Belgian eID Middleware

How to build the middleware

22/01/2009

Contents

1 Introduction

The goal of this document is to explain how to build the Belgium eID Middleware.

2 Third party components

The following components are required to build the eID Middleware. These components are not part of the package and must be installed and/or build separately.
(F=Fedora 9/10, D=Debian etch 4.0r6,S=OpenSuSE 11.0/11.1)

Component	**Used for**	**Recommended version on Windows**	**Recommended version on Mac**	**Recommended version on Linux**
Qt 3	2.6 compatibility	3.3.4	NA	F,D,S: default
Qt 4	Beid Gui	4.5.0	4.5.0	4.5.0
Qt 4 (static)	Gui of the Quick installer and eDT	4.5.0	4.5.0	NA
OpenSSL	Cryptographic operation of the application layer.	0.9.8g	0.9.8g	F,D,S: default
Xerces	Xml parser of the application layer	2.8.0	2.8.0	F,D,S: 2.7.0
Platform SDK 2003	Windows specific feature for the application layer	To avoid conflict with OpenSSL this version must not define OCSP_RESPONSE in the file wincrypt.h	NA	NA
Platform SDK 2008	Windows specific features for the QuickInstaller	Including file newdev.h	NA	NA
MSI merge module	CRuntime redistributable	Microsoft_VC80_CRT_x86.msm + policy_8_0_Microsoft_VC80_CRT_x86.msm	NA	NA

The following table shows the tools needed for building the MW. They must be installed before trying to build.

Tools	**Use to**	**Recommended version on Windows**	**Recommended version on Mac**	**Recommended version on Linux**
Visual Studio	Build the exe, libraries and msi. (Component needed : Visual C++, Visual C#, Redistributable Merge Modules)	2005 prof. SP1	NA	NA
XCode	Build the QuickInstaller	NA	3.1	NA
WiX	Build the msi	3.0.4415	NA	NA
Doxygen	Generate the documentation	1.5.7	NA	F,D,S: default
Swig	Generate the Java and C# wrapper	1.3.35	1.3.31	F,D,S: default
Jdk	Compile the java code	1.5.0	1.5.0	F,D,S: 1.5.0

3 How to build?

3.1 On Windows

3.1.1 Before you build

All the components and tools can be configured using the **set_path.bat** script. This script must be filled in with the path to each third party components. The following table gives explanation on all the environment variables.

Variable	Description	Remarks
BEID_DIR_QT_334	Path to Qt 3.3.4 (ex: C:\Qt\3.3.4)	The following files, at least, must be available from this path: **"lib\qt-mt334.dll" "lib\qt-mt334.lib" "bin\uic.exe" "bin\moc.exe"** and header files in **"include"** subfolder.
BEID_DIR_QT_450_EXE	Path to Qt 4.5.0 executable (ex: C:\Qt\4.5.0)	The following files, at least, must be available from this path: **"bin\uic.exe" "bin\moc.exe" "bin\rcc.exe"**
BEID_DIR_QT_450_INCLUDE	Path to Qt 4.5.0 includes (ex: C:\Qt\4.5.0)	The following files, at least, must be available from this path: **"include\QtGui\QtGui" "include\QtCore\QtCore" "src\gui\dialogs\qdialog.h" "src\corelib\kernel\qcoreapplication.h"**.
BEID_DIR_QT_450_DYNAMIC	Path to Qt 4.5.0 dynamic libraries (ex: C:\Qt\4.5.0)	The following files, at least, must be available from this path: **"lib\qtmain.lib" "lib\qtmaind.lib" "lib\QtCore4.lib" "lib\QtCored4.lib" "bin\QtCore4.dll" "bin\QtCored4.dll" "lib\QtGui4.lib" "bin\QtGuid4.dll" "bin\QtGui4.dll" "lib\QtGuid4.lib" "plugins\imageformats\qjpeg4.lib" "plugins\imageformats\qjpegd4.lib" "plugins\imageformats\qjpeg4.dll" "plugins\imageformats\qjpegd4.dll".**
BEID_DIR_QT_450_STATIC	Path to Qt 4.5.0 static libraries. (ex: C:\Qt\ 4.5.0_static)	The following files, at least, must be available from this path: **"lib\qtmain.lib" "lib\qtmaind.lib" "lib\QtCore.lib" "lib\QtCored.lib" "lib\QtGui.lib" "lib\QtGuid.lib" "lib\QtXml.lib" "lib\QtXmld.lib" "plugins\imageformats\qjpeg.lib" "plugins\imageformats\qjpegd.lib".**
BEID_DIR_OPENSSL_098G	Path to OpenSSL	The following files, at least, must be available from this path: **"lib\libeay32_0_9_8g.dll" "lib\ssleay32_0_9_8g.dll" "lib\libeay32_0_9_8g.lib"**

