NuDesign MultiProtocol Evaluation SDK (7.0) for Embedded uCLinux/SnapGear (Xscale) Installation and Usage Notes

Document History

Date	Rev.	Remarks
2005-07-04	A001	Original release
2006-02-13	A002	update for 7.0 release

Introduction

This document describes installation and usage of this SDK. The SDK enables building executables from projects produced by with the NuDesign Visual Embedded xAgentBuilder for C++.

This SDK is specifically for uCLinux/SnapGear ARM XScale (IXP4xx) targets.

The document discusses these main components of this release:

- Requirements
- SDK installation.
- SDK Configuration
- Building Generated Projects
- Executable use
- Limitations

Requirements

Visual Embedded xAgentBuilder for C++ produces projects that require include files and libraries specific to the target to complete the build process. For this version of Visual Embedded xAgentBuilder for C++, the NuDesign specific files are available in this software development kit.

Additional include files and libraries for the source distribution (e.g. SnapGear, uCLinux) and GNU are required.

The following is a specific list of requirements to build the projects produced for this target.

- PC Linux development host
- Arm-Linux GCC 3.3.2 and related tool chain built for the target and for big endian mode of operation.
- SnapGear 3.1.1 (or a comparable uCLinux) source distribution
- OpenSSL (libcrypto.so.0.9.7d a target specific build is provide in the SDK lib directory)
- NuDesign's xAB SDK for uCLinux/SnapGear and ARM XScale (BE)

The SDK has been tested using a PC "desktop" Linux, such as Mandrake, as the cross development host, using the versions indicated. Other versions or non-Linux platforms for these GNU tools are currently not supported, though may be suitable for development with this SDK.

The projects produced by this version of Visual Embedded xAgentBuilder for C++ require GNU development tools configured for ARM as a target.

Also note, this SDK has been built for the big endian (BE) mode of ARM XScale operation.

Installation

This release comes in the form of a PC Linux executable named
“`xABuCLinuxEvalSDK-7.0-linux-installer.bin`”.
:
By default the software will be installed under
`/usr/local/NuDesign/MultiProtocol`. The makefiles produced by the Visual Embedded xAgentBuilder for C++ expect that these defaults have been used. Installation may also require that you run the installer at a `root` privilege level.

The libraries, executables and web pages provided in the SDK also need to be available on your target. This may be accomplished a variety of ways but ultimately they will need to be made part of the OS image for distribution. The libraries should be placed under either the `/lib` or `/usr/lib` directories and the executables and related configuration files under either `/bin` or `/usr/bin`. The web files should be placed under `/usr/local/NuDesign/web` to correspond with the default configuration files for the executables.

Building Wizard Produced Projects

All xAgentBuilder projects have the same directory structure. The root directory contains the C++ source and header files relating to instrumentation code of the sub agent created by code generation. The names of these modules are directly related to key components (branches & tables) of the MIB used to generate the project. All modules in this directory are compiled and become part of a static library. This library is used as input to building executables or shared object libraries.

Subordinate to the root are three directories, `EXE`, `DLL` and `REM` which contain files that are used in conjunction with the static library to produce an independent executable (EXE), sometimes called a standalone agent, a shared object library (DLL) loadable from the NuDesign's SNMP service daemon or a remote sub agent.

Copy the project produced by the Visual Embedded xAgentBuilder for C++ wizard, to a convenient location on your development computer. The build facility is provided by a hierarchical system of `Makefiles`. The top most `Makefile` is situated in the project root directory. Invoking it by running `make`, will build all subordinate sub projects, for both `debug` and `release` versions of the project.

Each sub project can also be built independently by changing to the directory of the sub project and invoking `make`.

Make Configuration

The makefiles of the system are essentially self contained, except for one aspect. This is the location of the include files and libraries for the SDK, SnapGear and the gcc arm linux libraries. These are defined in each `makefile` as the following macros:

- **`TARGETSRC`**. This is the path to the SnapGear (source) distribution. By default it refers to `/projects/snapgear`. This **LIKELY** needs to be altered to build the project.

- **`GNU_ARM_TOOL_LIBS`**. This is the path to the arm-linux tool libraries. By default this is set to `/usr/local/lib/gcc-lib/arm-linux/3.3.2`. Alter this if GNU arm linux toolchain has been installed in a different location.

- **`GNU_ARM_LIBS`**. This is the path to the arm-linux gnu **b**ig **e**ndian libraries. By default this is set to `/usr/local/arm-linux/be`. Alter this if GNU arm linux toolchain has been installed in a different location.

