OpenML ML SDK 1.0
Developer's Manual
April 2004

New in this SDK

Users familiar with previous (Beta) versions of this SDK should note that the ML system no longer relies on the "mldaemon" tool, and the daemon is no longer distributed. See "Installing and running ML executables" on page 8 for more information.

Installing the SDK

Installing on Windows

The SDK for Windows is distributed as a Microsoft Installer package. To install this package, un-zip the file into a temporary location by right-clicking on the zip file and selecting "Extract all...". Then view the extracted files, and right-click on the Windows Installer Package named "mlSDK". Select "Install", and follow the prompts. When installation is complete, you may delete the installation files you extracted from the zip'ed package.

Note: if you plan on re-building the SDK from source (see the section "Building the SDK" on page 6), you may find it more convenient to install the SDK package in a non-default location, and ensuring that the installation path you choose does not contain spaces in any of the directory names. For instance, you may choose to install the SDK into `C:/OpenML` (rather than the default location `C:/Program Files/OpenML`).

Installing on Linux

The SDK for Linux is distributed as a compressed tar file. Go to the location in which you wish to install the SDK, and use the command:

```
    tar zxvf <ML tar file name>
```

This will place all SDK files in the "ml" sub-directory.

SDK Contents

Windows SDK Contents

The Windows SDK is contained in the "OpenML" directory, which is located in the top-level directory chosen during installation – by default, this is:
`C:/Program Files/OpenML`. The "OpenML" directory contains the following sub-

directories:

- `bin`: pre-compiled versions of the “mlquery” utility and the “mlAudioCapture” sample program.
- `include/ML`: header files for the “ml” and “mlu” libraries.
- `lib`: pre-compiled versions of the “ml” and “mlu” libraries. These are the “IMPLIB” libraries only, i.e.: the libraries required for linking an ML application program. The DLLs are described below.
- `examples/ML`: source code and Microsoft Visual Studio project files for ML sample programs.
- `man`: documentation, in the Unix standard “man page” format, for functions provided by the “ml” and “mlu” libraries (this information is also provided as a PDF document, in the “docs” directory).
- `src`: contains a zip file of the ML SDK source tree. See “SDK Source Tree Contents” on page 3 for details.
- `docs`: the ML User Guide and an errata to that guide, and PDF versions of the “man page” documentation for the “mlu” utility library and the DI/DD API (for ML module developers). Note that Chapter 1 of the ML User Guide is a good “Beginner's Guide” to programming with ML.

In addition to these directories, the SDK installation process places files in the following system directories:

- `Windows/system32`: contains the DLLs for the “ml” and “mlu” libraries, required to run any ML program (including the “mlquery” utility). This directory also contains the `audiofile` DLL, which is required for certain sample programs. Since this system directory is automatically searched for DLLs, no extra configuration is required before using ML programs.
- `Windows/system32/openml/mlmodule`: contains pre-compiled versions of the sample modules. This directory is searched by default by the ML libraries, so no extra configuration is required before using ML on your system.

Linux SDK Contents

The Linux SDK is contained in the “ml” sub-directory where the package was installed. This directory contains the following files and directories:

- `oss`: source tree for the ML SDK, as described below.
- `linux`: pre-compiled versions of the libraries, tools and modules. Note that if the SDK is re-compiled, the compiled components will be placed in this same directory structure, thus over-writing the pre-compiled versions distributed with the SDK.

The SDK does not contain a copy of the libaudiofile shared library. This is generally pre-installed on Linux systems; however, if your system does not have this library, you may find it at: `http://oss.sgi.com/project/audiofile`.