		"lib\ssleay32_0_9_8g.lib" and header files in **“include\openssl”** subfolder.
BEID_DIR_XERCES_280	Path to Xerces	The following files, at least, must be available from this path: **"bin\xerces-c_2_8.dll" "lib\xerces-c_2.lib"** and header files in **“include\xercesc”** subfolder.
BEID_DIR_PLATFORMSDK_2003	Path to Microsoft Platform SDK (for application layer)	Must at least contain: **"Include\winhttp.h"** To avoid conflict with OpenSSL this version **must not define OCSP_RESPONSE in the file wincrypt.h**
BEID_DIR_PLATFORMSDK_2008	Path to Microsoft Platform SDK (for MSI and QuickInstaller	Must at least contain: **"bin\msitran.exe" "bin\msidb.exe" "Include\newdev.h"**.
BEID_DIR_VS_2005	Path to Visual Studio (ex: C:\Program Files\Microsoft Visual Studio 8)	Use to run the build from make_win.bat Must at least contain: **"Common7\IDE\devenv.exe"**
BEID_DIR_DOXYGEN	Path to Doxygen (ex: C:\Program Files\doxygen)	Must at least contain: **"bin\doxygen.exe"**
BEID_DIR_SWIG	Path to Swig	Must at least contain: **"swig.exe"**
BEID_DIR_JDK_142	Path to the Jdk	Must at least contain: **"bin\javac.exe" "bin\jar.exe" "include\jni.h"**
BEID_DIR_WIX	Path to WiX	You need to install WiX (and its ProjectAgregator for interfacing with Visual Studio). The Install create an environment variable WIX that is remap to BEID_DIR_WIX (for consistence)
BEID_DIR_MSM	Path to MSI Merge Module	Must at least contain: **"Microsoft_VC80_CRT_x86.msm" "policy_8_0_Microsoft_VC80_CRT_x86.msm"**

3.1.2 Building

The windows folder, in the archive, contains scripts to easily build the middleware.

- The **“make_win.bat”** script check if all the components and tools are available and then it build everything. The output comes in the same folder.
- The **“make_edt_win.bat”** script, build the diagnostic tool.
- The **“make_sdk_win.bat”** script, build the sdk.
- If you just want to set the environment variable and open the solution in visual studio, you can run the “**open_ezbuild_sln.bat”** script.
- And **“open_ezbuild_edt_sln.bat”** open the diagnostic tool solution in VS.
- And **“open_ezbuild_sdk_sln.bat”** open the sdk solution in VS.

3.2 On Linux

The following Linux distributions are supported:

- Fedora 9,10
- OpenSuSE 11.0/11.1
- Debian etch 4.0r6

The build system is setup such that, after installation of the necessary libraries, tools, etc… a single command will perform the build on either of these distributions. This description assumes a clean installation of the Linux distribution. Since no third party software is delivered with the source package, an operational internet connection must be available before installing.
A script is provided that can be used to install the necessary third party packages.

