Polyhedra®

Installing Polyhedra

Enea Polyhedra Ltd

ENEA

Copyright notice

Disclaimer

Trademarks

Manual Details

Manual title	**Installing Polyhedra**
Document number	**src/release/install**
Revision number	**8.9.0**
Revision Date	**22nd July 2014**

Software Version

This documentation corresponds to the following version of the Polyhedra software:

Version	**8.9**

Preface

This document describes the Polyhedra installation kit and discusses how to install the Polyhedra system and make the executables available for use.

CONTENTS

1. Introduction

Welcome. You are about to start using the Polyhedra database management system. This document describes how to install and run Polyhedra components, and then works through a series of simple demos each of which introduces and illustrates a feature of the Polyhedra product set.

This document describes the mechanics of installing Polyhedra and the demo kit, and how to run the supplied executables. Please read it carefully before starting your installation and evaluation. It covers a number of platform-specific issues, and introduces the notation used in the companion document on evaluating Polyhedra.

The first section of the companion document 'Evaluating Polyhedra' (evaluate.pdf) describes the various demos provided as part of the installation kit. It is strongly advised that those new to Polyhedra work through these demos sequentially, in order to gain familiarity with the product. The final section of the companion document describes how you can continue familiarising yourself with Polyhedra, covering topics such as writing and compiling you own client applications, more details of the fault tolerance mechanisms, and an introduction to the Polyhedra manual set.

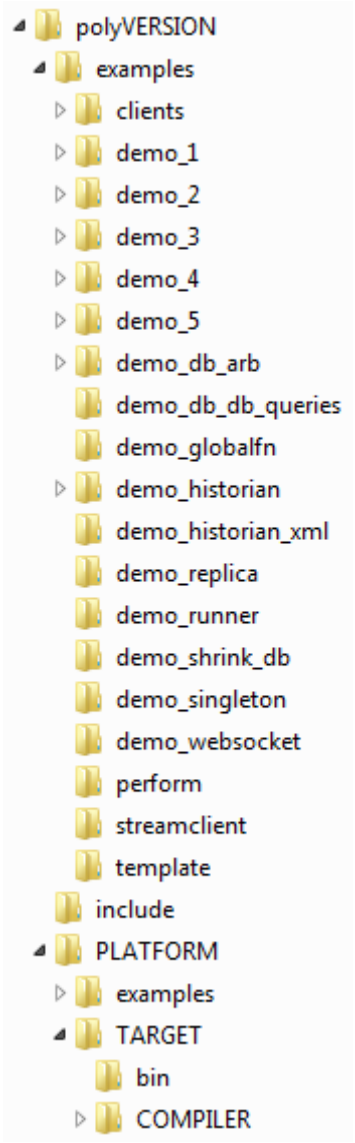
2. Installing and Running Polyhedra

This section describes how to set up your system to run the supplied executables, and general points about invoking the various commands.

2.1 Installing Polyhedra and the demonstration suite

Both the Lite and full versions of Polyhedra are supplied as a single compressed file. It comprises this document, the Polyhedra executables, the demonstrations described in the companion document 'Evaluating Polyhedra', together with the sample client applications in executable form. For Windows hosts, the file is a zip file, but on Unix hosts you will need to run gzip with the –d option to uncompress it and then use tar to unpack it (though on Linux and other platforms using the gnu version of tar, the decompression phase can be omitted if you include the '-z' option to tar).

Having unpacked the compressed file containing the Polyhedra demonstration kit you will have a directory structure as follows:



For Windows, the 'platform' is win32, the 'target' i386, and 'compiler' msvc, whereas for linux on Intel the 'platform', 'target' and 'compiler' will be linux, i386 and gcc respectively. Thus, the linux executables will be found in

poly<version>/linux/i386/bin, for example. The <platform>\<target>\<compiler> subdirectory structure makes it easier to install versions of Polyhedra for multiple platforms in one filing system.