SDK Source Tree Contents

The source for the SDK, in the “oss” directory, consists of the following:

- `build, make`: makefiles and scripts for the multi-platform building environment. The build environment is command-line based (i.e.: Unix-like) on all platforms. For more information, refer to the section “Building the SDK” on page 6.
- `lib/ml, lib/mlu`: source for the ML SDK libraries “ml” and “mlu”. The “ml” library implements the core of the ML API, as defined in the 1.0 specification. The “mlu” library implements additional utility functions to ease the process of developing an OpenML ML application.
- `tools/mlquery`: source for the ML query utility. This utility is a command-line program that uses ML services to report on the devices available on the system; it can also report on the devices' properties.
- `devices/nullxcode, devices/ossaudio, devices/v4l, devices/winaudio, devices/ustsource`: source for the sample devices (“mlmodules”) distributed with the SDK. “nullxcode” is a sample transcoder, and compiles on Linux and Windows. “ossaudio” and “v4l” are sample audio and video modules respectively; they are built on top of Linux APIs, and can be built on Linux only. “winaudio” is a sample audio module built on top of the Windows WAVE API, and can be built on Windows only. “ustsource” is an example of a module that provides a system-wide UST source; it can be built on all platforms (note however that on Windows, it is not pre-installed in the system directory. In order to use this module, it is necessary to compile it) Note that the modules included with the SDK are intended as examples only, and are neither fully functional nor fully compliant with the 1.0 specification.
- `man/man3dm`: man pages for the API implemented by the “ml” and “mlu” libraries, as well as man pages on the Device-Independent/Device-Dependent (DI/DD) API, used by module developers.
- `doc`: ML documentation, in PostScript and PDF formats. The user guide (especially Chapter 1) serves as a good introduction to ML programming; it should be used in conjunction with the OpenML specification document (available from the Khronos web site: http://www.khronos.org/openml/spec.html), which serves as a reference guide.
- `examples`: sample command-line programs that use ML to perform audio or video I/O. In order to use these programs, ML modules must be available (either sample modules distributed with the SDK, or production modules distributed with OpenML-compliant I/O hardware).
  The `examples/mlAudioCapture` directory contains a sample audio capture program that uses a graphical user interface; it allows capturing audio at various sample rates and writing the result to a “.wav” file. This sample compiles on Windows-based platforms only.

In addition to these directories, the SDK contains a few files at the same level as the “oss” directory. Of particular interest are:

- `LICENSES`: license information for the OpenML ML SDK source code.
- `SET_ENV`: script (for tcsh shells) to set environment variables required to build the SDK.
- `GNUmakefile, BUILD_LINUX`: top-level makefile and build script, to build the entire SDK. See the section "Building the SDK".

Finally, the SDK contains a local installation directory, which is where header files and binaries are installed when you re-build the SDK source tree. This directory – which is created at build-time if it does not yet exist – is at the same level as the "oss" directory, and is named either "linux" or "win32", depending on the platform. Its contents are somewhat different depending on the platform, but are generally as follows:

- an "include" directory, for all "ml" and "mlu" header files;
- a "lib" directory, for the "ml" and "mlu" libraries ("IMP" libraries only, on Windows);
- a "bin" directory, for the ML tools (and DLLs, on Windows);
- a "man" directory, in which all man pages are installed;
- an "examples" directory, in which the source code for the sample programs is installed – note that this does *not* contain pre-compiled binaries.

Software Requirements and Dependencies

The ML SDK provides a common building environment on both Windows and Linux. This environment is command-line based; while this is standard for Linux, and thus requires no special tools, on Windows it does require the installation of the cygwin development environment. Note however that this is only necessary for building the full SDK; if the intention is only to build the sample programs, the cygwin environment for Windows is not required – see details in the section "Building the Sample Programs" on page 5.

The specific requirements for each platform are listed below.

Linux Requirements

The ML SDK supports the following operating system and compiler versions:

- RedHat Linux version 8.0
- gcc version 3.2

Other versions may also work, but have not been tested.

Windows Requirements

The ML SDK supports the following operating system and compiler versions:

- Windows XP Professional

- Microsoft Visual Studio (C++), version 6.0, service pack 5
- Metrowerks CodeWarrior, version 8 (not fully supported)
- gnu gcc version 3.2 (not fully supported)
- cygwin development environment, version 1.3 or later.

