC has an OMVS segment defined as:

```
UID= 0000000200
HOME= /u/p390
PROGRAM= /bin/sh
```

How it Works

VORTEXserver's VTXNETD process waits for an incoming connection request. It parses the request to determine which DBMS proxy to start. It then starts the DBMS proxy in one of two ways: job or started task. This is determined by the target DBMS's JOBINI entry. If the entry specifies a job, then VTXNETD reads in the jcl for the job, modifies it to match the incoming connection request and then submits it to the internal reader. This method works with JES2 but does not work with JES3. For JES3 installations, you must use started tasks. In this case, VTXNETD makes an SVC99 call using the target DBMS's PROC file to start the DBMS proxy.

Driver-Specific Instructions

The following sections detail actions necessary to set up your environment for database-specific driver support. Only make the modifications necessary for the drivers you are going to use.

NOTE: *You must not modify the job name in the VTXn job. It must be //VTXnSTC for the started tasks and //Vn000000 for the jobs. If your installation requires job names to begin with a different prefix, uncomment the job_prefix line in the JOBINI file and set the prefix to what your site requires.*

Oracle Driver (VTX0)

Customize job VTX0 and PROC VTX0STC so that

1. STEPLIB points to the TRIFOX.LOAD library and the Oracle CMDLOAD library.
2. SYSTCPD DD statement points to your system's TCP/IP Data file.
3. ORA@ORA1 DD DUMMY card points to the correct Oracle subsystem/instance that you plan to access.

For example, if your Oracle instance is ORA2 then you should change //ORA@ORA1 DD DUMMY to //ORA@ORA2 DD DUMMY.

NOTE: *The current VTX0 load module in TRIFOX.LOAD is for Oracle 7.3. If you plan to access an Oracle 8.0 database, you must rename the load module VTX0804 to VTX0.*

Design Vision (DVrun Driver for DB2 version 6 (VTX1))

Customize job VTX1 and PROC VTX1STC so that

1. STEPLIB points to the TRIFOX.LOAD library and the DB2 LOAD library.
2. SYSTCPD DD statement points to your systems TCP/IP Data file.

Customize job BIND7612 so that

1. JOBLIB points to the DB2 LOAD library.
2. DBRMLIB DD points to the TRIFOX.DBRM library.
3. SYSTSIN DD cards point to the DB2 Subsystem you plan to access.
4. Submit job BIND7511 to MVS to bind DBRM TDB7612 into the DB2 sub-system you plan to access.

VORTEXnet Driver (VTX3)

Customize job VTX3 and PROC VTX3STC so that

1. STEPLIB points to the TRIFOX.LOAD library.
2. SYSTCPD DD statement points to your systems TCP/IP Data file.

ADABAS C Version 6.2 GENESIS Driver (VTX4)

The GENESIS data source file specified in the AFILE parameter is read from the PDS library specified in the `GENESIS_HOME` environment variable. The `GENESIS_HOME` environment variable is defined in the environment variables file, ENVFILE, in the TRIFOX.LIB library.

Customize job VTX4 and PROC VTX4STC so that

1. STEPLIB points to the TRIFOX.LOAD library and the ADABAS LOAD library.
2. SYSTCPD DD statement points to your system's TCP/IP data file.
3. ADARUN member in the TRIFOX.JOBS library points to the correct SVC and DBID of the database you plan to access.

Customize ADALOD JOBS so that

1. ADALOD JOB LODCATLG loads the GENESIS Catalog (TRIFOX.GENESIS.UNLD file) into your ADABAS database.

2. ADALOD JOB LODORG loads the DEMO TABLE ORG (TRIFOX.ORG.UNLD file) into your ADABAS database.
3. ADALOD JOB LODSTAFF loads the DEMO TABLE STAFF (TRIFOX.STAFF.UNLD file) into your ADABAS database.
4. ADALOD JOB LODZIPCO loads the DEMO TABLE ZIPCODES (TRIFOX.ZIPCODES.UNLD file) into your ADABAS database.

Customize the GENESIS datasource members so that

DB*nnn* in the TRIFOX.GENESIS library points to the Database and File number of the GENESIS catalog that was loaded by the LODCATLG job. For example: PDS member DB235 would look like:

```
database   235
dictionary  20
```

NOTE: *You must enter the case-sensitive keywords "database" and "dictionary" in lower case.*

Customize LE environment variables so that

ENVFILE in the TRIFOX.LIB library points GENESIS_HOME (environment variable) to the TRIFOX.GENESIS high level qualifier. For example:

```
GENESIS_HOME='TRIFOX.GENESIS
for datasets TRIFOX.GENESIS
             TRIFOX.GENESIS.LIB
```

Customize the PROCs GDS6PROC and GDS6FPRC so that

1. STEPLIB points to the TRIFOX.LOAD library and ADABAS LOAD library.
2. Make these PROCs available to your MVS system by either:
   - Adding the TRIFOX.JOBS library to the list of PROC libs

   or
   - Copying them to one of your current PROC libs.

Customize job GDS6INIT so that

1. GDS6.EV DD statement points to the TRIFOX.LIB library.
2. GDS6.DDCARD DD statement points to the TRIFOX.JOBS library.
3. Initialize the GENESIS Catalog by submitting job GDS6INIT to MVS.

Setup for Trifox Demonstration Files

Customize PDS members so that

1. ORGIN in the TRIFOX.MISC library points to the File number of the ORG table that was loaded by the LODORG job. (Note that it currently points to file 21.)

2. STAFFIN in the TRIFOX.MISC library points to the File number of the STAFF table that was loaded by the LODSTAFF job. (Note that it currently points to file 22.)
3. ZIPIN in the TRIFOX.MISC library points to the File number of the ZIPCODES table that was loaded by the LODZIPCO job. (Note that it currently points to file 23.)

Customize job GDS6ADD so that

1. GDS6.EV DD statements point to the TRIFOX.LIB library.
2. GDS6.DDCARD DD statements point to the TRIFOX.JOBS library.

Incorporate the table definitions for ORG, STAFF and ZIPCODES by submitting job GDS6ADD to MVS.

Setup for ADABAS Demonstration Files

Customize job GENDDADA so that

1. GDS6FDT.EV DD statements point to the TRIFOX.LIB library
2. GDSFDT.GDS6FOUT DD statements point to the TRTFOX.MISC library.
3. GDSFDT.GDS6FSYN DD statements point to the TRTFOX.MISC library.
4. The AFILE parameters in job GENDDADA points to the GENESIS datasource and files numbers for the EMPLOYEES and VEHICLES demo files.

   The current defaults are:

   ```
   DB235 1 for the EMPLOYEES file
   DB235 2 for the VEHICLES file
   ```

   and “1” and “2” are the file numbers of the EMPLOYEES and VEHICLES demo files.
5. Generate the GENESIS Data Definition cards for EMPLOYEES and VEHICLES by submitting job GENDDADA to MVS.

Customize job IMPDDADA so that

1. GDS6.EV DD statements point to the TRIFOX.LIB library
2. GDS6.DDCARD DD statements points to the TRIFOX.JOBS library.
3. The AFILE parameters in job GENDDADA to point to the GENESIS datasource and the TRIFOX.MISC library.
4. Incorporate the GENESIS Data Definition cards for EMPLOYEES and VEHICLES by submitting job IMPDDADA to MVS.

DB2 Driver (VTX7)

Customize job VTX7 and PROC VTX7STC so that

1. STEPLIB points to the TRIFOX.LOAD library and the DB2 LOAD library.
2. SYSTCPD DD statement points to your systems TCP/IP Data file.

Customize job BIND7511

1. JOBLIB to point to the DB2 LOAD library.
2. DBRMLIB DD to point to the TRIFOX.DBRM library.
3. SYSTSIN DD cards to point to the DB2 Subsystem you plan to access.
4. Submit job BIND7511 to MVS to Bind DBRM TDB7511 into the DB2 sub-system you plan access.

NOTE: *To access a DB2 6.1 database, rename load module VTX761 to VTX7 and customize BIND7612 to bind DBRM TDB7612 into the DB2 subsystem you plan to access.*

ADABAS SQL Server Version 1.4 Driver (VTX9)

Customize job VTX9 and PROC VTX9STC so that

1. STEPLIB points to the TRIFOX.LOAD library, ESQ LOAD library and VO PARM LOAD library.
2. SYSTCPD DD statement points to your systems TCP/IP Data file.
3. ADARUN member in the TRIFOX.JOBS library points to the correct SVC and DBID of the database that ESQ accesses.

Customize the ESQ Parameters for the ESQ Server to the following settings:

1. BUFFER SIZE in the GLOBAL section to at least 50000.
2. MAX REPLY LENGTH in the SERVER section to at least 32000.
3. MAX DYBANIC NPS in the RUNTIME section to at least 25.
4. MAX CURSORS in the RUNTIME section to be at least 50.
5. MAX STACK SIZE in the RUNTIME section to at least 500.

Example

```
GLOBAL
BEGIN
   DIRECTORY ( DBID = xxx, FNR = xxx )
   ERROR     ( DBID = xxx, FNR = xxx )
   FILE OUTPUT RECORD LENGTH = 80
   FILE INPUT  RECORD LENGTH = 80
   BM ( BUFFER SIZE = 50000 )
   END
   SERVER
   BEGIN
      MAX REPLY LENGTH = 32000
      END
      RUNTIME
   BEGIN
      MAX DYNAMIC MPS = 25
      MAX CURSORS     = 50
      MAX STACK SIZE  = 500
   END
```

NOTE: *If you plan to run VTX9 in linked-in mode, you must specify the linked-in version of your VO PARMs module in the STEPLIB. You must also include an ESQPARMS DD statement that points to a linked-in version of your ESQPARMS file.*

Unix System Services Extension

This section describes how to install and configure VORTEXserver and necessary database driver on Unix Systems Services on the OS/390 (MVS) platform.

Get the Files

If you have not already gotten a compressed file from Trifox, you can get an evaluation version of VORTEX from `http://www.trifox.com/evalform/`or download by anonymous FTP. If you are a supported customer, follow instructions for getting necessary components from your specific directory using your own UID and password.

Preparing the Server

One you have the necessary file(s), prepare to configure VORTEX.

1. Create a installation directory on your target server machine, for example

   ```
   /usr2/trim/
   ```

2. Uncompress and tar the downloaded file into the installation directory. If you do not have gzip on your USS system, uncompress the file on Windows using WinZip or Unix using gzip and then ftp it to your USS installation directory, making sure to specify binary as the transfer method.

   ```
   cd /usr2/trim
   gzip -d compressed-filename
   tar -xvf uncompressed-filename
   ```

3. Set the +p extended attribute on vtxnetd.

   ```
   extattr +p vtxnetd
   ```

Please see the notes at the end of this section if the `extattr` command is not enabled on your system.

4. Set environment variable LIBPATH to include trim/bin directory.

   ```
   export LIBPATH=/usr2/trim/bin
   ```

5. Set environment variable STEPLIB to include the TRIFOX Load library and the database vendor's Load Libraries, for example:

   ```
   export STEPLIB=TRIFOX.LOAD:DSN610.SDSNLOAD:SAG.ADA622.LOAD
   ```

6. If you are planning to use GENESIS, you may want to set GENESIS_HOME before starting the vtxnetd process.

   ```
   export GENESIS_HOME=//\'TRIFOX.GENESIS
   ```

7. Start the vtxnetd in background

```
nohup vtxnetd -p1959 -a &
```

Enabling the extattr Command

You can set up the RACF definitions to allow the `extattr` command to be issued.

The server daemon for USS issues `setuid()` commands and requires that the `+p` attribute be set. Issue the following command to set the correct extended attribute for vtxnetd:

```
extattr +p vtxnetd
```

NOTE: *Only users with the correct permission can turn on the extended attribute.*

The following example shows the RACF command used to give this permission to user IBMUSER:

```
RDEFINE FACILITY BPX.FILEATTR.PROGCTL UACC(NONE)
PERMIT BPX.FILEATTR.PROGCTL CLASS(FACILITY) ID(IBMUSER)
ACCESS(READ)
SETROPTS RACLIST(FACILITY) REFRESH
```

You must also Grant userid BPXOINIT access to the `BPX.DAEMON` facility:

```
PERMIT BPX.DAEMON CLASS(FACILITY) ID(BPXOINIT) ACCESS(READ)
```

ADABAS C

Some releases of ADABAS C require a 'ddcard' file in the directory where you start the vtxnetd process. This file is called 'ddcard' and contains the following:

```
ADARUN SVC=237,DATABASE=240,MODE=MULTI,PROGRAM=USER
```

This information is similar to what is located in TRIFOX.JOBS(ADARUN). Obviously your settings for SVC and DATABASE may be different. If you are having difficulty getting your USS connection to ADABAS C to work, then your release may require this ddcard file.

Summary

You have now successfully installed the USS portion of VORTEXserver. Proceed to ***Chapter 6, Starting Vortex***

Chapter 3 Installing VORTEX on OpenVMS

This chapter describes how to install and configure VORTEXserver and necessary database drivers on the OpenVMS platform.

Getting the Files

If you have not already downloaded the installation file from Trifox, then you can get an evaluation version of VORTEX from http://www.trifox.com/evalform/ or download by anonymous FTP. If you are a supported customer, follow instructions for getting necessary components from your specific directory using your own UID and password.