The **poly<version>\examples\clients** directory has at least two subdirectories for each client – one containing the source code for the client written using the Callback API and one containing a functionally identical client written using the ODBC API. For a discussion of the two different APIs see the 'Further Directions' section of the companion document 'Evaluating Polyhedra'. The demo directories each contain an 'init' subdirectory containing the necessary files that can be used to recover or rebuild the database for the demo to its original condition.

The **poly<version>\<platform>\<target>\bin** directory contains all the executables or load modules that constitute the Polyhedra Demo suite and all the example database clients used throughout the demonstrations. Some demonstration kits will not provide a bin directory, and some also provide a **poly<version>\<platform>\examples** directory.

The **poly<version>\<platform>\<target>\<compiler>\lib** and **poly<version>\include** directories contain the libraries and header files needed to compile and link your own client applications.

The **poly<version>\<platform>\<target>\doc** directory contains instructions for building your own client applications and the standard Polyhedra executables (RTRDB, SQLC and CLC) using the libraries supplied in the kit.

The **poly<version>\<platform>\<target>\examples\common\src** directory contains example implementations of the user supplied functions that are required when building your own client applications and the standard Polyhedra executables.

Only the contents of the subdirectories of the **poly<version>\<platform>** directory vary according to the platform; the remainder are platform-independent.

2.2 Making the executables accessible

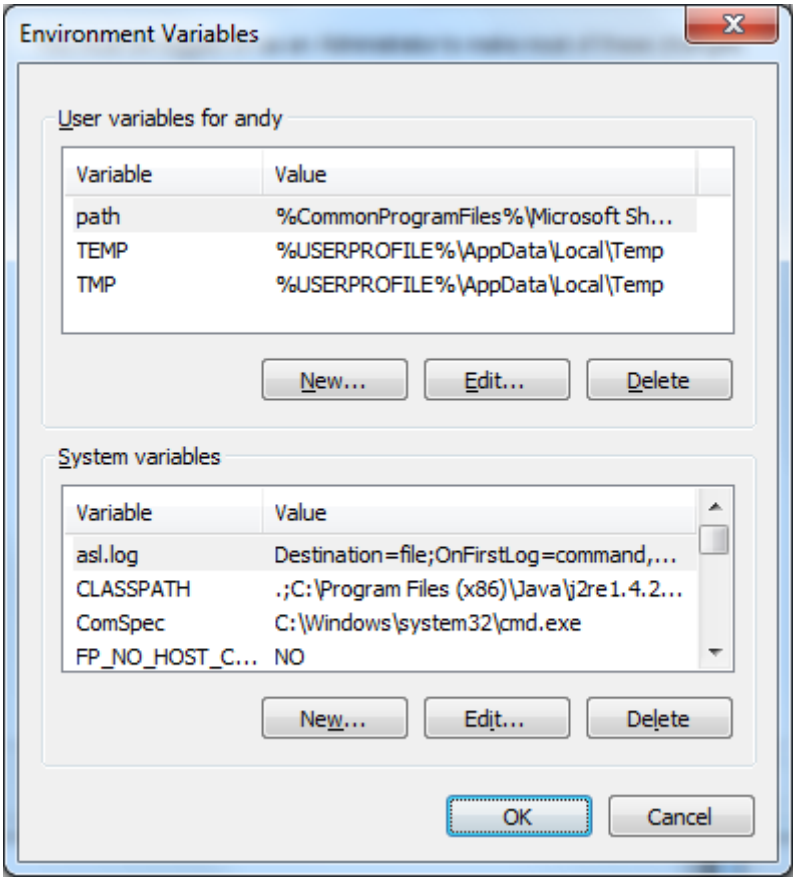
Operating systems allow you to invoke executables by typing in their full path names (optionally followed by any parameters) into the 'command shell', but to make life easier they also allow you to omit the directory part of the file name if the executable is in one of a special set of directories (defined by the 'command path'). Rather than move the executables from **poly<version>\<platform>\<target>\bin** to one of these directories, it is often better to extend the 'command path' to include this directory. The following subsections illustrate how this can be achieved.