To install the cygwin environment, go to `http://www.cygwin.com` and click on "Install or update now!". This will launch the setup program. Follow the instructions until you reach the "Select Packages" dialog. In this dialog, ensure that the "Devel" category is selected for installation. It is recommended that you also select the "Xfree86-bin" package (in the "Xfree86" category): this contains the "makedepend" utility, which is optionally used with the Microsoft Visual C++ compiler. Finally, you should select the "tcsh" package (in the "shells" category) for installation.

The build environment is currently configured for use with the "tcsh" shell. To automatically start a tcsh shell (rather than the default "bash" shell), edit the file `C:/cygwin/cygwin.bat` and change the last line (which should start with "`bash...`") to the following: "`tcsh -l`". If you prefer to use bash, you will need to adjust the environment-setting script located in `ml/SET_ENV`.

Building the Sample Programs

The sample programs can be found either in the SDK source tree, along with the rest of the SDK source, or in an "end-user" installation directory. This second location is separate from the SDK source itself, and does not require the same build environment – generally speaking, it is closer to what an "ML application" build environment might look like.

Unless you wish to also fully re-build the SDK, it is recommended that you compile the sample programs in the "end-user" installation directory, as it is simpler and more convenient.

Building on Linux

The "end-user" installation directory is `ml/linux/usr/src/ml/examples`. This directory contains a makefile (named "GNUmakefile") that is independent of the SDK build environment. The following environment variables may be used to customize the behavior of the makefile:

- `ML_INCLUDE_PATH`: may be set to the directory that contains the "ML" sub-directory in which all the ML headers may be found. If the headers are installed in the standard include search path (i.e., `/usr/include/ML`), this variable is not needed. For example, if you installed (un-tar'ed) the SDK in `/home/OpenML`, you could set this variable to `/home/OpenML/ml/linux/usr/include`.
- `ML_LIB_PATH`: may be set to the directory containing the "ml" and "mlu" shared libraries. If these libraries have been copied to the standard library search path

(generally /`usr/lib`), this variable is not needed.
For example, given the same installation path as above, you could set this to: `/home/OpenML/ml/linux/usr/lib`.

- `ML_DEBUG`: You may set this to a non-zero value to compile the sample programs with debug settings (no optimization and full debugging symbols). If the variable is not set, or if it is set to 0 (zero), the programs will be compiled with optimized settings and no debugging symbols.

See the section "Building the SDK" for instructions on compiling the sample programs from within the SDK source tree.

Building on Windows

The stand-alone sample programs are installed in the `ml/examples` sub-directory of the top-level installation directory. This directory contains Microsoft Visual Studio 6.0 project files (and a workspace file) to build all samples. Thus, to build only the code contained in this directory, it is not necessary to install the cygwin environment: only the Visual Studio environment is required.

In order to compile the sample programs with one of the other supported compilers, it is necessary to use the command-line build environment (which requires that cygwin be installed) from within the SDK source tree (in `ml/oss/examples/common`). Refer to the section "Building the SDK" on page 6 for instructions.

Building the SDK

The SDK is packaged with pre-compiled binaries for all libraries, tools and modules, so it is not essential to compile any of these yourself (please note that compiling the sample programs may be done *without* compiling the full SDK – see the previous sections). However, if you wish to compile the SDK, you may do so: instructions are given below for Linux and Windows.

Building on Linux

The simplest approach to compiling on Linux is to use the `BUILD_LINUX` script, in the `ML` directory. This will set all required environment variables, and will invoke the "make" utility in all appropriate directories.