- **`NUDESIGNDIR`**. This is the path to the SDK includes and libraries. By default this is set to `/usr/local/NuDesign/MultiProtocol/SnapGear-XScale`. Alter this if the SDK was installed in a different location.

Build Targets

There are two build "targets" for building a project. These are:

- **`debug`**
- **`release`**

Typing either of these on the make command line will cause only that version of the project to be built.

The build mode `debug`, builds with no optimization and with debug symbols, `release` builds with 'O2' optimization and all symbols stripped.

In addition following targets may by specified on any `make` command line.

- **`all`**, default target. Builds all objects and libraries.
- **`clean`**, removes all objects and libraries in the subprojects.

Build Options

Make Environment Variables

The configuration variables **`SNAPGEAR, GNU_ARM_TOOL_LIBS, NUDESIGNDIR`** and **`MODE`** variables can be specified by exporting their values to the shell invoking make or by specifying their values on the make command line, using the `variable=value` construct.

E.g. `'make MODE=debug'`

Conditional Building

A number of NuDesign specific conditional build macros are recognized. The following is this list that is defined and used in some or all make files:

- **`ND_PLATFORM_COYOTE`**. Always defined for this distribution. Further defines the platform.

- **`NDCFG_RUNTIME_INITIALIZATION`**. When defined causes the use of runtime initialization code for several common static properties. These properties are part of either the `BMibSubTreeHandler` (`MultiProtocol/Include/mibh/MibImpl.h`) or the `BCmdHandler` (`MultiProtocol/Include/mibh/CmdHandler.h`) classes.

  For this implementation, this macro must be defined.

  In all cases, these properties are declared and initialized by the following macros:

  - **`DECLARE_CMD_MAP`**
  - **`BEGIN_CMD_MAP`**
  - **`CMD_ENTRY`**
  - **`END_CMD_MAP`**
  - **`DECLARE_CONSTRAINT_MAP`**
  - **`CENUM`**
  - **`CENUM_TC`**
  - **`CENUM_ITEM`**
  - **`CRANGE`**
  - **`CRANGE_TC`**
  - **`CRANGE_TCX`**
  - **`BEGIN_CONSTRAINT_MAP`**
  - **`END_CONTRAINT_MAP`**
  - **`DECLARE_TC_CONSTRAINT_MAP`**
  - **`BEGIN_TC_CONSTRAINT_MAP`**
  - **`END_TC_CONTRAINT_MAP`**

Addition macros affecting the build of this source distribution are defined in the header file, `MultiProtocol/Include/sys/Linux/Config.h`. More documentation is presented in this file, but of note are the following:

- **`NDCFG_BIG_ENDIAN`**. When defined, converts byte order specific code to big endian. Default is little endian. This must be defined for this platform.

- **`NDCFG_STL_STRING_SUPPORT`**. When defined, indicates to include STL string support. When not defined, a C++ string implementation is provided. This platform currently does not define this, meaning a local implementation is provided.

- **`NDCFG_NEW_AND_DELETE`**. When defined, provides local implementations of `new` and `delete` based on system specific memory functions. This is defined in this implementation, ultimately based on `malloc()`, `calloc()` and `free()`. (See `NDAlloc()`, `NDFree()` and `NDReAlloc()` implementation in `MultiProtocol//NDSys/Linux/MemImpl.cpp`)

Build Output

The makefiles produce three sets of output files. A set is produced for debug and another for release builds. Each set contains

- A shared object library. (.so.1.0.0 extension). This library is dynamically loadable by the (master) agent, NDAgD.
- A static library archive (.a extension)
- A standalone executable (no extension) with the same name as the project.

Executables

The two executables included with the SDK are described in this section.

NDAgDE

Synopsis: NDAgDE [options]

Description:

This is the NuDesign Multi Protocol SNMP/HTTP/CLI (master) agent. It should reside in the `/bin` directory.

`NDAgD` has the following dependencies on the following shared object support libraries produced from building this SDK:

```
o libNDWebs.so.1
o libNDWebsLinux.so.1
o libNDSubAgE.so.1
o libNDSProxy.so.1
o libNDMibHX.so.1
o libNDVacm.so.1
o libNDNVol.so.1
o libNDSAgent.so.1
o libNDSUdp.so.1
o libNDSocket.so.1
o libNDSocketLinux.so.1
o libNDSnmp.so.1
o libNDSCryp.so.1
o libNDSsi.so.1
o libNDMibHE.so.1
o libNDSys.so.1
o libNDSysLinux.so.1
```

Additionally, the following system library (in ./lib) is required.

```
o libcrypto.so
```

Lastly the sub agent libraries are available to load into the master agent.

```
o libNDMIB2.so.1
```

- `libNDHost.so.1`

can be placed here as well, depending on the configuration of `ndagdext.xnv` (see below).