The VORTEXserver file is a product PCSI file. If you need to move it to OpenVMS from a different operating sytem, then be certain to always specify binary as the transport method.

Installing the Server

In the directory in which you stored the PCSI file, type

```
product install *
```

and follow the prompts. You may get a warning message

```
-PCSI-I-NOTSIGNED, product kit is not signed and therefore has no
manifest file
```

It is safe to continue. When the installation has completed, be sure to add

```
@sys$common:[trifox.cin]trifox$symbols.com
```

to your users' login.com file.

You are now ready to connect to a database. Proceed to ***Chapter 6, Starting Vortex.***

Chapter 4 Installing VORTEX on Unix

This chapter describes how to install and configure VORTEXserver and necessary database drivers on the Unix platform.

Getting the Files

If you have not already gotten a compressed file from Trifox, you can get an evaluation version of VORTEX from http://www.trifox.com/evalform/ or download by anonymous FTP. If you are a supported customer, follow instructions for getting necessary components from your specific directory using your own UID and password.

Preparing the Server

Once you have the necessary file(s), prepare to build VORTEX.

1. Create a VORTEX installation directory on your target server machine, for example

   ```
   /usr2/vortex
   ```

2. Uncompress and untar the downloaded file into a temporary directory.

   ```
   gzip -d filename
   ```

Customize the Makefile

Since each database has a variety of link libraries, you may have to modify the appropriate makefile before proceeding.

1. **Choose the correct makefile and executable name**.

If you use …	Choose …	Executable	Modify DBLIBS
ADABAS C w/GENESIS	makesrv.gen6	VTX4	
ADABAS D	makesrv.aad	VTX10	
DB2 (IBM)	makesrv.db2	VTX7	
Informix	makesrv.inf	VTX5	
ODBC w/a vendor solution	makesrv.cli	VTX11	X
Oracle	makesrv.ora	VTX0	
Oracle v8 or higher	makesrv.ora8	VTX0	
Synergex ISAM	makesrv.gen0	VTX4	

If you use …	Choose …	Executable	Modify DBLIBS
Sybase	makesrv.syb	VTX2	
VORTEXaccelerator host service	makesrv.mux	VTXMUXH (vtxhost.mux on Unix)	
VORTEXnet host service	makesrv.net	VTX3	

2. **Modify `DBLIBS`, if necessary.**

   If you are using a database that requires DBLIBS modification (refer to preceding table) you must open the makefile and go through the list of shell variables to find the one that represents the combination of *database* and *operating system* that you are using. Change the shell variable in the makefile to read **`DBLIBS.`**

3. **Run the makefile to create the VORTEX executables.**

   The command uses a compiler to link files. You specify the makesrv extension that represents your database. For example, the statement for Oracle is:

   ```
   makedb ora
   ```

   which generates the Oracle driver VTX0.so and moves it to the bin directory under the installation directory, eg. /usr2/vortex/bin.

   If this command does not work for you, contact Trifox support (see "***Support***" on page 2 for details).

4. **Update the PATH and LD_LIBRARY_PATH environment variables.**

   For example,

   ```
   PATH=$PATH:/usr2/vortex/bin
   LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr2/vortex/bin
   export PATH LD_LIBRARY_PATH
   ```

   If you are running on Solaris 64 bit, you must modify LD_LIBRARY_PATH_64.

You are now ready to connect to a database.

Chapter 5
Installing Clients

VORTEX offers accessware to enable connections from client applications in a variety of languages to many different databases. Two of the languages, COBOL and C, require that you pre-compile applications. C++, ODBC, JDBC, and Java simply require compilation. Perl, an interpreted language, of course requires only that you create a Perl script with VORTEXperl calls.

After you've installed the client components, go to "***Connecting Your Application to VORTEX***" on page 46 for instructions on how to proceed.

VORTEXperl

VORTEXperl documentation and release notes are available online at `www.trifox.com`.

Installing on Windows

1. Create a directory on your machine for the VORTEXperl files. For example,

   ```
   C:\VTXperl
   ```

2. Put the downloaded `.tar` file in that directory and untar it.

   Your directory should contain VORTEXperl files with the extensions `*.pm` and `*.ph` and sample applications, `sample*.pl`, which are set up to connect to a Linux ADABASD database on `ftp.trifox.com`.

3. Move the `dbChannel.pm`, `dbCursor.pm`, and `dbDefine.ph` files into your Perl lib directory. Alternatively you can add the directory you created in step 1 to your Perl scripts' `INC()` directory list.

Installing on Unix

1. Create a directory on your machine for the VORTEXperl files. For example,

   ```
   /usr2/vortex
   ```

2. Put the downloaded `.tar` file in that directory and untar it.

   Your directory should contain VORTEXperl files with the extensions `*.pm` and `*.ph` and sample applications, `sample*.pl`, which are set up to connect to a Linux ADABASD database on `ftp.trifox.com`.

3. Because Perl is an interpreted language, you must ensure that the Perl executor's location is correct in the samples. The samples ship with the location specified as `/usr/local/bin`. If that's not where your Perl interpreter is located, you must change the line in the sample programs. The following error typically means that the location is not correct in the sample file:

```
sample0.pl: command not found
```

VORTEXcobol

VORTEXcobol is a precompiler. You must run it against your COBOL source before actually compiling the COBOL source file.

A reference manual is available for you to read online or download and print. Go to `www.trifox.com/library/docs.html`.

Installing on Windows

VORTEXcobol is installed as an option in your VORTEX for Windows package.

Installing on Unix

1. Create a directory on your machine. For example,

   ```
   /usr2/vortex
   ```

2. Put the downloaded `.tar` file in that directory and untar it.

   The directory now contains four subdirectories:

   - bin — vtxcob
   - lib — Support files
   - obj — Linking files
   - demo — Some sample programs

3. Create an environment variable for VORTEX. For example,

   ```
   setenv VORTEX /usr2/vortex          /* C shell */
   VORTEX=/usr2/vortex;export VORTEX /* Bourne shell */
   ```

4. Add `$VORTEX/bin` to your shell's PATH variable.

   -OR-

   Move/link the vtxcob executable to some directory that is already in your PATH, for example, `/usr/local/bin`.

5. Once you have precompiled your COBOL source, you need to compile it with your COBOL compiler. VORTEXcobol generates code supported by the following compilers:

   - AcuCOBOL
   - Fujitsu COBOL
   - Micro Focus COBOL

6. Link your object(s) with your COBOL vendor's libraries as well as the VORTEX libraries. The evaluation kit only supports VORTEXserver connections, so the VORTEX library list is

   ```
   -L$VORTEX/bin -lVTXAPI.so $VORTEX/obj/tb2.a $NETLIB $DLLIB
   ```

For HPUX, set `DLLIB="-Wl,+s"`, for Linux, `"-ldl"`, all others leave it empty. For SOLARIS, set `NETLIB="-lsocket -lnsl"`, for Linux, `"-lcrypt -lnsl"`, all others leave it empty.

7. Your connect string contains the network information, as described in "***Connecting Your Application to VORTEX***" on page 46.

VORTEXc

VORTEXc is a precompiler. You must run it against your C source before actually compiling the C source file.

A reference manual is available for you to read online or download and print. Go to `www.trifox.com/library/docs.html`.

Installing on Windows

VORTEXc is installed as an option in your VORTEX for Windows package.

Installing on Unix

1. Create a directory on your machine. For example,

   ```
   /usr2/vortex
   ```

2. Put the downloaded `.tar` file in that directory and untar it.

   The directory now contains four subdirectories:

   - bin — vtxc
   - lib — Support files
   - obj — Linking files

3. Create an environment variable for VORTEX. For example,

   ```
   setenv VORTEX /usr2/vortex          /* C shell */
   VORTEX=/usr2/vortex;export VORTEX /* Bourne shell */
   ```

4. Add `$VORTEX/bin` to you shell's PATH variable.

   -OR-

   Move or link the vtxc executable to some directory that is already in your PATH, for example, `/usr/local/bin`.

5. Once you have precompiled your C source, you need to compile it with your C compiler. You may want to use the sample compile and link script in the `bin` directory – `ccvtx`. To run it, simply type

   ```
   ccvtx filename
   ```

6. Your connect string contains the network information, as described in "***Connecting Your Application to VORTEX***" on page 46 .

VORTEX++

A reference manual is available for you to read online or download and print. Go to `www.trifox.com/library/docs.html`.

Installing on Windows

VORTEX++ is installed as an option in your VORTEX for Windows package.

Installing on Unix

1. Create a directory on your machine for the VORTEX++ files. For example, `/usr2/vortex/c++`
2. Put the downloaded `.tar` file in that directory and untar it.

   The directory now contains a set of header files (`*.hxx`), two libraries, a shared library and sample code (`*.cxx`). The make file builds the sample programs.
3. Move the `VTX3.so` file into a directory pointed by your operating system's shared library environment variable, typically `LD_LIBRARY_PATH`. Alternatively you can add this directory to the `$LD_LIBRARY_PATH` environment variable:

   C Shell

   ```
   setenv LD_LIBRARY_PATH "$LD_LIBRARY_PATH":"/usr2/vortex/c++"
   ```

   Bourne Shell

   ```
   LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr2/vortex/c++
   export LD_LIBRARY_PATH
   ```

VORTEXodbc

Installing on Windows

VORTEXodbc is installed as an option in your VORTEX for Windows package.

Installing on Unix

1. Create a directory on your machine for the VORTEXodbc files. For example, `/usr2/vortex/odbc`
2. Put the downloaded `.tar` file in that directory and untar it.

   The directory now contains three directories:

   - bin — VORTEXodbc shared libraries
   - lib — odbc.ini file
   - example — test program

3. Add the bin directory to the `$LD_LIBRARY_PATH` environment variable:

   C Shell

```
setenv LD_LIBRARY_PATH "$LD_LIBRARY_PATH":"/usr2/vortex/odbc/
bin"
```

Bourne Shell

```
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr2/vortex/odbc/bin
export LD_LIBRARY_PATH
```

4. Assuming that you are using odbcUNIX, use ODBCConfig to create a VORTEXodbc driver, specifying /usr2/vortex/bin/libtod.so as the driver and /usr2/vortex/bin/libtodadm.so as the setup program.

   Create a VORTEX_HOME environment variable that points to your installation directory:

   C Shell

   ```
   setenv VORTEX_HOME /usr2/vortex/odbc
   ```

   Bourne Shell

   ```
   VORTEX_HOME=/usr2/vortex/odbc
   export VORTEX_HOME
   ```

5. Modify the /usr2/vortex/odbc/lib/odbc.ini file. As not all Unix odbc installations have a driver manager such as unixODBC, VORTEXodbc uses this file to build the actual connect string. It does this by gathering the userid, password, and DSN name given during the SQLConnect() call and searching the odbc.ini file for an entry "dsn_<DSN>". For example, if your unixODBC DSN name is vortex, VORTEXodbc will look for "dsn_vortex" in $VORTEX_HOME/lib/odbc.ini, and replace the %s characters with the userid and password.

   If $VORTEX_HOME/lib/odbc.ini does not exist or the "dsn_<DSN>" entry is not found, then the driver will attempt to load libodbcinst.so. If this is successful, it will then use the unixODBC calls to get its configuration parameters from the unixODBC odbc.ini file.

Parameter	Type	Description
ConnectString	String	For network connections !VTX4[,remote env vars[,...]]
DataSource	String	Genesis driver:filename, e.g. rms:rms.ini
Description	String	DSN Description
EnvVars	String	Comma separated local env vars
FetchBufferSize	Integer	Size of fetch buffer in bytes (default: 4096, min: 0, max: 100MB)
Host	String	Name or IP address of remote server
MaxColumns	Integer	Maximum number of resultset columns (default: 256, min: 4, max: 32768)

Parameter	Type	Description
MaxDBCursors	Integer	Maximum number of database cursors (default: 64, min: 4, max: 1024)
MaxStatements	Integer	Maximum number of database statements (default: 64, min: 4, max: 32768)
MemorySortPages	Integer	Maximum number of sort pages kept in memory (default: 1000, min: 100, max: 999999)
MergeBufferSize	Integer	Maximum number of merge buffer pages (default: 10000, min: 0, max: 999999)
Password	String	Connection password
PortNumber	Integer	Network port number
TotalSortPages	Integer	Total number of memory and disk sort pages (default: 10000, min: 1000, max: 99999999)
UserID	String	Connection userid
VortexDriver	Integer	Genesis - 0 VortexNet - 1

VORTEXjdbc and VORTEXjava

Installing on Unix

1. Create a directory on your machine for the VORTEXjdbc files. For example,

   `/usr2/vortex`

2. Put the downloaded `.tar` file in that directory and untar it.

   The directory now contains two subdirectories and a `vortex.jar` file:

   - ***sample*** — Sample VORTEXjava and VORTEXjdbc programs.
   - ***html*** — HTML VORTEXjava and VORTEXjdbc method descriptions.