You may also build individual directories manually. In order to do this, you must first set the required environment variables. The BUILD_LINUX script can be invoked with the "-n" argument (or simply answer "n" to the question "Continue with the build...?") to set the variables in a "bash" shell. With a "tcsh" script, use the command:

```
      source ./SET_ENV
```

See the section "Compiling Individual Sub-directories" on page 8 for details on manually building the source tree.

Building on Windows

Before attempting to build the SDK on Windows, it is necessary to extract the source tree from its zip file (which is located, in a default installation, in `C:/Program Files/OpenML/src/ml-sdk-source.zip`).

Important Note: to avoid problems when setting your shell environment variables, the path to which you extract the source tree **must not** contain spaces. For this reason, the source tree may not be extracted "in place", in the default SDK installation directory. You must create a new directory structure, with no spaces in the directory names. For example, you could choose to extract the SDK source to `C:/OpenML`. Alternately, you could choose a non-default directory (such as `C:/OpenML`) when first installing the SDK package – in this case, you may extract the SDK source "in place".

The build environment is shared among both the Linux and Windows platforms; as a result, a single set of makefiles is used to build the source.

Currently, only "manual" compilation is available on Windows – requiring you to set your environment variables and invoke the make process directly. There is currently no Windows version of the `BUILD_LINUX` script.

To set your environment variables, issue the command:

```
     source ./SET_ENV
```

in the `ML` top-level directory. Note that this works only for the "tcsh" (or "csh") shell; if you wish to use the "bash" shell, you will need to convert the `SET_ENV` script to use "sh" syntax.

By default, the Windows version will be compiled with the Microsoft Visual Studio 6.0 C compiler. To select a different compiler, set the `USE_COMPILER` environment variable to one of the following settings:

- `gnu`: GNU gcc compiler. Note that this compiler is not fully supported, and may produce unreliable results. In particular, if you choose to use this compiler, you must recompile the entire SDK with it; it is not possible to mix-and-match libraries and executables compiled with GNU with those compiled with either Visual Studio or CodeWarrior. This setting assumes that the GNU compiler is installed through cygwin, and is available through your `PATH` environment variable.
- `mwcc`: Metrowerks CodeWarrior version 8. This assumes that the compiler is installed such that it is available through your `PATH` environment variable. If that is not the case, you may set the `MWCC_PATH` environment variable to point to the directory containing the CodeWarrior command-line executables, followed by a trailing "/". Note that this compiler is not fully supported, and may produce unreliable results.
- `msvc`: Microsoft Visual Studio 6.0 (default setting). This assumes that the compiler is installed such that it is available through your PATH environment variable; furthermore, the `INCLUDE` environment variable must point to the directory containing the standard header files (this is set by default during normal installation of Visual Studio). Alternately, you may set the `MSVC_PATH` environment variable to the root of the VC98 directory tree, with no trailing "/" (for example, "`C:/Program\ Files/Microsoft\ Visual\ Studio/VC98`")

By default, the Windows version will compile fully-optimized libraries and executables that contain no debugging information. In order to compile non-optimized binaries with debugging information, set the `OPENML_DEBUG` environment variable to any value except 0 (zero). Setting the environment variable to 0 (zero) selects an optimized, non-debug build (the default setting).

With the CodeWarrior and gcc compilers, the makefiles automatically determine the dependencies for each source file; this mechanism ensures that objects are re-compiled automatically when one of their dependencies changes. Support for this mechanism with Visual Studio is still experimental, and is disabled by default. To enable automatic dependency-generation with the Microsoft compiler, set the `USE_MAKEDEPEND` environment variable to any non-zero value. Otherwise, in the absence of dependency information, when you change a header or source file, it is safer to perform a full rebuild, by using "make clean" before issuing the regular "make".

Now that all environment variables are set, you are ready to compile the source tree. The following section describes how this is done.

Compiling Individual Sub-directories

To compile the entire SDK, including modules, issue the "make" or "make install" command in the `ML` top-level directory. It is also possible to use the "make clean" target in this directory, in order to remove all temporary files from the source tree. Note however that files installed in `ml/linux` or `ml/win32` (more on this later) will not be removed by a "make clean".