Command Line Options:

When executed without command line options, it displays a short help and exits. The following command line options are recognized.

-c, run the agent in the console.

-d, run the agent as a background process/service (daemon).

-p, available only when used in conjunction with the '–d', enables a communications pipe that be accessed using '`NDAgClient`'.

-s, stops the agent running in the background.

-l <name>, "loads" the named sub agent (as given in `ndagdext.xnv`) with the agent running in the background.

-u <name>, "unloads" the named sub agent (as given in `ndagdext.xnv`) from the agent running in the background.

-f <name>, freshens ("unloads" and "loads") the named sub agent (as given in `ndagdext.xnv`) from the agent running in the background.

-r, causes the agent running in the background to re evaluate the `ndagdext.xnv` file.

Configuration:

`NDAgD` utilizes two files for nonvolatile storage. These are `ndagd.xnv` and `ndagdext.xnv`. The former controls global configuration items and the latter is used to manage loadable sub agents.

(See xAgentBuilder help documentation for more information on the format of these files.)

As a means of illustrating the information in the `ndagdext.xnv` file, following is the taken from the default content of the file.

```
[ExtensionAgents]
1=NDMIB2
2=NDHost

[NDMib2]
Load=0
Path=/lib/libNDMIB2.so.1
Web=/usr/local/NuDesign/web/NDMIB2

[NDHost]
Load=0
Path=/lib/libNDHost.so.1
Web=/usr/local/NuDesign/web/NDHost
```

The section identified by `[ExtensionAgents]` is used to enumerate the sub agents that are available to load or unload. In this case, two are defined, one each for `NDMib2` and `NDHost`.

The right side of the expression defines a section name in which sub agent specific information is found. In each such named section are three expressions defined as follows:

- `Load=`, takes a value of '`0`' or '`1` '. When the value is '`1`', it indicates that the sub agent is to be loaded the next time the file is evaluated, otherwise it is unloaded. (See the –r command line option for the (daemon) agent)
- `Path=`, specifies the path to the sub agent shared object library.
- `Web=`, specifies to path to the HTML files associated with this sub agent.

(Note: the Web path examples above will **likely** need to be set appropriate for the target configuration. Also note the above examples happen to be for the typical installation location of the xAB Embedded SDK for this platform.

The file, `ndagd.xnv` has a wide range of configurable options. See xAgentBuilder documentation for more inform. The following is taken from a sample XNV file:

```
[Web Agent]
Port=8080
Root=/usr/local/NuDesign/web
HomePage=Home.htm

[Web VACM]
SecurityName=private
SecurityModel=secSNMPv2c(2)
SecurityLevel=noAuthNoPriv(1)
ContextName=""
```

```
[SNMP Agent]
;Version=SNMPv1(0)
Version=SNMPv3(3)
Port=161
```

Where:

`[Web Agent]`	identifies the section associated with the HTTP server.
`Port=`	is the TCP port number on which the HTTP server responds to requests.
`Root=`	is the path to the root directory for the server’s HTML pages.
`HomePage=`	identifies the specific default HTML page .
`[Web VACM]`	identifies the VACM section associated with the HTTP server. This specifies the MIB “view” available to web server. Objects requested in pages that are “in view” will return results. Those outside will not. Changing this configuration should generally not be necessary unless the default configuration provided is invalided by administrative changes to the V3 configuration of the agent.
`SecurityName=`	is the V3 security name the web server uses when accessing MIB objects. This may be any `usmUserSecurityName` currently in the `usmUserTable`. By default `‘private’` is specified. An entry in the `usmUserTable` for this name is also default.
`SecurityModel=`	is the V3 security model the web server uses when accessing MIB objects. This may be one of `secSNMPv1(1), secSNMPv2c(2) or secSNMPv3(3)`.
`SecurityLevel=`	is the V3 security level the web server uses when accessing MIB objects. Default is `noAuthNoPriv(1)`. It could also take the values `authNoPriv(2)` or `authPriv(3)`
`[Snmp Agent]`	identifies the section associated with the SNMP component.
`Port=`	is the UDP port number on which the SNMP agent responds to requests.