3. Move the `vortex.jar` file into a directory pointed to by the Java JVM `$CLASSPATH` environment variable. Alternatively you can add this directory to the `$CLASSPATH` environment variable:

   C Shell

   ```
   setenv CLASSPATH "$CLASSPATH":"/usr2/vortex"
   ```

   Bourne Shell

   ```
   CLASSPATH=$CLASSPATH:/usr2/vortex
   export CLASSPATH
   ```

Chapter 6
Starting VORTEX

VORTEX enables you to connect client applications on a variety of platforms to databases on another variety of platforms. You can connect everything from a Java client to a legacy (flat file) database, and do it easily with VORTEX.

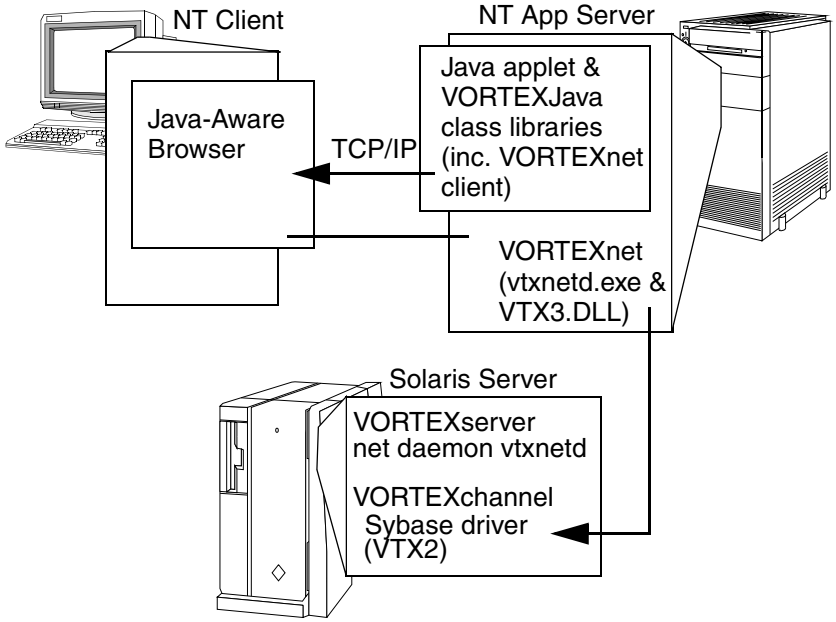
VORTEXnet is the VORTEX component that connects clients and servers that are on different machines. You can have an unlimited number of "hops" through intermediary machines connected by VORTEXnet. The only requirement is that you have VORTEXnet on each of the machines in the path.

VORTEXnet is installed as part of the default options when you install VORTEX. The only time you do not need this component is when the client application and database are on the same physical machine.

You can customize VORTEXnet operations with command-line switches or by setting keywords in the `net.ini` initialization file.

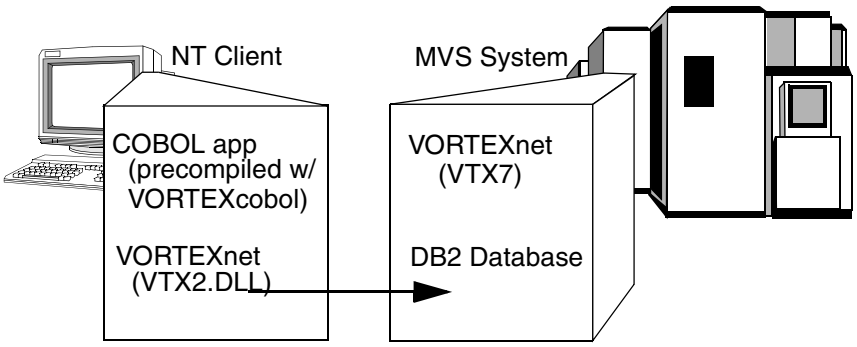
Working Scenarios

Scenario 1



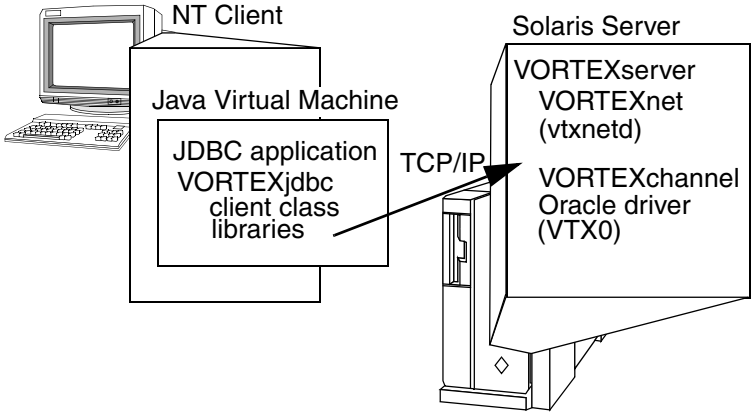
A Windows client running a Java-aware browser (and Java application in that browser) connects through a Windows application server to a database on Solaris. The application server machine is running a Java applet and VORTEXJava class libraries. The Solaris machine hosts a Sybase database, and VORTEX's Sybase driver.

Scenario 2



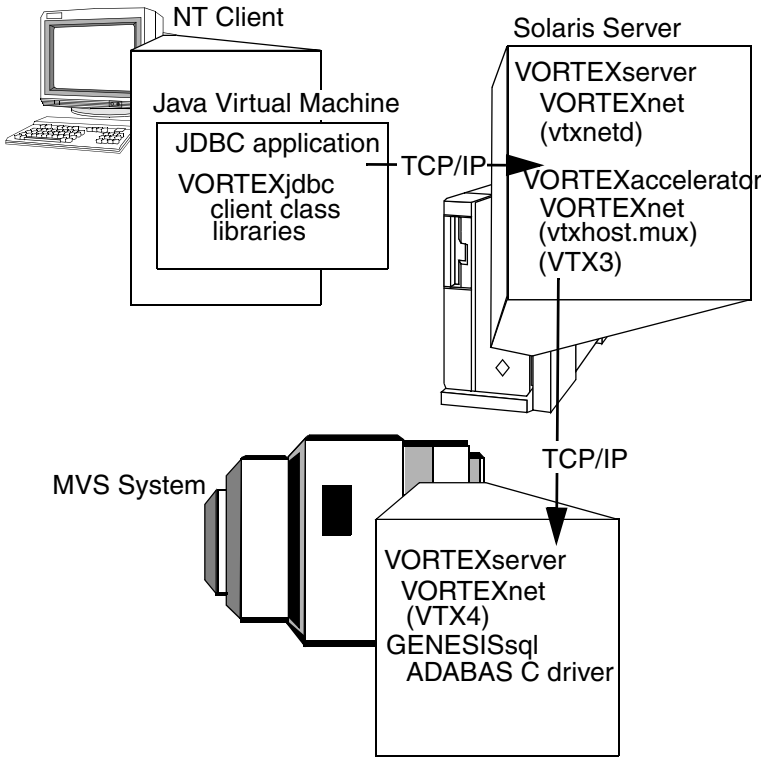
A Windows client running a COBOL application (precompiled with VORTEXcobol) connects to a DB2 database on MVS.

Scenario 3



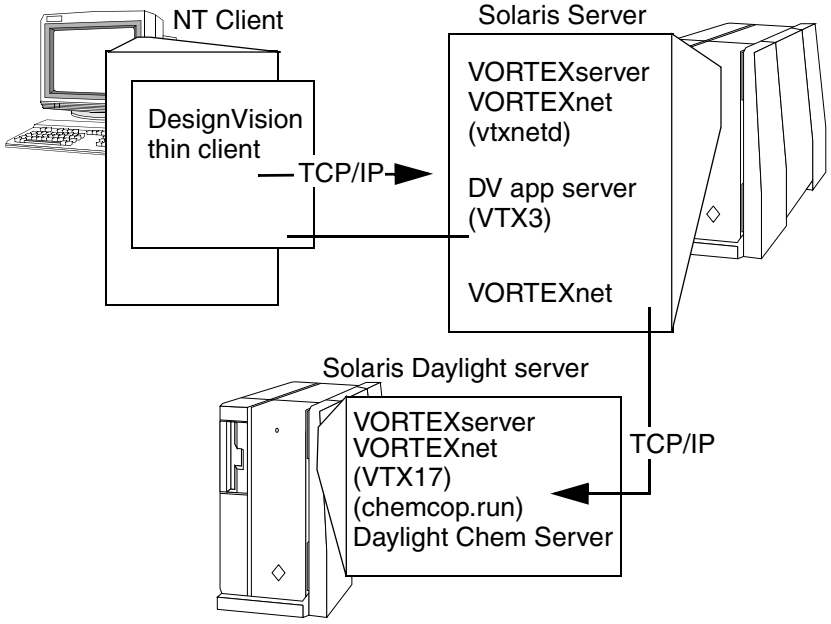
A Windows client running a Java Virtual Machine with JDBC application and VORTEXjdbc connects to Oracle on Solaris.

Scenario 4



A Windows client running a Java Virtual Machine with JDBC application and VORTEXjdbc connects through a Solaris machine running VORTEXaccelerator to ADABAS C on MVS via the GENESISsql database driver.

Scenario 5



Here, a Windows client connects through Solaris to another Solaris machine that runs the chemical-industry specific database, Daylight. The DesignVision thin client can run in a browser.

Operations

VORTEXnet listens on a socket opened on a port, either specified with the `-p` option, for example, `-p1958` on the command line, or read from the `/etc/services` file and might look like:

```
vtxnet  1958/tcp
```

When VORTEXnet receives a connection request, it starts a new process or thread to service the connection and then listens for the next connection request. You pass the name of the program to start in the connection string, as described in "***Connecting Your Application to VORTEX***" on page 46. VORTEXnet is not involved in the client/server connection after this point.

Access Security

You can provide a level of security to the VORTEXnet connection with a switch, `-a`. This option tells VORTEXnet to expect an operating system userid and password in the connect information. VORTEXnet validates the user using the operating system's authorization scheme.

VORTEXnet encrypts the provided password and then compares it to the operating system's encrypted password for the userid. If these match, then `setuid` and `setgid` are used to change the new process's userid and groupid.

Windows

To use this option, the VORTEXnet owner must have `LOGON_AS_BATCH` capability. This is set using the **`Control Panel > Administrative Tools > Local Security Policy > Local Policies > User Rights Assignment`** dialog. Click on ***Logon as Batch*** and make sure that the user has this capability.

If the operating system userid/password information is in the form of domain\userid/password, VORTEXnet uses that domain name for authorization checking. Otherwise VORTEXnet checks the `VORTEX_AUTH_DOMAIN` environment variable for the domain name to use for authentication. If this variable is not set, then `GetComputerName()` is called to get the machine name. Then VORTEXnet attempts to logon the new process or thread using the provided userid, password, and domain.

OpenVMS

VORTEXnet validates the userid and password against the UAI system information. If the password is valid, it calls the *Persona* system services to switch the userid.

MVS

The switch is only valid when the VTXHOST processes are run as Batch jobs. If you're running the VTXHOST processes as Started tasks, then this option has no effect.

SSL OpenSSL

VORTEXnet supports SSL on Windows, Unix, and OpenVMS using OpenSSL.

NOTE: *On Windows, the VTXSSL.dll's Properties->Details->Product name must be "VORTEXnet - OpenSSL Utils". Alternatively if VORTEXnet is started with the 'log'*

option, the logfile will show for example "SSL compile/library: OpenSSL 1.1.1b 26 Feb 2019/OpenSSL 1.1.1b 26 Feb 2019".

Server

Use the `-e` option to start VORTEXnet in SSL mode:

```
-e certificatefile keyfile [protocol[,protocol[,,,]]]
```

The optional protocol(s) specify the permitted TLS levels that VORTEXnet will support and are in the form 1.x. The default is for VORTEXnet to accept all incoming TLS level connections from 1.2 upward.

Client

VORTEXnet clients may specify a certificate file in their `[TRIM_HOME|VORTEX_HOME]/lib/net.ini` file or by using the VORTEXcli `VTXCMD` function with the `TDB_CMD_DRV_CONFIG` option. If the client specifies a certificate file, then it is also searched for a PRIVATE KEY or RSA PRIVATE KEY. If this fails, then the client searches for <certificate file>key.pem. If this is not found or the key is not valid, then an error occurs. A client certificate file is necessary only if the server requests a client certificate.

VORTEXnet does not support pass phrase encoded key files.

SSL SChannel

VORTEXnet on Windows can also use Secure Channel (SChannel) which is part of the Window operating system.

NOTE: *On Windows, the VTXSSL.dll's Properties->Details->Product name must be "VORTEXnet - SChannel Utils". Alternatively if VORTEXnet is started with the 'log' option, the logfile will show "SSL compile/library: SChannel/SChannel".*

Server

Use the `-e` option to start VORTEXnet in SSL mode:

```
-e certificate CA [protocol[,protocol[,,,]]]
```

The optional protocol(s) specify the permitted TLS levels that VORTEXnet will support and are in the form 1.x. The default is for VORTEXnet to accept all incoming TLS level connections from 1.2 upward.

There are multiple methods for specifying the server certificate:

PEM filename A fully qualified filename specification. An error is returned if the file is not found or a valid certificate is not found. The PEM file may include the PRIVATE KEY or RSA PRIVATE KEY. If not, then the file <filename>key.pem is opened in the directory. An error is returned if the key file is not found or there is no valid key in the file.