3.2.1 Preparing the build system

As indicated before, several libraries are required by the software. On the Linux distributions, most of these libraries can directly be downloaded and installed by the package installer.
To make the work easier, a script is provided that will install all necessary libraries from the distributions package repository.
To install the necessary libraries:

- Go to the directory: <basedir>/beid-middleware-X.Y.Z-source-<buildnr>/linux
  cd <basedir>/beid-middleware-X.Y.Z-source-<buildnr>/linux
- Make the scripts executable:
  chmod +x *.sh
- Login as root and run the package install script
  su
  <password>
  install-pkg.sh
- Install the JDK 1.5
  Download the JDK 1.5 from the Sun website and install it according to the install instructions
  Verify if the system will find the jdk files with 'which javac'. If not, set the PATH: export PATH=<java_install_dir>/bin/:$PATH
- Install Qt 4.5.0
  Install Qt 4.5.0 from the sources:
  - download the file 'qt-x11-opensource-src-4.5.0.tar.gz'
  - unzip the file
  - go to the directory of the unzipped files
  - ./configure
    (to build Qt faster, build only the release version: ./configure -release)
  - make

- login as root
- make install

You can verify with 'which qmake' if the system will find the Qt 4.5.0 files. If not, set the path: export PATH=/usr/local/Trolltech/Qt-4.5.0/bin/:$PATH

- Logoff as root
  <ctrl>D
- Set the PATH environment variable to the java binaries:
  export PATH=<java_install_dir>/bin :$PATH
- Set the PATH environment variable to the Qt binaries:
  export PATH=/usr/local/Trolltech/Qt-4.5.0/bin/:$PATH

or/and put this command in the login script such that the PATH variable set each time you log in to the system

For Fedora 9 only:

- Install the ACR38u driver for Linux (Fedora 7) from the ACS website (www.acs.com.hk)
- Check if 'openct' is installed. If so, uninstall it as follows:
  - Open a terminal window
  - Login as root
  - rpm --nodeps --erase pcsc-lite-openct
  - rpm --nodeps --erase openct

For OpenSuse 11.0 only:

- The default openSuSE installation installs Qt4.4.0. If possible uninstall Qt4.4.0.
- Install from the OpenSuse site (http://www.opensuse.org/) wxGTK
  - On the OpenSuse site, go to the package search
  - Select OpenSuse 11.0
  - Search for wxGTK-devel
  - Install the package
  - Search for wxGTK-gl
  - Install the package

For OpenSuse 11.1 only:

OpenSuse 11.1 comes by default with Python 2.6, while OpenSuse 11.0 comes with Python 2.5. Since Python 2.5 and 2.6 are not 100% compatible, Python 2.5 should be installed next to Python 2.6 on the OpenSuse 11.1 system.

- Download Python 2.5 from the Python site www.python.org
- Unzip the downloaded file to a base directory
- Go to the base directory
- Run: ./configure
- Run: make

- Login as root
- Run: make altinstall
- Python 2.5 is now installed in /usr/local/bin/python2.5
- Verify where python 2.6 is installed with 'which python'
- Go to the directory where python 2.6 is installed
- The executable 'python' is a link to python2.6. Remove the link with 'rm python'
- Make a link to python 2.5: ln –s /usr/local/bin/python2.5 python
- Logoff as root

Remark: the OpenSuse 11.1 is now prepared to use Python 2.5. Do not forget to reset the link to Python 2.6 when finished.

3.2.2 Building

To simplify the build process, a script is provided. Run the script:

./make-linux.sh [option]

With:

[option]: --verifypkg : verify that all packages are installed before building.

If Qt 4.5.0 is not found, verify the Qt installation and set the PATH environment variable to the Qt binaries directory.
If build process succeeds, a tarball with the binaries and install script will be generated in the 'linux' directory.

3.3 On Mac OS X

For Mac, the easybuild will build only for OS X on i386.

3.3.1 Before you build

Before the middleware can be built, several packages must be installed on the build machine:
This includes:

- XCode 3.1 (gcc/g++ compiler, SVN)
- Qt4 version 4.5.0
- Xerces 2.8 (Automatically install by the easybuild)
- Jdk 1.5 or higher (Automatically install by Mac OS Software Update on Leopard)

These packages must be installed before the build procedure 'make-mac.sh' is started.
It is strongly recommended to use Qt 4.5.0. Installing other versions of Qt as proposed in this procedure can result in build errors or different runtime behaviors. The build procedure expects the installation of Qt 4.5.0 as described in the installation procedure below.