2.2.1 Setting the path in Windows

Assuming you unpacked the demo kit into the directory C:\, then under Windows you can initiate a command shell and then issue the command…

```
set path=%path%;c:\poly<version>\win32\i386\bin
```

… which will take immediate effect but will only apply to the current command shell. To avoid having to issue such a command every time you start a command shell from which you may want to start off a Polyhedra executable, you could set up a 'short cut' that on start-up executes a script containing this line. Alternatively, you can edit the registry setting to achieve this effect. The simplest way to achieve this is via the Environment variables dialogue box from the "System" settings on the control panel. The dialogue box looks as follows:



Edit the path environment variable and append the directory containing the Polyhedra executables onto the end of the path. If at a later stage you decide to remove Polyhedra from your system, the environment variable should be changed back to its earlier value.

2.2.2 Setting the path in Unix

If you are installing the Polyhedra demo kit for your own use, and have unpacked the compressed file into a subdirectory of your home directory - ~/poly, say – then to make it easy to run the supplied executables you need to ensure the directory containing the executables are in the 'command path', as defined by the PATH shell variable. How one sets the variable depends on the command shell being used; with csh and tcsh, for example, on Linux for Intel one could issue the following command:

```
setenv PATH ~/poly<version>/linux/i386/bin:$PATH
```

… whilst with bash the following syntax is used:

```
export PATH=~/poly<version>/linux/i386/bin:$PATH
```

With these shells, the PATH variable is expected to hold a colon-separated list of directories, in which the shell will look in turn when looking for an executable; the above examples are both extending the current list to include the directory ~/poly<version>/linux/i386/bin.

NOTE:	**Other command shells may use their own syntax. Please refer to their relevant documentation.**

To avoid having to issue the above commands when initiating a new shell, they can be embedded in the 'initialisation file'; for csh and tcsh, this would be ~/.cshrc, whilst for bash this would be ~/.bashrc – other shell programs have their own conventions.

If multiple usage of Polyhedra is likely, it is best to move the contents of **poly<version>\<platform>\<target>\bin** into one of the 'standard' directories already contained in the command path of those users. Typically, this could be /usr/local/bin, or /usr/bin. Similarly, if more than one person will be developing client applications, moving the headers to /usr/include and the library files to /usr/local/lib or /usr/lib will make life easier. To move files in these standard places normally require root

privileges – if you do not know how to logon as or su to 'root', ask your system administrator to move the files for you. Polyhedra executables do not normally need root privileges to run.

2.2.3 Setting the path on OSE

You will need to transfer the load modules to a file system on your OSE target, for instance /ram or /tftp.

The PATH shell variable is then set to the directory containing the load modules using the `setenv` OSE shell command. For example, if the load modules are located in the directory /ram/poly<version>/ose/powerpc/bin, the PATH shell variable is set using the command:

```
setenv PATH /ram/poly<version>/ose/powerpc/bin
```

2.3 Installing the Polyhedra ODBC driver on Windows

The Polyhedra ODBC driver, which can only be installed on Windows, provides access to Polyhedra databases from a standard ODBC application. The sample ODBC applications use the Polyhedra ODBC API rather that the driver and so do not require the ODBC driver to be installed, even when running them on Windows.

The Polyhedra ODBC driver is supplied as a zip file, which should only be installed on Windows. To install the driver you must extract the contents of the zip file into a suitable directory and then run the set-up program poly<version>\odbc\setup.exe from that directory. This copies the driver and related files into the Windows system32 directory and registers the driver with the ODBC driver manager. It also configures an example data source and invokes the ODBC Data Source Administrator control applet, allowing further data sources to be configured, if required. Please see section 2.3.1 for details of how to add a Polyhedra ODBC data source. Note that it is always possible to configure data sources after the installation has completed. When you have finished configuring any data sources, click on the 'OK' button to continue the installation. When the set-up program has completed successfully the extracted files can be deleted.

