User Manual

1. Information

Building with GNU tools (Linux, *BSD, MacOS X, mingw, etc.)

Generally these should be all that are needed to build the libraries, applications, and samples:

```
$ ./configure
$ make dep && make clean && make
```

Building Win32 Target with Microsoft Visual Studio

Generally we can just do these steps:

1. Visual Studio 6: open **pjproject.dsw** workspace,
2. Visual Studio 8/2005: open **pjproject-vs8.sln** solution,
3. Build the `pjsua` application.
4. Run

Building for Windows Mobile

Generally these are all that are needed:

1. Open **pjsip-apps/build/wince-evc4/wince_demos.vcw** EVC4 workspace,
2. Build the **pjsua_wince** application.

2. Layout of File Directories

All libraries;

PJLIB,
PJLIB-UTIL,
PJSIP,
PJMEDIA, and
PJMEDIA-CODEC)

are put into a folder collectively called **APOROYE/PJPROJECT** or just **PJ** libraries.

I. Download and unzip the content in the folder aporoye into your directory (e.g. C :\)

II. Brief Explanation of each library

Directories	Description
/build	Contains makefiles that are common for all projects.
/build.symbian *	Contains MMP files for building Symbian target
/pjlib	Contains header and source files of PJLIB. PJLIB is the base portability and framework library which is used by all other libraries
/pjmedia	Contains PJMEDIA and PJMEDIA-CODEC header and source files. The sources of various codecs (such as GSM, Speex, and iLBC) can be found under this directory.
/pjsip	Contains PJSIP header and source files. This library is the SIP protocol stack implementation.
/pjsip-apps *	Contains source code for PJSUA and various sample applications, including the Python wrapper.
Others for future work	e.g. • third_party • pjlib-util • pjnath

III. Each directory has these sets of subdirectories

Subdirectories	What they are
Src	Source files of the directory i.e. project
Lib	Libraries made from the build process
Include	Header files
Docs (used doxygen)	Contains doxygen configuration file to generate online documentation from the source files
Build/wince-evc4/output	Library, executable, and object files generated windows mobile build process
Build/wince-evc4	Project/workspace files to build CE/wince version of the project Microsoft Embedded Visual C++4
Bin	Binaries from building the process
Build	**Makefiles**, scripts, project files, workspace tohelp build the project

3. Disk Space Requirements

The build process will require

- About >=60 MB of disk space to store uncompressed source files
- Another 40 MB of additional space for building each target

4. Requirements

Building with GNU tools (Linux, *BSD, MacOS X, mingw, etc.)

In order to use PJ'S or APOROYE/PJPROJECT GNU build system, the following are needed:

GNU **make**
GNU **binutils**
GNU **gcc**

Running configure

```
$ cd  pjproject
$  ./configure
```

For help

```
$ cd  pjproject
$  ./configure  --help
```

Running make

```
$ cd  pjproject
$  make dep
$  make
```

Note:

Descriptions of all make targets supported by the Makefile's:

all
The default (or first) target to build the libraries/binaries.

dep, **depend**
Build dependencies rule from the source files.

clean
Clean the object files for current target, but keep the output library/binary files intact.

distclean, **realclean**
Remove **all** generated files (object, libraries, binaries, and dependency files) for current target.

Note:

make can be invoked either in the top-level PJ directory or in **build** directory under each project to build only the particular project.

5. Building for Windows Targets with Microsoft Visual Studio

The Visual Studio based project files can be used with one of the following tools:

- Microsoft Visual C++ 2005 (including Express edition),

For the host, the following are required:

- Windows 2000, XP,

6. Building the Projects

Follow the steps below to build the libraries/application using Visual Studio:

1. For Visual Studio 8 (VS 2005): open **pjproject-vs8.sln** solution file.
2. Make **pjsua** as Active Project.
3. Select **Debug** or **Release** build as appropriate.
4. Build the project. This will build **pjsua** application and all libraries needed by **pjsua**.
5. After successful build, the **pjsua** application will be placed in **pjsip-apps/bin** directory, and the libraries in **lib** directory under each projects.

7. Building for Windows Mobile Targets Windows (CE or PDA or SmartPhone)

One of the following development tools is needed to build SIP and media components for Windows Mobile:

- Microsoft Embedded Visual C++ 4 with appropriate SDKs, or
- MVS 2005 for Windows Mobile with appropriate SDKs.(VS2005 for Windows Mobile can import EVC4 workspace file)

Building Project

The Windows Mobile port is included in the main source distribution. Please follow the following steps to build the WinCE libraries and sample application:

I. Open **pjsip-apps/build/wince-evc4/wince_demos.vcw** workspace file. If later version of EVC4 is being used, this may cause the workspace file to be converted to the appropriate format.
II. Select **pjsua_wince** project as the Active Project.
III. Select the appropriate SDK (for example Pocket PC 2003 SDK or SmartPhone 2003 SDK)
IV. Select the appropriate configuration (for example, Win32 (WCE Emulator Debug) to debug the program in emulator, or other configurations such as ARMV4, MIPS, SH3, SH4, or whatever suitable for the device)
V. Select the appropriate device (Emulator or the actual Device).

VI. Build the project. This will build the sample WinCE application and all libraries (SIP, Media, etc.) needed by this application.

8. Building for Symbian Targets

Click Symbian document link

9. Running the Applications

Basic Peer-to-Peer

The easiest way to use pjsua is to use it in serverless configuration, to call or receive calls from other SIP user agents directly.

Running pjsua without any arguments will bind pjsua to TCP and UDP port 5060 of local host:

```
$ ./pjsua
```

This command below will initiate outgoing call to some SIP URL:

```
$ ./pjsua sip:192.168.0.10
```

Registering with SIP Provider

This command below will make pjsua send outgoing requests (such as INVITE) with *from:* header set to *sip:alice@example.com*, but pjsua will not register to any SIP servers:

```
$ ./pjsua --id sip:alice@example.com
```

To make pjsua register to a SIP provider, the command below can be used. This will make pjsua register to *sip: example.com* server using user-id *alice* and password *secret*. All command line arguments are mandatory:

```
$ ./pjsua --id sip:alice@example.com --registrar sip:example.com \
      --realm * --username alice --password secret
```

10. Config File

The config file basicly are the same command line arguments, saved in a file! For example, to specify above command line in the file called alice.cfg:

```
# This is a comment in the config file.
--id sip:alice@example.com
--registrar sip:example.com
--realm *
--username alice
--password secret
```

And to use the config file:

```
$ ./pjsua --config-file alice.cfg
```

11. References

1. www.pjsip.org
2. www.nokia.forum.com