XCode:

Go to the apple site and download from the ADC site the XCode 3.1 developer tools.This contains the gcc and g++ compiler and other build tools

- download the Xcode 3.1 tools
- mount the dmg
- install the default tools

Qt4.5:

- download qt-mac-opensource-4.5.0.dmg
- install the package by following the instructions

The package is installed in '/usr/local/Qt4.5'

Xerces 2.8

The script *"download-install-xerces.sh"* will automatically download and install xerces 2.8, if it is not found in /usr/local/lib. The script is run within the *make-mac-universal.sh* script. It downloads the ppc and x86 binaries of xerces, creates a universal binary and copies it in the excepted location.

3.3.2 Building

The build procedure can be started with:

- cd <basedir>/beid-middleware-X.Y.Z-source-<buildnr> /macos
- ./make-mac-universal.sh

If the build process succeeds, a pkg and a dmg will be generated in the 'macos' directory.

Remark: During assembly of the pkg and dmg the tool hdiutil can fail for unknown reasons, resulting in an invalid pkg and/or dmg. Restarting the build process fixes the problem.

4 Folder organization

Here is a view of the folder organization.

- **beid-middleware-X.Y.Z-source-BUILD** : *Root folder*
  - **_src** : *Root folder for the sources*
    - **beid-2.6** : *Source for old MW 2.6 (backward compatibility)*
      - **src**
        - **eidlib** : *Eidlib 2.6 (Rebuild to take the new root into account).*
        - … : *Other source from the old MW needed for header files.*
      - …: *other folders from old MW.*
    - **eidmw** : *Source for new MW.*
      - **_Binaries35**: *Contain the binaries result of the build (Folder created at build time).*
      - **_Builds**: *Visual studio solution and other general make file.*
      - **_DocsExternal**: *The generated api documentation (Folder created at build time).*

- **_DocsInternal**: *The scripts to generate the api documentation*
- **applayer**: *Source for application layer library.*
- **cardlayer**: *Source for the card layer library.*
- **common**: *Source for the common library*
- **Csp**: *Source for the CSp component*
- **dialogs**: *Souce for the dialogs component*
- **eidgui**: *Source for the gui*
- **eidlib**: *Source for the C++ eidlib library*
- **eidlibCS**: *C# eidlib project.*
- **eidlibCS_Wrapper**: *Source for the C# wrapper (generated by swig).*
- **eidlibJava**: *Java eidlib project.*
- **eidlibJava_Applet**: *Source for the java applet.*
- **eidlibJava_Wrapper**: *Source for the java wrapper (generated by swig).*
- **misc**: *Miscellaneous.*
  - **bw_compatibility**: *Files for backward compatibility.*
  - **certs**: *Root certificates*
  - **licenses_files**: *Licenses files*
  - **mac**: *Mac specific (Installer…)*
  - **setup_win**: *File for 'Old' Installshield installer.*
  - **Wix_MW35**: *File for MSI installer*
    - **beidcleanup**: *Source of the cleanup tool.*
    - **IncludeFolder**: *Exe to create wxs file with the full content of a folder (To embed the html api in the sdk)*
    - **Kill_Process**: *Exe to kill a process (previously embedded in MSI)*
    - **MarkHidden**: *Exe to mark a folder as hidden (embedded in MSI)*
    - **MW35Wix**: *Wix project for the MW.*
    - **MW35Wix-Sdk**: *Wix project for the sdk.*
- **pkcs11**: *Source for the pkcs11 component.*
- **sdk**: *Samples for the sdk.*
- **tokend**: *Source for the Tokend.*
- …: *Other unit test projects.*

- **eID-QuickInstaller**
  - **eID-EZinstaller**: *Source for the QuickInstaller*
  - **EIKDiag**: *Source files for diagnostic framework.*
  - **MakeDmg**: *Script to generate the Dmg container for the Mac QuickInstaller.*
  - **QtZipper**: *Tools to zip the driver package.*
  - **ReaderDrivers**: *Reader drivers for windows.*
  - **zrc**: *Static library embedded in the QTZipper.*
- **ThirdParty** : *Source for Thirdparty (You can add here folders for TP components)*
  - **beid_sdk**
    - **2.6** : *Binaries for backward compatibility that are not rebuild here*

        - `3.5` : *Cross platform binaries and signed jar to include in sdk*
    - `linux` : *Easy build scripts for linux*
    - `macos` : *Easy build scripts for Mac OS*
    - `windows` : *Easy build scripts for windows*