Store-Name\Path\[Common-Name|Thumbprint]	Any valid Windows Certificate Store name, e.g. LocalMachine, CurrentUser, Users, and any valid path, e.g. MY. Examples: LocalMachine\MY\VortexServer LocalMachine\MY\d7890d96a266ad39d347aa991e17a2875ecf8175
LocalHost	The keyword 'LocalHost' indicates that VORTEXnet server will search in LocalMachine\MY for any valid certificate that has the Server Authentication Usage and an IP address or DNS name that matches the server.

There are two options for the CA specification:

cert_store	The keyword 'cert_store' indicates that the Window Certificate Store CA certificates will be used to validate the client if necessary.
PEM filename	A fully qualified filename specification. An error is returned if the file is not found or a valid certificate is not found.

Client

VORTEXnet clients may specify a certificate in their `[TRIM_HOME|VORTEX_HOME]/lib/net.ini` file or by using the VORTEXcli `VTXCMD` function with the `TDB_CMD_DRV_CONFIG` option. Currently only a Storename\Path\[CommonName | Thumbprint] may be specified.

Limiting Options

You can use VORTEXnet to start up any program or script. To limit permitted options, use the `-f` switch. The file specified after `-f` contains the names of the services that VORTEXnet starts, one per line. VORTEXnet looks for the file exactly as specified; if not found, it then looks in the "lib" directory under the VORTEX_HOME directory environment variable. Lines beginning with "#" are comments. These lines must ***exactly*** match the service name passed in the connect string, including any directory path information. For example, if the file contains the string

```
#old style
/usr/local/bin/VTX0
#new style
service:/usr/local/bin/VTX0
```

then the connect string must have the complete path. Anything that does not match perfectly does not work. The "service:" keyword was added to make the file clearer if preloaded dlls are also specified.

Preloading DLLs

The threaded versison of VORTEXnet can preload DLLs in order to speed up connections. To preload DLLs, use the `-f` switch as described above. The DLLs are specified as

```
preload:VTX4
```

```
preload:VTXIPC
```

Logging Activity

You can create a logfile with the `log`[n] option. A file called `tcm_processidd.log` is created in the directory where VORTEXnet was started. The file contains all VORTEXnet activity and is useful for debugging connection failures. Setting the option to `log` or `log1` logs all VORTEXnet activity whereas `log2` does not log connection or ping requests.

You can also set VORTEXserver to log all authentication attempts by setting the `-aL` option in the command line. On UNIX systems, the messages are sent to the syslogd process. On OpenVMS, the messages are sent to the operator logfile.

Initialization File Settings

Most of the Trifox tools and sub-systems read configuration and initialization data from special `.ini` files. These files typically have the same format:

```
option                  value
```

The *option* is the name of the initialization option, setting name, or parameter. Lines with un-recognized options are ignored.

Value is the value of the option. Depending on the *option, value* can be a number, a yes/no, or a text string. The value can also represent one or more environment variables expressed as:

```
$(name)
```

The environment variable(s) are expanded before the value is evaluated.

The files support text strings as values, but they must be enclosed in double quotes ("), SQL-style, if blanks or quotes are part of the string. If no ending quote mark is provided, the string is terminated with a `\n`.

If an option is not found in the file, then the default value is used.

The various relevant `.ini` files are described in detail in the following section(s).

Edit them using any ascii-based text editor. If you are reinstalling a product, we recommend you edit a ''clean'' copy of each `.ini` file, rather than modifying an existing one from your environment. You must ensure that the processes that use initialization files have read access to them.

If the information in the `.ini` file differs from the command line connect string parameters, the connect string parameters always take precedence.

net.ini

`net.ini` is used by VORTEXnet.

`host` is either a name in `/etc/hosts` or in the `n.n.n.n` format.

`executable` is the full pathname of the program to service the connection.

All the network options, except `packetsize`, can be overridden in the connect string.

connect_timeout

Type number

Default 0 (no timeout)

Description Time (in seconds) to wait for socket connect completion.

Example The following specifies that the client will wait 5 seconds for a socket connect timeout:

```
connect_timeout   5
```

device*n*

Type text

Default none

Description TLI only: Device to use for connection*n*. The value for *n* can be from 0 to 3 (only DesignVision and TRIMtools can use 1 to 3).

Example The following specifies which device to use if not using sockets:

```
device0 /dev/tcp
```

hostenv*n*

Type text

Default none

Description Comma-delimited list of environment variable(s) to set on host for connection *n*. The value for *n* can be from 0 to 3 and only DesignVision or TRIMtools can use 1 to 3.

Example The following specifies that ORACLE_SID is set to A and ORACLE_HOME is set to `/usr4/oracle`:

```
hostenv0 ORACLE_SID= A,ORACLE_HOME=/usr4/oracle
```

hostnamesvc*n*

Type text

Default none

Description Name of host and service to use for connection *n*. The value for *n* can be from 0 to 3 and only DesignVision or TRIMtools can use 1 to 3.

Example The following specifies that the host machine is hawk and the service is `vtxhost.db2`:

```
hostnamesvc hawk!/usr2/trim/bin/vtxhost.db2
```

key_connect

Type number

Default none

Description Connect string masking key (see vtxnetd -k option). Used to encrypt the OS pwd and DBMS connection (from ":" to "@") information.

Example The following specifies a masking value of 12345:

```
key_connect 12345
```

packetsize

Type number

Default 8192

Description Size (in bytes) of the buffer used to aggregate send operations.

Example The following specifies that holding buffer is 1024 bytes long:

```
packetsize 1024
```

port

Type number

Default vtxnet (entry in `/etc/services`)

Description Communication port number.

Example The following specifies that the machine should connect on port 1958:

```
port   1958
```

read_timeout

Type number

Default 0 (no timeout)

Description Time (in seconds) to wait for read completion.

Example The following specifies that the client will wait 60 seconds for a read timeout:

```
read_timeout   60
```

return_errno

Type yes/no

Default no

Description Return the actual operating system error code for a communication error.

Example The following specifies that the client will receive the actual operating system error code for communiction errors, e.g. gethostbyname_r():

```
return_errno   yes
```

ssl

Type yes/no

Default no

Description If yes, then the client requires SSL communication with the server. If no and the server requires SSL, then the client will also use SSL.

Example The following specifies that the client must receive SSL confirmation from vtxnetd:

```
ssl   yes
```

ssl_CAcertstore, ssl_certfile

Type filename

Default System defined CAcertificate store. Environment variables SSL_CERT_DIR and SSL_CERT_FILE can be used to modify the system defined CAcertificate location and filename. Applies only to OpenSSL.

Description The specified CAcertificate trust store will be used to validate the server certificate. The certificates must be in PEM format.

Example The following specifies that the client will use a certificate trust store stored in the crt.txt file:

```
ssl_CAcertstore   /etc/certs/crt.txt
```

ssl_protocol

Type Comma separated list of accepted protocols

Default all

Description The list of accepted TLS protocol levels. The supported levels are 1.1,1.2,1.3. The order is not important.

Example The following specifies that the client requires TLS levels 1.1 or 1.2 from vtxnetd:

```
ssl_protocol   1.1,1.2
```

tcp_keepalive

Type Slash separated keepalive option values

Default 360/1/10

Description Keepalive_time secs/keepalive_interval secs/keepalive_probes cnt.

Example The following specifies keepalive_time 12 minutes, keepalive_interval 9 seconds, and keepalive_probes 11:

```
tcp_keepalive 720/9/11
```

write_timeout

Type number

Default 0 (no timeout)

Description Time (in seconds) to wait for write completion.

Example The following specifies that the client will wait 60 seconds for a write timeout:

```
write_timeout   60
```

Examples

```
rem ---------- TRIM/NET specific
  hostnamesvc0   192.0.2.4(P390/P390)!4
  hostenv0       GENESIS_HOME='TRIFOX
  packetsize     8192                    -- network packet size (send/recv)
  port           1958                    -- port (overrides /etc/services)


#------------- VORTEXnet specifics
hostnamesvc0   tri62!VTX0
hostnamesvc1   tri62!VTX0
#
packetsize      65535                     -- network packet size (send/recv)
port            1958                      -- port (overrides /etc/services)
connect_timeout 15
read_timeout    1440                      -- max time to wait on a socket read
#write_timeout   30                       -- max time to wait on a socket write
#key_connect     1999                     -- connect string masking key
return_errno    no
ssl             no
ssl_certfile    /usr/local/ssl/cert.pem
ssl_protocol    1.2,1.3
```

Command Syntax

vtxnetd [-p<*port*>] [-a*[l]*] [-k*n*] [-f<*file*>] [-v6] [-s*n*] [-tn] [-w*n*] [-e certfile keyfile [protocol]] [log[n]]

Options

-a[l] Activates the operating system authorization checking. This option lets one user start a process and pass ownership to another.

You must pass a valid operating system userid and password as part of the connect string, so the spawned service knows who owns it.

vtxnetd uses the VTXAUTH shared library to authenticate the userid and password. The VTXAUTH shared library has one entry point, VTXAUTH_CHECK, which returns one of three possible values:

0 - Authentication failed
1- Authentication succeded
2 - Do the default authentication.

If VTXAUTH is not found, then default (O/S) authentication is performed.

If the *l* modifier is specified, vtxnetd logs the outcome of the authentication request to the system log. On Unix, this is syslog (auth). On Windows it is the event log. On MVS it is written in the output log (see log below).

-e *cert_file key_file [protocol]* Activates SSL handling. *Cert_file* contains the server certificate and *key_file* contains the server private key. *Protocol* is optional and specifies the supported TLS levels. The supported TLS levels are 1.0,1.1, and 1.2 and are specified as a comma separated list (see ssl_protocol in the net.ini discussion). The current default is 1.0, 1.1, and 1.2. Future releases will remove 1.0 from the default.

-f*list_file* Specifies a file where valid service names are stored. Only those services in the connect string that completely match (including any directory strings) a service in the file can be started on the service.

-k*n* Specifies a masking key value used to decrypt incoming OS and DBMS uid/pwd information. Clients must have the same value in their net.ini file in the key_connect section.

-p*port_number* Assigns the listener port number for clients to request service. If not specified, then vtxnetd looks in the services file for the vtxnet service.

-s*stack_size* Specifies the stack size limit in KB for threads (Windows only)

-t*timeout* Specifies how long VTXNETD waits for the child process to take the socket (MVS only).

-v6 Specifies that the port allows IPV6 connections. Not supported on VMS or OpenServer. Some operating systems support IPV4 and IPV6 connections on the same socket. Please refer to your operating system documentation.

-wseconds Seconds to wait for threaded connections to exit before vtxnetd exits.

log[n] Starts VORTEX logging. It puts a logfile named `tcm_processid.log` in the directory where vtxnetd is started. If n >= 2, then only errors are logged; otherwise errors and connection requests are logged.

Operating System Specifics

Unix

When you connect to a database, a message goes to VORTEXnet (`vtxnetd`) asking it to start up a server program (VTX*n*). You can put the required port number information in `/etc/services`, or you can specify the information at the command line. (The sample program offers examples of how to specify the connection information.)

1. Identify a port number, necessary to start the vtxnetd, either

   in a command line as

   ```
   vtxnetd -p1958 &
   ```

   - OR -

   If you put the **`vtxnet`** entry into `/etc/services`, then you only need to type

   ```
   vtxnetd &
   ```

2. Start the VORTEX listener daemon (vtxnetd).
3. Test the connection by typing

   ```
   vtxping machine_name
   ```

Remember to specify the port number if you have not added the **`vtxnet`** entry to `/etc/services`.

Windows

```
vtxnetd (multi-threaded)
vtxnet2 (single-threaded)
```

OpenVMS

The installation process creates a command file in TRIM_EXEC called *startnet.com*. From the TRIM_EXEC directory, type

```
@startnet
```

The *startnet.com* file sets several parameters that you may want to change for your installation. It uses the *net.com* file to set the VORTEXserver command line parameters.

MVS

VORTEXnet on MVS is started with a batch job that is submitted to the JES Queue for execution. Batch job VTXNETD resides in the TRIFOX Jobs library TRIFOX.JOBS. The parameters are specified on the PARM= parameter to the VTXNETD executable.

The following is a copy of the VTXNETD job:

```
//VTXNETD JOB NET,CLASS=A,MSGCLASS=X,TIME=1440
//DAEMON EXEC PGM=VTXNETD,
// PARM='ENVAR(_CEE_ENVFILE=DD:EV)/-p1958 -t60 -a log'
//STEPLIB  DD DISP=SHR,DSN=TRIFOX.LOAD
//SYSPRINT DD SYSOUT=*
//SYSTERM  DD SYSOUT=*
//SYSRDR DD SYSOUT=(X,INTRDR)
//SYSTCPD  DD DISP=SHR,DSN=TCPIP.TCPIP.DATA     TCP/IP DATA FILE
//EV       DD DISP=SHR,DSN=TRIFOX.LIB(ENVFILE)   ENVIRONMENT VARS
//*
//* THE SUPPORTED DB DRIVERS
//*
//VTX0 DD DISP=SHR,DSN=TRIFOX.JOBS(VTX0) // Oracle driver
//VTX1 DD DISP=SHR,DSN=TRIFOX.JOBS(VTX1) // DVrun for DB2
//VTX2 DD DISP=SHR,DSN=TRIFOX.JOBS(VTX2) // MUX monitor
//VTX3 DD DISP=SHR,DSN=TRIFOX.JOBS(VTX3) // NET driver
//VTX4 DD DISP=SHR,DSN=TRIFOX.JOBS(VTX4) // Genesis driver
//VTX5 DD DISP=SHR,DSN=TRIFOX.JOBS(VTX5) // MUX driver
//VTX6 DD DISP=SHR,DSN=TRIFOX.JOBS(VTX6) // DVrun for MUX
//VTX7 DD DISP=SHR,DSN=TRIFOX.JOBS(VTX7) // DB2 driver
//VTX9 DD DISP=SHR,DSN=TRIFOX.JOBS(VTX9) // ESQ driver
```

Chapter 7 Connecting Your Application to VORTEX

Because of its flexibility, the string that connects VORTEX clients and servers can become long and look somewhat complicated. However, the runtime parameters are distinct and easy to pick out with a little experience.