`Version=`	is the maximum SNMP version number the SNMP agent will respond to. Default is `SNMPv3(3)`. `SNMPv1(1)` or `SNMPv2c(2)` may also be used.

Default V3 SNMP Community/USM Configuration

By default, the following configurations are available for accessing the agent. Note: the configuration provided is the default configuration for all NuDesign SNMP products and as a result is more or less public information. As such it should only be used for a test and or development deployment. Deploying this configuration in a production environment is **NOT** recommended.

`public`	V1/V2c read-only community string with access to all implemented MIB objects.
`private`	V1/V2c read-write community string with access to all implemented MIB objects.
`public`	**`noAuthPriv`**, V3 read-only USM User with access to all implemented MIB objects. No authorization or privacy passwords.
`private`	**`authPriv`** (MD5, DES), V3 read-write USM User with access to all implemented MIB objects. Auth password: **`privateauth`**, privacy password: **`privatepriv`**.
`md5`	**`authNoPriv`** (MD5), V3 read-write USM User with access to all implemented MIB objects. Auth password: **`md5auth`**, privacy password: none.
`sha`	**`authNoPriv`** (SHA), V3 read-write USM User with access to all implemented MIB objects. Auth password: **`shaauth`**, privacy password: none.
`md5des`	**`authPriv`** (MD5, DES), V3 read-write USM User with access to all implemented MIB objects. Auth password: **`md5desauth`**, privacy password: **`md5despriv`**.
`shades`	**`authNoPriv`** (SHA, DES), V3 read-write USM User with access to all implemented MIB objects. Auth password: **`shadesauth`**, privacy password: **`shadespriv`**.

NDAgClient

Synopsis: `NDAgClient`

Description:

`NDAgClient` is a client application that exposes the command line interface of an agent service that is running in the background as a daemon. There is a limitation of one active client application running at one time. If another client is running and another is started, the user will see a message indicating this. This application uses a file "lock" on a temporary file. The file used is "`/tmp/NDAgD-client-lock`". If for any reason this client application is terminated abnormally (i.e. without invoking the '`exit`' command in the application, see below) then the user may be required to remove the lock file if it was not removed by the terminated session.

Executable Command Line UI

The following are the commands available from both the agent console and `NDAgClient`. Note: details about parameter use are available via the '`?`' and '`help`' commands.

Command	Function
q	Stops the agent.
?	Displays a list of command options.
`help`	Displays help on a specific command.
`clivacm`	Displays or modifies the command line's VACM configuration.
`agents`	Displays, suspends or resumes the SNMP or HTTP agents.
`agparams`	Displays current agent operations parameters.
`get`	Performs a get on an object.
`getnext`	Performs a get-next on an object.
`gget`	Performs a group get of a scalar group of objects. E.g. gget SNMPv2-MIB.system.
`mib`	Displays the list of MIB objects currently available in the agent.
`rget`	Performs a get of a row of a table.
`rgetnext`	Performs a get next of a row of a table.
`samport`	Show the current sub agent registration port.
`set`	Performs a set on an object.
`setti`	Performs a set and increment on an object. (For "Spinlock" variables.)
`snmp`	Display or manage SNMP trace facililty.
`tget`	Performs a get of a table. E.g. `tget RFC1213-MIB.ifTable.`
`vacm`	Display or manage the view access control table.
`walk`	Walk some or all objects in the agent.
`xdll`	Display load status of, load or unload sub agents.

Additionally, there is a circular command history buffer of the last 10 commands executed, available by using the up or down cursor keys.

When using `NDAgClient`, there are two additional operations available. These are:

Command	Function
`exit`	Exits NDAgClient, but leaves the daemon executing.
`!<command>`	Executes the provided `command` in the daemon's context. E.g. !ls –l There is a limitation in this mechanism in that the output from the requested application must go to stdout.

Limitations

The evaluation version of this product has several limitations.

- The evaluation master agent does not support loading sub agent shared objects libraries created by embedded xAgentBuilder.
- The evaluation master agent does not support remote sub agent created by embedded xAgentBuilder.
- **ALL** OCTET STRING values, returned by the agent, whether through SNMP or HTTP, are replaced with an OCTET STRING that contains the ASCII displayable string “NuDesign MultiProtocol (SNMPv3/HTTP/CLI) Evaluation for Linux”.
- The evaluation agent service does not perform the functionality of a SNMP V3 Proxy.

These limitations do not apply to the full licensed version of the SDK.