You may also go to individual sub-directories and issue "make" in order to compile only selected components of the SDK. The makefiles (which are always called `GNUmakefile`) reside in the "common" sub-directory of each component, e.g.: `ml/oss/lib/ml/common`.

Note that if the header files for the "ml" or "mlu" libraries have changed, or are no longer present in the `ml/linux/usr/include/ML` (or `ml/win32/usr/include/ML`, depending on the platform) directory, you must first issue the "make headers" command in `ml/oss/lib/ml/common` and `ml/oss/lib/mlu/common`. This will install or update the header files so that other components of the SDK may include them.

The result of compiling the "ml" and "mlu" libraries will automatically be installed in the platform-specific distribution directory structure (`ml/linux` or `ml/win32`). The same is true of the modules. The tools (`mlquery`), however, do not automatically get installed; to install them, use the "make install" command.

Installing and running ML executables

There are two options to run the ML SDK executables:

- the executables and libraries may be used directly from their local installation directories, in `ml/win32/...` or `ml/linux/...` (depending on the platform).

This requires no modification of system directories; it does however require you to set certain environment variables (see platform-specific sections below for details). Note that on Windows, the local installation directories are initially empty (and may not even exist) – it is necessary to build the SDK in order to obtain binaries in these locations. On Linux, however, the SDK ships with pre-compiled binaries installed in these directories.

- the libraries and executables may be installed in the standard system locations, making them available system-wide. Note that on Windows, the SDK installation process automatically places the libraries in the standard system locations.

Note that previous versions of the SDK included the tool "mldaemon", which had to be running in order for any ML program to work. This is no longer the case: there is no need to start any special program before running an ML application or utility. On Windows, it is no longer necessary to start a special ML service.

Installing and running on Linux

In order to run the executables directly from their default location, you must set the `LD_LIBRARY_PATH` environment variable to point to the sub-directory containing the `ml.so` and `mlu.so` libraries (`ml/linux/usr/lib)`. It is also necessary to set the `_MLMODULE_ROOT` environment variable to `ml/linux` in order for the ML library to find the modules. Both environment variables are set for you in the `SET_ENV` script.

You may then call the ML executables; the query utility is in:
`ml/linux/usr/bin/mlquery`

To install the ML libraries and executables in a system-wide location, you will need root permissions. Refer to your operating system's documentation, or ask your system administrator, for the appropriate directory; generally, this will be `/usr/lib` for the libraries and `/usr/bin` for the mlquery utility. The modules should be installed in `/usr/lib/mlmodule`. When this is done, there is no need to set environment variables.

Installing and running on Windows

The SDK installation process automatically copies the "ml" and "mlu" DLLs to the standard system directory (generally `C:/Windows/system32`). The sample modules are copied to the standard module directory (generally `C:/Windows/system32/OpenML/mlmodule`). If you wish to use these libraries and modules, there is no need for any extra configuration.

You may run the `mlquery.exe` utility directly from the installation directory, which is (for a default installation of the SDK): `C:/Program Files/OpenML/bin`

If you re-build the SDK, the DLLs (`ML10.DLL` and `MLU10.DLL`) will be placed in the local installation directory `ml/win32/usr/bin`. You must either copy these libraries to the system directory, copy them to the directory containing the executables with which you wish to use them, or copy them to a directory that is listed in your `PATH` environment variable. To avoid accidentally using the wrong libraries, you should first delete the old DLLs from the system directory.

If you wish to use sample modules that you have built, you must set the `_MLMODULE_ROOT` environment variable to point to the top of the directory structure containing the devices. For the default location of the modules, set this to `ml/win32`. This is correctly set by the `SET_ENV` script.

When you rebuild `mlquery.exe`, it is installed in `ml/win32/usr/bin`, along with the ML dynamic libraries.