You can "hop" through several machines in your connection. Each hop is defined in the connect string as simply a piece of the string. VORTEX extracts the information that it requires, from left to right, and passes on the rest of the connect string to the next hop. You can put some of the connect information into a `net.ini` initialization file, as described in "***Initialization File Settings***" on page 38, but if the information in the `.ini` file differs from the connect string parameters, the connect string parameters always take precedence.

If you have a huge number of client applications connecting to a database through VORTEX, you may want to consider evaluating *VORTEXaccelerator*. It's not for every installation, but in the correct situation, it can significantly improve your system performance.

Syntax

```
[driver:]db_connect_string[@network_string]
```

Where the various pieces are described below.

driver (required)

If your system includes more than one database driver or you are using Windows as your client platform, you must specify the driver for the connection.

`driver` can be one of the following:

Driver	Vendor	Executable
oracle	Oracle	VTX0
rdb	Oracle (formerly DEC)	VTX1
sybase	Sybase	VTX2
net	Trifox	VTX3
genesis	Trifox	VTX4
informix	Informix	VTX5

Driver	Vendor	Executable
db2	IBM	VTX7
oledb	Micosoft	VTX8
adabasc	Software AG	VTX9
adabasd	Software AG	VTX10
odbc	Microsoft	VTX11
sqlserver	Microsoft	VTX12[n]. If you are using the native client driver VTX12n.dll, then you must either specify vtx12n as the driver name or copy vtx12n.dll to vtx12.dll in which case you can specify sqlserver as the driver name.
teradata	NCR	VTX13
mysql	MySQL	VTX14
postgresql	PostgreSQL	VTX16
TRIMrpc	Trifox	VTX17
VORTEXaccelerator	Trifox	VTXMUXH (vtxhost.mux on Unix)

db_connect_string (required)

This is the minimum required syntax. This string is passed, un-modified, to the actual database driver. Note that some drivers have multiple connect string options.

Database	Connect String
Oracle	uid/pwd[/tns_name]
Rdb	database_filename
Sybase	uid/pwd/database/server/appname/natlang/charset
Genesis	uid/pwd/DS:DSN (See the GENESIS Users Guide for more information)
Informix	pathname/dbname[@server] uid/pwd/pathname/dbname[@server] uid/pwd/dbname[@server] dbname[@server]
DB2	dbname uid/pwd/dbname

Database	Connect String
DB2 (MVS)	dbname sqlid/dbname The current path is set based on the value of the `VORTEX_DB2_PATH` environment variable, if it is defined.
OLEDB	uid/pwd/initial_catalog/datasource/provider
AdabasC	uid/pwd/conn/server
AdabasD	uid/pwd/dbname/server
ODBC	uid/pwd/DSN
SQL Server	uid/pwd/DSN uid/pwd/database/server/appname/language/keywords/driver If an item has a '\', e.g. server\instance, then the '\' must be escaped, i.e. server\\instance. Keywords are any valid keyword-value pairs separated by ';'. For example, /DSN=testdb;SAVEFILE=test.sav Note that if you are using the Native Client version (VTX12n) and you are not using a DSN and you do not specify "DRIVER=", then the driver will attempt to connect to: ODBC DRIVER 13 FOR SQL SERVER ODBC DRIVER 11 FOR SQL SERVER SQL SERVER NATIVE CLIENT 10.0 SQL SERVER NATIVE CLIENT SQL SERVER NATIVE CLIENT 11.0 in that order.
Teradata	uid/pwd
MySQL	uid/pwd/database/server
PostgreSQL	uid/pwd/dbname/options Options is a string with keyword = value pairs.
TRIMrpc	appname
VORTEXaccelerator	Configuration-dependent

The slash (/) is a placeholder. If you don't need pieces of `db_connect_string` or they are null, simply leave them empty. For example, if your Sybase uid is "sa," the pwd is null, and your database is "master2," the following `db_connect_string` is correct:

```
sa//master2
```

network_string (optional)

VORTEX scans the `connect_string` from back to front when it searches for the ampersand (@). If the `network_string` needs one, you must specify an additional character to provide one for network string. If you need a colon (:) in the `device`, or elsewhere, and you haven't already specified `device`, you must specify an additional ":".

If any of the necessary optional parameters are missing your client applications consult `net.ini`.

The network_string is completed as follows:

```
[port:]host[(uid/pwd)]!service[,envA[,envB[,envC...]]]
```

port (optional) This number overrides the value for vtxnet in `/etc/services`.

host The name (from `/etc/hosts`) or internet address (nn.nn.nn.nn format) of the machine to which the connection is to be made.

uid/pwd (optional) This is the host OS login information. Note that on Windows systems, you can also pass in **domain\uid/pwd**. Do not use "():,@" in your **pwd**.

service Name of the (executable) program on the host that is the server process. The executables are listed in "***driver***".

envA,… (optional) Environment variables to be set on the host. The format is: `name=value`

with no spaces around the "=". Note that on Unix systems, the LD_LIBRARY_PATH env var cannot be modified. LD_LIBRARY_PATH must be set in the shell where vtxnetd is started.

Examples

Connect to a local Oracle database:

```
oracle:scott/tiger
```

Connect to a host Oracle database (uses `net.ini` parameters):

```
net:scott/tiger
```

Connect to a host Rdb database on an OpenVMS server:

```
net:sql$database@88.0.0.12!VTX1
```

Connect to a network host and then to a host Informix database:

```
net:myinf@88.0.0.12!VTX3@@host1!VTX5
```

Remember that you may need @@ to force the first @ to become the network string delimiter. The @@ is passed as @ to the next layer.

Operating System Specifics

Unix

Client Products

Because Unix clients use dynamically linked executables, the ***driver*** parameter is *always* required.

Network Service

The VORTEXserver daemon forks a new process to service the client's request. The service name is typically `VTXn` where *n* is beween 0 and 17. The VORTEXserver starts a `VTXn` which immediately loads a`VTXn.so`. The exception to this rule is when a VORTEXaccelerator connection starts `vtxhost.mux`.

Windows

Client Products

Because Windows clients use DLLs, the ***driver*** parameter is *always* required.

Network Service

The VORTEXserver daemon forks a new process to service the client's request. The service name is typically `VTXn` where *n* is beween 0 and 17. The VORTEXserver starts a `VTXn.exe` which immediately loads a `VTXn.dll`. The exception to this rule is when a VORTEXaccelerator connection starts `vtxhostm.exe`.

VMS

Client Products

Because VMS clients use dynamically linked executables, the ***driver*** parameter is *always* required.

Network Service

The VORTEXserver daemon forks a new process to service the client's request. The service name is typically `VTXn` where *n* is beween 0 and 17. The VORTEXserver starts a `VTXn.exe` which immediately loads a VTX*n*so.exe. The VTX*n*so.exe files are located either with a define/exec logical VTX*n*so or by looking in the default directory, typically SYS$SHARE:VTX*n*so.exe. The exception to this rule is when a VORTEXaccelerator connection starts a vtxhostm.exe.

Client Product Specifics

All Trifox products follow the same connect string methodology. However, implementations do have slight differences.

DesignVision and TRIMpl

You can initiate a database connection in one of two ways with DesignVision:

- The DVrun command line.

- The TRIMpl `connect()` function.

Both options follow the previously described connect string syntax.

VORTEXjava

NOTE: *VORTEXjava is intended for use only with VORTEXgenesis.*

VORTEXjava uses the `connect()` method to initiate a database connection. The five parameters are:

1. hostName — host with optional (uid/pwd/driver:confirgfile)
2. port — portnumber
3. hostProgram —service
4. dbConnectString — db_connect_string
5. envVariables — envA,...

For example, connecting to mywin using port 1958 to access the TRIM List driver:

```
   db.connect("mywin",1958,"VTX4","uid/pwd/
list:list.ini","GENESIS_HOME=/usr2/genesis");
```

If you intend to execute multiple VORTEXserver hops, this information is stored in the `dbConnectString`. Using the above example, mywin is a firewall system and the TRIM List database is on a Unix system called myunix:

```
   db.connect("mywin",1958,"VTX3",
      "uid/pwd/list:list.ini@1958:myunix!VTX4,
      GENESIS_HOME=/usr2/genesis","");
```

This string sends a connection request to mywin on port 1958 telling it to start service VTX3, the VORTEXserver client. VTX3 gets the `dbConnectString` which tells it to send a request to myunix on port 1958 to start the VTX4 service using the GENESIS_HOME environment variable.

VORTEXjdbc

NOTE: *VORTEXjdbc is intended for use only with VORTEXgenesis.*

VORTEXjdbc uses the `DriverManager.getConnection()` method to initiate a database connection. You can specify the connection information in the URL, in the Properties object or the vortex.properties file. The order of precedence is URL, then Properties object, then vortex.properties file. The vortex.properties file must be located in the same directory as the vortex.jar file. To connect to the mywin server as above:

```
   String URL = "jdbc:vortex://uid/pwd/
list:list.ini@1958:mywin!VTX4";
```

Using the Properties object, you would specify the following:

```
   Properties info = new Properties();
   String URL = "jdbc:vortex://";
   info.put("port","1958");
   info.put("host","mywin");
   info.put("service","VTX4");
```

```
info.put("login","uid/pwd/list:list.ini");
```

To embed properties in the URL, specify them after the configuration file:

```
String URL = "jdbc:vortex://uid/pwd/
list:list.ini?key_connect=1234&connect_timeout=42@1958:mywin!VTX4
```

To use mwin as a firewall to access the TRIM list databsae server on myunix as above:

```
String URL = "jdbc:vortex://uid/
   pwd/list:list.ini@1958:mywin!VTX3@1958:myunix!VTX4,
   GENESIS_HOME=/usr2/genesis";
```

Do ***not*** escape the @ as in the VORTEXjava example.

The properties that can be embedded in the URL, in the Properties object or the vortex.properties file are:

Property	Definition	Default
connect_timeout	Connection timeout value in milliseconds.	0 no timeout
db_cursor	Number of actual database cursors.	128
environment_variables (vortex.properties file)	Environment variables to set before starting service.	
envvars (embedded URL, Properties object)	Environment variables to set before starting service.	
fetch_size	Fetch buffer size in bytes.	8192
host	Target hostname or IP address.	
jdbcsql<n>	SQL commands to be executed after connection succeeds. n = 0 - 20 without gaps.	
key_connect	Connection string masking integer. It must match the -k parameter value on VORTEXserver.	

Property	Definition	Default
logfile	Name of the logile on the client. ".log" will be automatically appended.	
logical_cursor	Number of logical cursors.	1024
login	Uid/pwd/ database:configfile.	
logopts	Logging options to apply to logfile. See Chapter 11 "Environment Variables" for options.	
max_column	Maximum number of resultset columns.	256
no_null_string	If YES, return empty string as ""; else null.	NO
password	Server OS authentication pwd.	
port	Port to use to connect to server.	
read_timeout	Read timeout value in milliseconds.	0 no timeout
service	Name of service to start on server.	
socks_hosts	Name of SOCKS host.	
socks_port	Port of SOCKS host.	
trim_ms	If YES, trim milliseconds from timestamps.	NO
user	Server OS authentication uid	

VORTEXperl

VORTEXperl uses the `dbConnect()` method to initiate a database connection. Similar to the VORTEXjava `connect()` method, the five parameters are:

1. hostName — host with optional (uid/pwd)
2. port — portnumber

3. hostProgram — service
4. dbConnectString — db_connect_string
5. envVariables — envA,...

For example, connecting to mywin using port 1958 to access Oracle:

```
dbConnect("mywin",1958,"VTX0","scott/tiger","");
```

To use mynt as a firewall to access the Oracle server on myunix as above:

```
dbConnect("mywin",1958,"VTX3",
   scott/tiger\@1958:myunix!
   VTX0
   ORACLE_HOME=/usr/oracle,ORACLE_SID=A","");
```

Note the backslash (\) used to escape the (@).

Chapter 8
VORTEX Web Services

In previous chapters you have seen how VORTEX enables you to connect client applications on a variety of platforms to databases on another variety of platforms. You can also use VORTEX Web Services to do the same with applications. VORTEX Web Services provide a way to use standard Web servers, eg. Apache or IIS, to connect clients with applications and databases. This eliminates the need for opening additional ports on your firewall. VORTEX requests simply move through the Web server the same as any other data stream.