2.3.1 Adding a Polyhedra ODBC data source

Before you can access a database with the Polyhedra ODBC driver you must add a data source for it. The ODBC Data Source Administrator control applet is used for all ODBC data source configuration, including adding, modifying and deleting data sources. To add a Polyhedra data source first start the ODBC Data Source Administrator control applet (which can be found either in "Administrative tools" or as a separate control panel item), then select either the 'User DSN' or 'System DSN' page and click on the 'Add…' button. In the 'Add Data Source' dialog box, select the 'Polyhedra 32-bit Driver' from the list of installed ODBC drivers and click on the 'Finish' button. The 'ODBC Polyhedra Setup' dialog box will now appear allowing you to enter information to set up the data source. The fields have the following meaning:

Field	Description
Data Source Name	The name used by the ODBC application to identify the data source.
Description	A textual description of the data source.
Data Service	The Polyhedra data service used to connect to the database.
Environment	An environment string passed to the database when the connection is established. Its precise use depends on the Polyhedra database being accessed.
Fault Tolerant	A check box indicating whether a fault tolerant connection to the database is required. If enabled, multiple databases can be specified in the Data Service field using a comma separated list of data services.

When you have completed entering the information into the dialog box, click on the 'OK' button. The new Polyhedra ODBC data source is now added.

2.4 Document conventions

The following subsections illustrate the typographic conventions used throughout this and the companion document, and some platform-specific notes.

2.4.1 Interaction between the user and the computer

Output from the computer will be displayed in the following text:

```
computer output
```

User input to the computer will be displayed in the following text:

```
user input
```

All user input will be preceded by a prompt that indicates where the user should be entering the command. The prompts used in this document are:

`%`	the command prompt (MS-DOS prompt in Windows, or at the shell prompt in Unix or on OSE)
`SQL>`	the interactive SQL client, SQLC.

> On Windows and Unix, the command prompt can be set up in a variety of ways – thus, it could be configured to show the current directory. For the purposes of exposition, a simple form is assumed. In addition, SQLC will use the | character as a prompt when you are entering commands that extend over many lines – essentially, to remind you to type in the semicolon ';' character to end the command.

2.4.2 Notes on starting the components under Windows

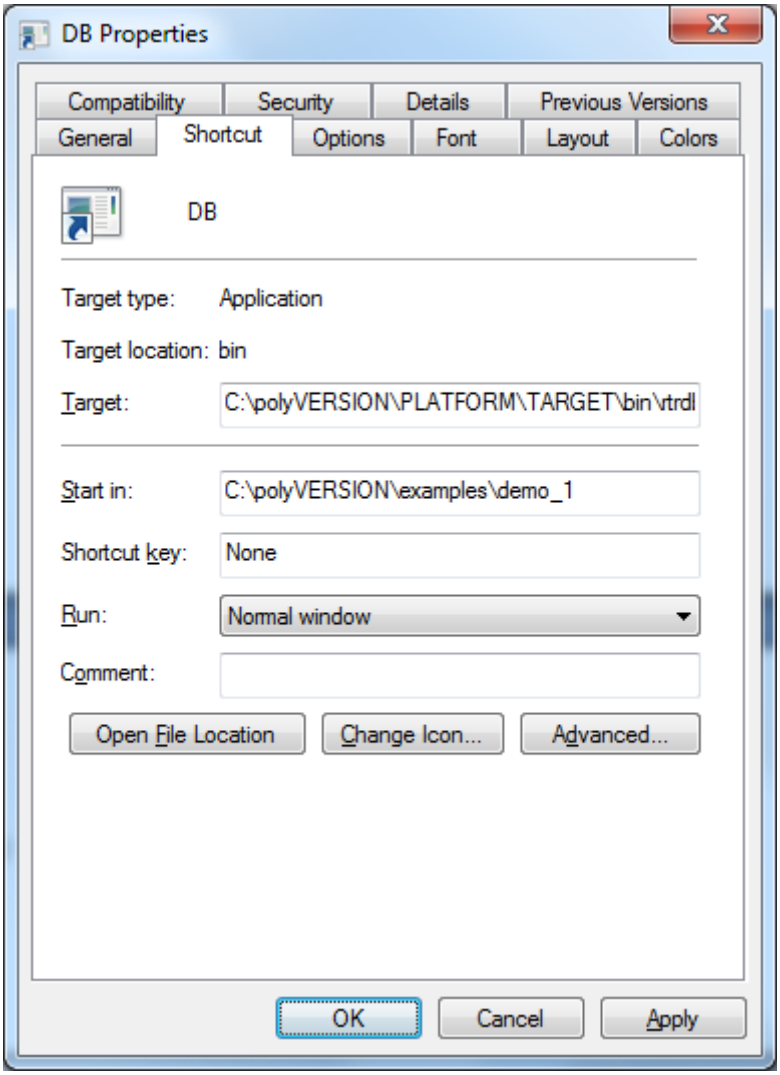
The SQLC component creates its own window for interacting with the user. All other Polyhedra components are console applications and so do not create their own window. If you want to send a copy of log messages to a file for later inspection, then instead of starting the RTRDB with a line such as…

```
% rtrdb db
```