DesignVision clients

DesignVision is Trifox' application system. It is typically configured with client machines using dvslave to connect to a dvrun process on a server machine through vtxnetd listening on a specific port. This provides a very fast socket-to-socket communication method. However if the server only allows known ports to be used, it is unlikely that vtxnetd can be configured with its owm port. In this case, there are two methods that can be used to support DesignVision clients.

DesignVision JavaScript

DesignVision JavaScript runs the DesignVision client in the browser's JavaScript engine. This method does not require any software installation on the client system. The recommended browser is Microsoft's Internet Explorer. Firefox may work however this is not completely verified. The drawback to using JavaScript is that the application does not have access to anything on the client system such as files or helper programs. While this is acceptable for most designs there are cases where helper applications on the client machine are required.

DesignVision HTTP

Applications that require access to files or programs on the client machine can still use VORTEX Web Services by executing dvslave on the client system. This is identical to the "standard" DesignVision client method, only the initial connect string is different. The advantage is that applications will run exactly the same as they did in the "standard" configuration but using the Web server to manage the connection to the application.

VORTEXnet HTTP

In the same way as you can use an HTTP server to manage DesignVision clients, you can also provide VORTEXnet access through a standard Web server. The only difference seen by the client system is the connect string used to access the database.

Operations

VORTEX Web Services is simple in concept and configuration. In this section, we will assume that it is configured in /cgi-bin/ however you can put it anywhere you like as long as the HTTP server has access that that directory. Although there are slight differences in the connection strings and methods between DesignVision JavaScript, HTTP, and VORTEXnet HTTP, they all function in a similar manner.

VORTEX CGI manager

All methods require the use of a VORTEX CGI index file refered herein as an ilf file. This file contains the mapping between a client and its server process. Every time a request is made from the client to the server, the ilf file is consulted to determine which server process will server the request. The vtxcgi program manages the ilf file.

```
                              VORTEXcgi - CGI utility.
                            Version 2.1.3.2 - Internal.
Copyright © 1989-2006, Trifox, Inc., California, USA.
                          All Rights Reserved Worldwide.


usage: vtxcgi <CGI name> [options]

       options: a[N] Ajax. Clean out dead processes
                     Optionally sleep for N seconds
                cN   Create a group with max N processes
                r    Release (nicely) all run processes
                k    Kill (nuke) all PIDs, MIDs and delete ILF
                l    List all run processes
                p    Ping all run processes
```

The <CGI name> parameter is the name of the ilf file and automatically has .ilf appended to it. You initialize an ilf file by using the cN option where N is the maximum number of client processes that will attach to the service. As we will see, the <CGI name> is sent in the connection URL so you can have many ilf files. Note the a[N] option. Since client processes can timeout or otherwise go away, it is possible for index entries to become invalid. Running vtxcgi <CGI name> a[N] as a background process automates the process of cleaning out invalid entries.

DesignVision JavaScript

To start DesignVision JavaScript, the browser makes the following URL request:

```
http://www.trifox.com/cgi-bin/dvjs?0 0 -idvjs.ini
```

This tells the HTTP server to start up the dvjs program and pass it three parameters. The dvjs program looks for two files, dvjs.ilf and dvjs.env. It reads the values in the dvjs.env file and creates environment variables with these values. A sample dvjs.env file is:

```
#
# DVjs variables
#
```

```
VTXNODE_RUN=./dvjs
VTXNODE_URL=http://www.trifox.com/cgi-bin/dvjs
VTXNODE_TIMEOUT=600
#VTXNODE_LOGFILE=/tmp/cgi.log
#VTXNODE_POPUP=no
#
# System variables
#
TRIM_HOME=/usr3/rad/cheetah
RAD=/usr3/rad
LD_LIBRARY_PATH=/usr2/lib:/usr/local/lib:/usr2/trim/bin
PATH=/usr2/bin
```

The VTXNODE_RUN var points to the name of the program used to actually manage the connection. In this case, dvjs is a bit deceptive as dvjs will append a 1 to $VTXNODE_RUN for Internet Explorer and a 2 for Firefox. So in this case, there will be two programs in /cgi-bin/: dvjs1 and dvjs2. VTXNODE_URL is used to formulate the return URLs for the rest of the session and should match the initial URL used to initiate the connection. The only time this URL would not match the initial URL is if the initial HTTP server redirected the incoming client to another URL. The rest of the vars should be familiar to DesignVision users. Only VTXNODE_RUN and VTXNODE_URL are required.

At this point, a new entry will be made in the dvjs.ilf file and the contents of dvjs.ini (passed in the original URL) are read. Dvjs is similar to dvslave and so this part of the process is the same. You will notice that when control returns to the browser, the URL will have new values in place of the original 0 0. These tell subsequent calls to dvjs to look in the dvjs.ilf file to find the appropriate process to service requests.

You can create many different entry points for DesignVision JavaScript on the same HTTP server. In our example, we used dvjs, dvjs.env, and dvjs.ilf as the file names. However if you copy dvjs to dvjsmax for example and create an ilf file called dvjsmax.ilf using vtxcgi, dvjsmax would look for dvjsmax.env and dvjsmax.ilf. Doing this can help you set up different environments for different users. Also note that you can send env vars on the initial URL. For example,

```
http://www.trifox.com/cgi-bin/dvjs?0 0 -idvjs.ini PGM=demo
```

would set an env var PGM=demo in addition to the env vars found in dvjs.env.

DesignVision HTTP

DesignVision HTTP uses the same dvslave client program as the “standard” DesignVision however the Hostname and Port entries are different. The Hostname entry is a URL to the HTTP server and the Port is (typically) 80 however this is based on how you configured your HTTP server. For example,

```
http://www.trifox.com/cgi-bin/vtxweb
```

tells dvslave to send an HTTP request to the www.trifox.com to open /cgi-bin/vtxweb. Similar to the DesignVision JavaScript case described above, the vtxweb program will look for vtxweb.env and vtxweb.ilf. However in this case, the VTXNODE_RUN env var is set to vtxrlay. Vtxrlay is a VORTEX relay program that manages the requests between the client dvslave and server application process. A sample vtxweb.env file is:

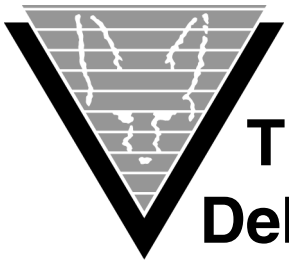
```
#
# vtxweb variables
#
VTXNODE_RUN=./vtxrlay
VTXNODE_INIFILE=dv.ini
VTXNODE_TIMEOUT=10
#VTXNODE_LOGFILE=/tmp/cgi.log
#
# System variables
#
TRIM_HOME=/usr3/rad/cheetah
RAD=/usr3/rad
LD_LIBRARY_PATH=/usr2/lib:/usr/local/lib:/usr4/oracle/product/
10.1.0.3/bin:/usr4/oracle/product/10.1.0.3/lib:/usr2/trim/bin
PATH=/usr2/bin
```

VORTEXnet HTTP

VORTEXnet HTTP is identical to DesignVision HTTP. In the hostname section of your VORTEXnet connect string, use a URL. Using vtxsql as an example,

```
vtxsql /cnet:scott/tiger@80:http:://www.trifox.com/cgi-bin/vtxweb!VTX0
```

will connect vtxsql to an Oracle database on the system hosting www.trifox.com.



Chapter 9 Troubleshooting and Debugging

You can gather information from VORTEX to help you troubleshoot server processes and debug your client applications. You control this information gathering, called "logging," by setting environment variables when you start processes and defining a file name that holds the results of the session's activities.

Setting Up for Logging

To log the data, you must specify the name of the log file and, optionally, the logging level you want. These items are specified through environment variables.

Environment Variables

A set of two environment variables let you gather and write errors and statements that cause errors on both a client machine and a server machine.

Client/Application

On the client the variables are:

- VORTEX_API_LOGOPTS
- VORTEX_API_LOGFILE

Server/Host

The same options are available on the server, as well, with the following environment variables:

- VORTEX_HOST_LOGOPTS
- VORTEX_HOST_LOGFILE

For a complete list of environment variables that you may use with Trifox products, consult "***Environment Variables***" on page 73.

Specifying Log Levels

To specify the logging level, you use the LOGOPTS variable for either the client or server with the option that represents the level that you want. You can create combinations of levels by using a plus (+) sign, although some combinations, ERROR + FULL, for example, don't make much sense. RECORD and PLAY, naturally, are mutually exclusive. Please refer to "***Environment Variables***" on page 73 for more information on setting the LOGOPTS variable.

Default

If you don't set VORTEX_*_LOGOPTS, then VORTEX automatically performs abridged logging. Abridged logging, more space-efficient, records selected items rather than the complete control structure.

Naming the Log File

The log file for FULL, TIME, or Abridged logging is `VORTEX_API_LOGFILE.log` on the client and `VORTEX_HOST_LOGFILE_`*`pid`*`.log` on the host, where *pid* is the host process id.

The RECORD and PLAY data files are `VORTEX_API_LOGFILE.vdf` on the client and `VORTEX_HOST_LOGFILE_`*`pid`*`.vdf` on the host.

Log files are created in the local directory that is current when VORTEX starts. If you want your log file to write to another location, you must specify a fully-qualified file name.

Examples

To create two files, `test.log` and `test.vdf` with FULL logging and RECORD on a Unix client at runtime, type:

```
setenv VORTEX_API_LOGFILE test
setenv VORTEX_API_LOGOPTS FULL+RECORD
```

Chapter 10 Codes & Messages

This chapter contains error codes and message for Trifox products. We use mnemonic codes instead of numeric error codes. Each section lists a product's errors alphabetically by mnemonic.

Some error messages, such as NOMEM (out of memory) appear in several sections.

VORTEX Error Messages

Messages are listed in alphabetical order.

BADCONV **Data conversion failed (hostvar: number)**

Cause: The requested data conversion failed.

Action: Check that the requested data is of the appropriate type. For example, this error occurs when a character column is fetched into an integer and the character data are not all digits.

BADSQLDA **SQLDA is invalid**

Cause: The SQLDA structure has not been initialized correctly.

Action: Check that the first three fields are set as required.

BLOBCOL **Invalid BLOB column ID**

Cause: The specified column is not a BLOB column.

Action: Verify that the specified column is a BLOB/CLOB column.

BLOBFILE **BLOB file operation *name* failed**

Cause: Several file operations can fail with relational databases that keep BLOB data in external files. The most common failed operation is a write.

Action: Verify that the process owner has write permission in the directory.

BLOBLEN **BLOB length mismatch**

Cause: The length of the BLOB does not match the length specified in an earlier call.

Action: Verify that the initial BLOB/CLOB length matches the length sent in the `VTXBLOB()` call.

BLOBPROC **Cannot return BLOB data via a stored procedure**

Cause: Some relational databases cannot return BLOB/CLOB data via stored procedure calls.

Action: Use BLOB specific functions to return BLOB data.

DLLENTRY **Could not find DB driver entry point *name* (handle: *number*)**

Cause: The loaded DLL OR SHARED LIBRARY does not contain the expected entry point. This error only occurs on machines that support DLLs or shared libraries and usually signals that the wrong DLL has been loaded.

Action: Verify that the VORTEX DLLs have not been overwritten by other DLLs and that there are no other DLLs in the path with the same name.

Finally, make sure that any resources that the DLL needs are available. You can determine which resources are required for a shared library on most Unix systems with the `ldd` command. For Windows systems, you must use third party tools, such as ScanBin, to perform the same function.

DLLLOAD **Could not load *filename* (errno: *number*)**

Cause: Could not load the specified DLL or shared library. The DLL is either missing or invalid. This error only occurs on machines that support DLLs.

Action: Ensure that the DLL or shared library specification is correct (including spelling) and check that the correct DLL is installed.

Finally, make sure that any resources that the DLL needs are available. You can determine which resources are required for a shared library on most Unix systems with the `ldd` command. For Windows systems, you must use third party tools, such as ScanBin, to perform the same function.

DRVCMDI **Expected an integer parameter**

Cause: The driver COMMAND expects the command line's first token to be an integer.

Action: Check the parameters to the `VTXCMD()` function call and make sure that the first token is an integer.

DRVCMDP **Invalid parameter**

Cause: A parameter to the driver COMMAND is invalid.

Action: Check the parameters to the `VTXCMD()` function call.

DRVCONF **Driver not configured**

Cause: The database driver has not been configured.

Action: Ensure that the first call is the CONFIG call.

DRVMULTI **Driver must be specified when multiple are present**

Cause: Multiple drivers are linked and the target database driver name is not specified in the connect string. Review connect string specifications in the *Trifox Resource Manual* for details.

Action: Verify that the connect string has the driver name. If only one driver is present in the program file, then the driver name is optional.

DRVNOTF **Driver *name* not found**

Cause: The driver specified in the connect string cannot be found.

Action: Verify that the driver in the connect string is correct or that the driver exists in the program file

FETCHOVER Fetch buffer overflow

Cause: The fetch buffer used in VORTEXaccelerator has a fixed maximum size in shared memory. VORTEXaccelerator's fetch buffer has been exceeded. This assertion error only occurs if VORTEXaccelerator is being used.

Action: Increase the fetch buffer parameter (bs) on the VORTEXaccelerator command line

FETCURCLO Attempting a FETCH from a closed cursor (*name number*)

Cause: The cursor being used for the FETCH has previously been closed.

Action: Check your program logic to make sure that it does not attempt to use a closed cursor.

FLIPOVER Flip buffer overflow

Cause: This assertion error occurs if too many parameters are specified. The current limit is approximately 250 parameters. Note that multiple dimensions are not included in this limit.

Action: Notify Trifox support.

INVCUR Invalid cursor

Cause: The cursor has not been initialized.

Action: Set the cursor to -1 before the first call and do not modify it after subsequent calls.

INVCURPOS Invalid cursor for positioned EXEC

Cause: The VTXEXEC() cursor is not valid.

Action: Ensure that the cursor is valid from a previous VTXOPEN() call and that the cursors was opened using the TDB_PRE_POPEN option.

INVDATE Invalid date/time

Cause: The format of the date and/or time data is invalid.

Action: Verify the date and/or time data format or the format mask being used.

INVDRVVER DB version mismatch (expected: *name*, found *number*)

Cause: The version of the database driver is not at the same level as the VORTEX runtime library. This error is most common when VORTEXclient/server is being used, but can also occur if an older driver has been linked with a newer runtime library.

Action: Ensure that the database driver is the correct version.

INVNUM Invalid (internal) number

Cause: The data being converted is invalid.

Action: Check the data being converted.

INVPREVER Invalid precompiler version

Cause: The version of the precompiler that generated the VORTEXcli calls is not at the same level as the VORTEXcli runtime library.

Action: Use the same version level for both products.

INVUPD **Invalid UPDATE statement**

Cause: Invalid UPDATE statement for BLOB processing. Sybase-specific error.

Action: Verify the syntax of the UPDATE statement.

MANYBIND **Too many bind(host) variables**

Cause: Too many bind variables are specified for a particular stored procedure or prepared statement.

Action: Verify that the number of bind variables matches the number of bind variable positions in the statement.

MANYCCUR **Too many concurrently open cursors (*max: number*)**

Cause: There are too many database cursors open.

Action: Either configure VORTEX to allocate more database cursors or close any cursors that you do not need.

MANYCOLS **Too many columns (*number*) returned by query**

Cause: The query returns more columns than have been allocated.

Action: Modify your query or allocate more columns when you initialize VORTEX in `gui.ini` (DVtools), the `VTXINIT()` call (VORTEXcli), the `dbChannel()` call (VORTEXjava, VORTEXperl), or the `connect()` call (VORTEXjdbc).

MANYCONN **Too many connections**

Cause: This error indicates that the sixty-four concurrent connection limit has been exceeded.

Action: Reduce the number of connections used.

MANYLCUR **Too many logical cursors**

Cause: Too many logical cursors have been requested.

Action: Allocate more logical cursors in `gui.ini` (DVtools), the `VTXINIT()` call (VORTEXcli), the `dbChannel()` call (VORTEXjava, VORTEXperl), or the `connect()` call (VORTEXjdbc).

NOCONN **Not connected**

Cause: A connect must be performed before any other operations.

Action: Connect to the desired relational database.

NODRV **No DB driver linked**

Cause: No database driver is available for the requested connection.

Action: Check that your connect string is correct. If you have built the product from VORTEX libraries make sure that the link sequence is correct.

NOMEM **Out of memory**

Cause: A memory allocation failed. Either there is no more heap memory available (rare) or the heaps have been corrupted.

Action: This is a fatal error. Notify your system administrator immediately.

ORAOOPT **oopt() requires two integer parameters**

Cause: Oracle's `oopt()` function requires two integer parameters.

Action: Review Oracle's OCI documents for more details.

PIGFILE **'Piggy-back': *operation name***

Cause: 'Piggy-back' file operation failed. The error message indicates more information. Typically the filename being used in a TRIMpl function like `file_copy()` is incorrect.

Action: Use the correct filename

POSBROW **Position EXEC requires a 'for browse' cursor**

Cause: A positioned EXECute (UPDATE or DELETE) requires a previously opened and positioned cursor. Only occurs in Sybase and Microsoft's SQL Server.

Action: Open the cursor in "for browse" mode.

POSEXEC **Position EXEC requires an open cursor**

Cause: A positioned EXECute (UPDATE or DELETE) requires a previously opened and positioned cursor.

Action: Verify the program logic.

UNDESDTY **Unknown DESCRIBEd datatype (*name number*)**

Cause: The datatype of a described datatype is unknown. This error may occur if a relational database has introduced a new datatype and an older VORTEX driver is being used.

Action: Notify Trifox and request an enhancement.

UNSUPFNC **Unsupported function (*FNC: number*)**

Cause: An unsupported database driver function has been encountered.

Action: Notify Trifox.

ZEROCOLS **ZERO columns returned by query**

Cause: A query has return zero columns. This error occurs when the query is not a SELECT statement.

Action: Use a SELECT statement.

VORTEXodbc Error Messages

Messages are listed in alphabetical order.

BADCONV **Data conversion failed (hostvar: number)**

Cause: The requested data conversion failed.

Action: Check that the requested data is of the appropriate type. For example, this error occurs when a character column is fetched into an integer and the character data are not all digits.

BADINI ***filename* is either missing or invalid**

Cause: The `.ini` file is missing or its contents are invalid.

Action: Check that the file exists and that it is correct.

CANCEL **Operation cancelled**

Cause: Operation was cancelled by the driver manager.

Action: Informational message; no action necessary.

CANFREE **Cancel treated as SQLFreeStmt with SQL_CLOSE**

Cause: No processing was being done on the statement, so the call was treated as a call to SQLFreeStmt with the SQL_CLOSE option. Function returns SQL_SUCCESS_WITH_INFO.

Action: Informational message; no action is necessary.

CURDUP **Duplicate cursor name**

Cause: The cursor name is already in use.

Action: Specify a different name.

DATATRUNC **Data truncated**

Cause: Data has been truncated. Either the data specified is too long or supplied output buffers are too small.

Action: Modify the size of the output buffers.

DIALOG **Dialog box failed**

Cause: This assertion error indicates that the connect dialog failed.

Action: Notify your system administrator.

FUNCSEQ **Function sequence error**

Cause: The sequence of functions called is invalid.

Action: Make sure that you follow the sequence specified by the ODBC documentation.

GENONLY **This version only supports GENESIS**

Cause: This version of VORTEXodbc only supports GENESIS.

Action: Please contact Trifox, Inc. for availability of other drivers.

INVARG **Invalid argument**

Cause: Invalid arguments specified.

Action: Consult the ODBC documentation for the correct syntax.

INVAUTH **Invalid authorization**

Cause: User not authorized to connect to specified Data Source.

Action: Check your userid and password or contact your system administrator.

INVBUFLEN **Invalid string or buffer length**

Cause: The length specified is invalid. Negative values, such as SQL_NTS, have special meaning but not all negative values are valid.

Action: Check the ODBC documentation for valid length specifiers.

INVCOLNUM **Invalid column number**

Cause: The specified column number is out of range.

Action: Verify that the correct column number is being used.

INVCURNAM **Invalid cursor name**

Cause: The specified cursor name is invalid.

Action: See the ODBC documentation for the maximum allowed length.

INVCURSTA **Invalid cursor state**

Cause: The state of the cursor (OPENed, CLOSEd, etc.) is not valid for the current operation.

Action: Make sure you've executed the necessary steps before calling this function.

INVDATA **Invalid data**

Cause: The data for the specified operation is invalid.

Action: Verify that the data being used is correct.

MANYCONN **Too many connections**

Cause: The concurrent connections limit has been exceeded.

Action: Reduce the number of concurrent connections.

MANYSTMT **Too many statements**

Cause: The number of allowable statements has been exceeded.

Action: Either increase the limit in the Setup Dialog or free any statements you are not using or do not need.

MISSENV **Missing environment variable**

Cause: The environment variable VORTEX_HOME is missing.

Action: Set the VORTEX_HOME environment variable.

MISSINDV **Data is NULL, but no indicator variable supplied**

Cause: NULL data supplied without an indicator variable. If the data is NULL then an address of an indicator variable must be supplied.

Action: Supply an address for the indicator variable.

NOCONN **Not connected**

Cause: A connect must be performed before any other operations.

Action: Connect to the data source before attempting any database operations.

NOCURNAM **No cursor name available**

Cause: Cursor name was never assigned.

Action: Assign a cursor name.

NODRV **VORTEX driver not specified**

Cause: No VORTEX driver was specified in the connect.

Action: Verify that the connect string specifies a VORTEX driver.

NODSN **No DSN specified**

Cause: The Data Source Name (DSN) was not specified.

Action: Verify that the connect string specifies the DSN.

NOMEM **Out of memory**

Cause: A memory allocation failed. This is a fatal error. Either there is no more heap memory available (rare) or the heaps have been corrupted.

Action: Notify your system administrator immediately.

NOTCAP **Driver not capable**

Cause: VORTEXodbc does not support requested capability.

Action: Modify your program to not request this capability.

NOTIMP ***feature_name*** **not implemented**

Cause: The feature has not been implemented yet.

Action: Contact your vendor.

NOWHDL **No window handle available**

Cause: No window handle available to open connect dialog.

Action: Notify your system administrator

OPTCHG **Option value changed**

Cause: This is an informational message. A value of an option has been changed.

Action: No action required.

PARMCNT **Wrong number of parameters**

Cause: The number of parameters specified does not match the number of parameters required by the statement.

Action: Modify your program to use the correct number of parameters for the statement.

PARMDTY **Invalid parameter data type**

Cause: The data type specifed for the parameter is unknown.

Action: Consult the ODBC documentation for valid data types.

PARMNUM **Invalid parameter number**

Cause: The parameter number specified is out of range.

Action: Verify that your program uses the correct parameter number.

TIDUSED **Statement already in use**

Cause: Another thread is currently using the statement.

Action: Verify that your program does not use the same statement as is being used in another thread.

UNDESTYP **Unknown descriptor type**

Cause: The fDescType for SQLColAttributes() is unknown.

Action: Modify your program to use the correct descriptor.

UNFETTYP **Unknown fetch type**

Cause: Currently only SQL_FETCH_NEXT is supported.

Action: Modify your program to use only SQL_FETCH_NEXT.

UNINTYP **Unsupported InfoType: *type***

Cause: The Info Type is unsupported.

Action: Consult the ODBC documentation for valid values.

UNOPT **Unknown option**

Cause: The option is unknown.

Action: Consult the ODBC documentation for valid options.

UNUNOPT **Unknown Uniqueness option**

Cause: The Uniqueness option is unknown.

Action: Consult the ODBC documentation for valid values.

UNXACOPR **Unknown transaction operation**

Cause: The transaction operation is unknown.

Action: Consult the ODBC documentation for valid values.

VORTEXnet Error Messages

All messages are listed in alphabetical order.

AUTHBAD **Invalid authentication syntax**

Cause: The authentication syntax is bad.

Action: Follow the host name with the userid/password.

AUTHFAIL **Authentication on *service* failed**

Cause: You are not authorized to execute the requested host service.

Action: Verify that the userid/password are correct or contact your system administrator.

AUTHREQ **Host '*name*' requires authentication**

Cause: The host you are connecting to requires additional authentication.

Action: Follow the host name with the userid/password.

BADINI **'net.ini' file is either missing or invalid**

Cause: The `net.ini` file is missing or is an invalid file.

Action: Verify that `net.ini` exists and is valid.

CONFIG **Expected a CONFIG call**

Cause: This assertion error notifies you that the first call must always be a CONFIG.

Action: Make sure a CONFIG call precedes any other call.

DLLENTRY **Could not find DB driver entry point**

Cause: The loaded DLL OR SHARED LIBRARY does not contain the expected entry point. This error only occurs on machines that support DLLs or shared libraries and usually signals that the wrong DLL has been loaded.

Action: Verify that the VORTEX DLLs have not been overwritten by other DLLs or that there are not other DLLs in the path with the same name.

DLLLOAD **Could not load DLL**

Cause: Could not load the specified DLL or shared library. The DLL is either missing or invalid. This error only occurs on machines that support DLLs.

Action: Ensure that the DLL or shared library specification is correct (including spelling) and check that the correct DLL is installed.

DLLSAFE **Loaded DLL is not thread safe**

Cause: Not all database drivers are thread safe.

Action: To avoid this error run the single threaded daemon: `vtxnet2.exe`.

EVALCORR **Evaluation ID is corrupt**

Cause: The evaluation license ID has been corrupted. This could indicate that someone has attempted to defeat the evaluation software.

Action: Download the evaluation software from the web site again or contact Trifox to purchase the software.

EVALEXP **Evaluation version of the software has expired**

Cause: The evaluation license has expired. The timeout period is 4 hours.

Action: To continue the evaluation, restart the VORTEX daemon. If you want a copy of the software that doesn't expire, contact Trifox to purchase the software.

EXECFAIL **Exec *name* failed on host *name***

Cause: The service (program) specified in the network connection string could not be started. This error occurs if the service could not be found, does not have the correct permissions, or is not listed as a valid service.

Action: Check that the service exists, is listed as a valid service, and that you are connecting with the correct user name and password.

HOSTNOTFOUND **Host *name* not found**

Cause: The host you are trying to connect to is not found.