…you would enter…

```
% rtrdb db > db.log
```

… to cause the log messages to be copied to a file. If you prefer not to use the Windows command shell, it is also possible to set up 'shortcuts' to start up Polyhedra components, by using the file/New/Shortcut… memory item in the directory window in which you wish the shortcut, and filling in the relevant details:



The ‘Start in’ field can be left blank if the shortcut is in directory in which you wish to run the executable. The ‘Run’ field can be set to ‘minimised’ (instead of the default value of ‘normal’) if you want to hide the output window.

2.4.3 Notes on starting the components on Unix

The RTRDB Polyhedra component would normally be started in the background, as it produces little output. Consequently where the instructions in this document read:

```
% rtrdb db
```

you would enter:

```
% rtrdb db &
```

In some of the demonstration exercises these components do provide significant output during their run. This is noted in the appropriate section describing the exercise. In these cases components can be run in their own window under Unix, primarily to distinguish the output from different Polyhedra components. (Under Windows the Polyhedra components (apart from the client applications, which have been linked as console apps) initiate their own window, and so no change is required.) If you wish, you can ‘capture’ the output to file – for csh or tcsh, for example, one could type a command like…

```
% rtrdb db >& db.log &
```

… to reroute the stderr output to db.log, or…

```
% rtrdb db |& tee db.log &
```

… to route a copy of the stderr output to db.log. Other shells use their own syntax to achieve similar effects.

2.4.4 Notes on starting the components on OSE

To run the Polyhedra RTRDB component in background on OSE use the `pgrun` shell command:

```
rtose@walnut> pgrun rtrdb.elf db &
```

To redirect its output to a file use:

```
rtose@walnut> pgrun rtrdb.elf db > /ram/db.txt &
```

2.5 Other platform-specific issues

Windows Platforms	Unix and OSE Platforms
Windows uses the back-slash character '\' to separate the directory component(s) of file names	Unix and OSE use '/'
	Unix and OSE have case-sensitive filenames, and also is case-sensitive when issuing commands
Windows uses the carriage return character code followed by the linefeed character code but can accept either convention. Consequently, if you have used a Windows tool to edit any of the files of the demo kit, it might give problems on Unix. Windows is more tolerant, accepting either convention – so the Unix convention has been used in all the supplied files.	Unix and OSE use the linefeed character code as a line terminator character

In this document, the examples use the windows file-separator '\', and the supplied files all have lower-case names.