Action: Make sure the spelling of the host is correct and/or that it really exists.

INVHOSTSYN **Invalid host/service name syntax**

Cause: The host/service syntax is bad.

Action: Review the connect string documentation in the *Trifox Resource Manual*.

INVVER **NET version mismatch (host: *number*, client: *number*)**

Cause: The version of VORTEXserver is not at the same level as VORTEXclient.

Action: Make sure that both sides of the network connection are at the same version level.

KEEPALIVE **Setting SO_KEEPALIVE failed**

Cause: This assertion error indicates that the socket option KEEPALIVE failed or was not set.

Action: Notify Trifox support or your system administrator.

LINGER **Setting SO_LINGER failed**

Cause: This assertion error indicates that the socket option LINGER failed or was not set.

Action: Notify Trifox support or your system administrator.

NOINTR **Host cannot be interrupted**

Cause: Your program requested a cancel operation. The host you are trying to cancel cannot handle interrupts.

Action: Modify your program so that it does not call the cancel operation.

NOMEM **Out of memory**

Cause: A memory allocation failed. This is a fatal error. Either there is no more heap memory available (rare) or the heaps have been corrupted.

Action: Notify your system administrator immediately.

NUMPARM **Wrong number of parameters**

Cause: The JCL parameters are incorrect (MVS (OS390) only). This assertion error normally means that the JCL is out of sync with the driver.

Action: Make sure you are using the correct versions.

SERVNOTFOUND **Service/Protocol *name* not found**

Cause: The service and/or protocol cannot be found.

Action: Ensure that `vtxnet` is specified in `/etc/services`. If it is not, you must add it, or explicitly specify the port number in the connect string.

SOCKET **Socket() failed**

Cause: Call to the `socket()` function failed. The operating system may have run out of file descriptors.

Action: Notify your system administrator.

TAKESOCK **takesocket() failed**

Cause: The takesocket operation (receiving a socket from a different address space failed) failed (MVS (OS390) only).

Action: Notify your system administrator.

UNDBID **Unknown Database ID**

Cause: Unknown database ID specified in `net.ini`.

Action: Use a valid database ID (from 0 to 3) in `net.ini`

Chapter 11
Environment Variables

Setting environment variables is an operating system-specific task. If you are responsible for setting up your environment and installing Trifox products, but are not familiar with the procedures for your operating system, consult the operating system manuals.

Note that several of the environment variables point to the same objects. For example, both `TRIM_MUX_NAME` and `VORTEX_MUX_NAME` contain the VORTEXaccelerator shared memory identifier. In all such cases, the DesignVision environment variable is checked last. In the previous example, `VORTEX_MUX_NAME` is searched for first and if it is not found, then `TRIM_MUX_NAME` is checked.

Name	Products	Description
`DV_CONFIRM_FILE_ERROR`	DV	Display an error message when a file error occurs. The default is to not display an error. Set to any value.
`DV_PREFIX`	DV	Replaces the default "**dv**" prefix with the value of DV_PREFIX. This affects the .kma, .img, .ini, .xaml files. If DV_PREFIX<.filename> is not found, then DV uses the default "**dv**" prefix.
`DV_TRANSPARENT_COLOR`	DV	RGB color used to make button icons transparent. For example, DV_TRANSPARENT_COLOR=255,0,255 uses magenta as the background to make buttons appear transparent.
`EDITOR`	DV/ TRIM	Name of editor program to use in DVapp and DVreport designers, as well as the TRIM equivalents.
`GENESIS_HOME`	GENESIS	Locator for the GENESIS DS description files.
`TRIM_HOME`	DV/ TRIM	Locator for lib, term, and qmr directories.
`TRIM_MUX_NAME`	DV/ TRIM	VORTEXaccelerator shared memory identifier.

Name	Products	Description
TRIM_SHM_ADDR	DV/ TRIM	Name of the shared memory preferred address file. This variable only applies to operating systems that allow a shared memory segment to be mapped to different addresses. Specifying this variable ensures (for those operating systems) that vtxmux clients can locate the correct address.
TRIM_SHM_BASE	DV/ TRIM	Defines a base address for shared memory loading. TRIM_SHM_ADDR takes precedence over TRIM_SHM_BASE if the object name is in TRIM_SHM_ADDR. Value is an address, either 8 or 16 digits and in the correct format for the underlying architecture. For example, on x86 systems, 00000008 means address 80000000.
TRIM_SHM_FILE	DV/ TRIM	Name of the shared memory description file.
VORTEX_API_LOGFILE	VORTEX	Name of the file to use for VORTEX logging.
VORTEX_API_LOGOPTS	VORTEX	Keyword that specifies logging options for VORTEX: • APPEND specifies that logging will append to an existing logfile. The default is to overwite. • FLIP specifies that PLAY must byte flip integers in the playback file. • FULL specifies detailed VORTEX logging. • MULTI specifies unique log filenames. • PLAY specifies data playback. • RECORD specifies data recording. • SQL specifies SQL file creation. • TIME specifies time to complete each call.
VORTEX_CCMAP_FILE	VORTEX	Name of the file to use to translate ASCII to EBCDIC and back again. This is set on the server system. The file contains 512 blank separated hexadecimal bytes, e.g. 0x7B 0x41, that define the ASCII to EBCDIC mappings.
VORTEX_DDT_MASK	DV/ TRIM/ VORTEX	The date/time format to use for formatting datetime data into strings and for interpretting strings for conversion into datetime data. The default is “DD-MON-RR”.
VORTEX_HOME	VORTEX	Locator for client lib directory.

Name	Products	Description
VORTEX_HOST_HIDEGPF	VORTEX	If false, then SEM_FAILCRITICALERRORS and SEM_NOOPENFILEERRORBOX are set. If true, then the above plus SEM_NOGPFAULTERRORBOX are set. See the VORTEX_HOST_NOSEM env variable.
VORTEX_HOST_LOGFILE	VORTEX	Name of the file to use for VORTEX logging on the server.
VORTEX_HOST_LOGOPTS	VORTEX	Keyword that specifies logging options for VORTEX on the server: • FULL specifies details VORTEX logging. • MULTI speifies unique log filenames. • PLAY specifies data playback. • RECORD specifies data recording. • SQL specifies SQL file creation. • TIME specifies time to complete each call.
VORTEX_HOST_NOSEM	VORTEX	If set to true, then no special error handling is done in the VORTEX drivers. If false, then see the VORTEX_HOST_HIDEGPF env variable.
VORTEX_HOST_SYSLOG	VORTEX	If set to true, then VORTEX host error messages are sent to the system log.
VORTEX_MUX_NAME	VORTEX (Mux)	VORTEXaccelerator shared memory identifier.
VORTEX_ODBC_CHAR	VORTEX ODBC	Set the ODBC datatype (integer) to be returned for a described char column. The default is SQL_CHAR. (1)
VORTEX_ODBC_DATETIME	VORTEX ODBC	Set the ODBC datatype (integer) to be returned for a described timestamp column. The default is SQL_TIMESTAMP (11).
VORTEX_ODBC_NUMBER	VORTEX ODBC	Set the ODBC datatype (integer) to be returned for a described numeric column. If datatype < 0, then VORTEXodbc maps the ODBC datatype based on precision and scale.
VORTEX_ODBC_TIME	VORTEX ODBC	Set the ODBC datatype (integer) to be returned for a described time column. The default is SQL_TIME (10).
VORTEX_ORACLE_FOOLISH	VORTEX	Allows NULL values to be bound for SELECT statement parameters
VORTEX_ORACLE_TS_CLEN	VORTEX	Sets the default TIMESTAMP character string length

Name	Products	Description
VORTEX_SERVICE_FILE	VORTEX	Specifies a file that lists NT services to start as well as environment variables for that operating system.
VORTEX_SHM_ADDR	VORTEX (Mux)	Name of the shared memory preferred address file. This variable only applies to operating systems that allow a shared memory segment to be mapped to different addresses. Specifying this variable ensures (for those operating systems) that vtxmux clients can locate the correct address.
VORTEX_SHM_BASE	VORTEX /DV	Defines a base address for shared memory loading. VORTEX_SHM_ADDR takes precedence over VORTEX_SHM_BASE if the object name is in VORTEX_SHM_ADDR. Value is an address, either 8 or 16 digits and in the correct format for the underlying architecture. For example, on x86 systems, 00000008 means address 80000000.
VORTEX_SHM_FILE	VORTEX (Mux)/ DV	Name of the shared memory description file.

Chapter 12
Format Masks

Trifox format masks let you specify how you want string data to appear. Trifox format mask are modeled after those used by Oracle databases.

char

The *char* datatype has only one format mask: `An` where *n* specifies the width.

numeric

Numeric data has a wide selection of formatting options. You can specify the width of a field or resulting conversions by the length of the mask.

Character	Description
%	Percent sign at right of number.
$	Dollar sign at left of number.
B	Display zero as blank.
0	Display leading zeros.
9	A digit position.
other	Delimiting character (not leading)

Formatting Examples

Value	Mask String	Result
1958	9,999.99	1,958.00
1958	099999	001958
56.789	$999.99	$56.79
0	999.99	0.00
0	099.99	000.00
0	B99.99	
4083692300	999/999-9999	408/369-2300

datetime

Datetime data has a variety of formatting options. You specify the width of the field or resulting conversions by the length of the mask.

NOTE: *Using the two-digit year may affect the Year 2000 Compliance of your application. Trifox recommends using the four-digit year.*

Sequence	Description	Use (I/O)
YYYY	Four-digit year	I/O
YY	Two-digit year (may affect Y2K compliance)	I/O
RR	Two-digit year in another century.	O
MM	Two-digit month of year (01 – 12)	I/O
MON	Three-character month (all upper case)	I/O
mon	Three-character month (all lower case)	I/O
Mon	Three-character month (first letter upper case)	I/O
MONTH	Fully named month (all upper case)	I/O
month	Fully named month (all lower case)	I/O
Month	Fully named month (mixed case)	I/O
RM	Roman numeral month (I - XII)	I/O
DDD	Three digit day of year (001 – 366)	I/O
DD	Two digit day of month (01 – 31)	I/O
D	Single-digit day of week (1 – 7)	O
DY	Three character day (all uppercase)	I/O

Sequence	Description	Use (I/O)
dy	Three character day (all lower case)	I/O
Dy	Three-character day (1st letter upper case)	I/O
DAY	Fully-named day (all upper case)	I/O
day	Fully-named day (all lower case)	I/O
Day	Fully-named day (mixed case)	I/O
AM, A.M.	Hour is AM	I/O
HH12	Two-digit hour (00 – 11)	I/O
HH,HH24	Two-digit hour (00 – 23)	I/O
PM, P.M.	Hour is PM	I./O
MI	Two-digit minutes (00 – 59)	I/O
SS	Two-digit seconds (00 – 59)	I/O
SSSSS	Seconds past midnight (0000 – 86399)	I/O
UUUUUU	Up to six digits of fractional second precision	I/O
J	Julian day	I/O
Q	Single-digit quarter of year (0 – 4)	O
W	Single-digit week of month (1 – 4)	O
WW	Two-digit week of year (01 – 52)	I/O
other	Delimiting character	I/O

Appending *th* (case insensitive) to any uppercase digit mask appends *ST*, *ND*, *RD*, or *TH* depending on the last digit. For lowercase digit masks the lowercase version is appended.

Put embedding characters that are valid masks inside double quotes (").

The default mask is **DD-MON-RR**.

Formatting Examples

Value	Mask String	Result
Feb 6, 1958	DD/MM/YYYY	06/02/1958
Feb 6, 1958	qth “quarter” of YY	1st quarter of 58
Nov 1, 1995 20:48:46	HH12:MI on Day	08:48 on Wednesday

User-Defined Masks

You can use a special format mask when built-in format masks don't meet your need. For example, when the database represents YES/NO as 1 and 0, none of the built-in masks adequately translate the values.

A user-defined format mask looks like

```
Unnfunc ([ ( parm[2] [, …, parm[n] ] ) ])
```

where

nn represents the width of the field in characters. The screen painter represents this mask with the character "u," as in UUUU.

func is the user-defined trigger responsible for the input and output from and to the field. func() has two implicit parameters:
- parm[0] false on output, true on input
- parm[1] output: the actual field; input: the string entered..

parm[2, 3, …, *n*] represents the optional parameters you can pass with the user-defined trigger.

The user-defined trigger is called every time the system reads or writes to the _D variable associated with the field variable and must return the following;

- ***output*** — any data.
- ***input (query)*** — any data.
- ***input***— data with the same datatype as field.

Note that you must even if you have no parameters, the function must include open and close parentheses.

Examples

A Y[es]/N[o] field where 1 is yes and 0 is no. Based on language used, the corresponding letters should be used, but the actual values are either 0 or 1.

```
Format mask:
   'U1yes_no()'

   Function/Trigger yes_no:
   if (parm[0])        /* input ?     */
      return(decode(G.lan,"SWE",decode(parm[1],"J",1,"N"0,-1),
                          "ENG",decode(parm[1],"Y",1,"N",0,-1));

   else                /* output      *
      return(decode(G.lan,"SWE",decode(parm[1],0,"N","J"),
                          "ENG",decode(parm[1],0,"N","Y"),"?"));
```