

NuDesign MultiProtocol Evaluation SDK (9. x) for Desktop Linux Installation and Usage Notes

This document contains confidential and proprietary information. Reproduction and or disclosure through any means is prohibited unless expressed, written consent of authorized representative of NuDesign Technologies Inc. is obtained.

[DOCUMENT HISTORY.....3](#)

[INTRODUCTION.....4](#)

[REQUIREMENTS.....5](#)

[INSTALLATION.....6](#)

[BUILDING WIZARD PRODUCED PROJECTS.....7](#)

[Make Configuration.....7](#)

[Build Targets.....8](#)

[Build Output.....8](#)

[EXECUTABLES.....9](#)

[NDAgDE.....9](#)

[Description:.....9](#)

[Command Line Options:.....10](#)

[Enabling SSL / HTTPS.....11](#)

[Enabling Web Support for Remote Sub Agents.....11](#)

[Configuration:.....13](#)

[Dynamic V3 SNMP Engine ID and USM User Creation.....18](#)

[V3 SNMP EngineID Generation.....18](#)

[USM Table Initialization.....19](#)

[Contents of Sample NDMPDynConfig.xny.....19](#)

[Default V3 SNMP Community/USM Configuration.....22](#)

[NDAgClient.....23](#)

[Description:23](#)

[Console Interface.....24](#)

[Limitations.....26](#)

This document contains confidential and proprietary information. Reproduction and or disclosure through any means is prohibited unless expressed, written consent of authorized representative of NuDesign Technologies Inc. is obtained.

Document History

Date	Rev.	Remarks
2005-03-28	A000	- Original release
2005-04-05	A001	- add to limitations section
2005-04-05	A002	- update limitation section.
2005-05-10	A003	- update requirements.
2006-02-13	A004	- update for 7.0 release
2006-06-14	A005	- update for 7.0.1 release.
2006-07-21	A006	- update for 8.0 release, inclusion of AES128 privacy support.
2006-08-24	A007	- update to describe \$(NDWEBROOT)
2006-09-07	A008	- update to miscellaneous documentation corrections
2007-06-06	A009	- 8.2 update.
2008-03-14	A010	- 8.4 update.
2008-03-20	A011	- add CLI VACM & EnableLocalSecurity descriptions.
2009-02-26	A012	- Update for 8.x. - Change default USM users for new defaults. - Add SSL usage information.
2010-02-19	A013	- 8.8 release update - Add chapter on dynamic engine ID and usm user creation.
2010-03-12	A014	- corrections to chapter on dynamic engine ID and USM user creation - update openssl information.
2011-02-16	A015	- Release 9.0 ipv6 update - Add chapter for web support for remote agents - update "Enabling SSL / HTTPS" section.
2011-05-19	A016	- change to dynamic loading of remote sub agent handling. - Add AgentX support documentation. - addition to chapter on dynamic engine ID and USM user creation.
2011-06-08	A017	- update for 64 bit builds
2011-09-21	A018	- Document corrections only

This document contains confidential and proprietary information. Reproduction and or disclosure through any means is prohibited unless expressed, written consent of authorized representative of NuDesign Technologies Inc. is obtained.

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 documentation is specifically for “Desktop” Linux targets on either 32 (i386) or 64 bit¹ (x86_64) platforms.

The document discusses these main components of this release:

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

¹ There is a different SDK for 32 bit and 64 bit x86 platforms.

This document contains confidential and proprietary information. Reproduction and or disclosure through any means is prohibited unless expressed, written consent of authorized representative of NuDesign Technologies Inc. is obtained.

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.

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

- A desktop Linux development host
- GCC and related tool chain built for the host and target.
- OpenSSL (libcrypto.so) that includes AES² and optionally libssl.so³, if your license includes web services and https is desired.
- Std c++ library 6.x (libstdc++.so.6)

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

² The SDK's agent currently expects to find the **OpenSsl** crypto library named as `'libcrypto.so'` on the library path. If your system provides the library with another name you might consider creating a symbolic link from it to `'libcrypto.so'` as a solution to satisfy the requirement. This should be a viable solution on most systems.

E.g. `'ln -sf libcrypto.so.0.9.8 libcrypto.so'`

³ Similarly, the library `'libssl.so'` is expected to be available if **https** is desired for the daemon's web server. The SDK has been tested with various versions of OpenSsl, including `'0.9.8'` and `'1.0'`

Installation

This release comes in the form of a desktop Linux executable named with a “.bin” file extension.

:

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 will require that you run the installer at a `root` privilege level.

The libraries, executables and web⁴ pages provided in the SDK also need to be available on your target to run. This may be accomplished a variety of ways but the easiest is to create links from the SDK to the appropriate directories. The shared object libraries should be linked (or copied) under the `/usr/lib5` directory and the executables and related configuration files under `/usr/bin`. As an alternative to copying the shared object libraries, you may also consider using the `LD_LIBRARY_PATH` environment variable to point to the SDK library directory.

E.g.

```
export LD_LIBRARY_PATH=/usr/local/NuDesign/MultiProtocol/PCx86/lib
```

or

```
export LD_LIBRARY_PATH=/usr/local/NuDesign/MultiProtocol/PCx86_64/lib
```

The web files can be used in place from the SDK.

⁴ Web pages are included only if the HTTP option has been purchased.

⁵ Or `/usr/lib64` on 64 bit environments.

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. These are defined in each `makefile` as the following macros:

- **NUDESIGNDIR**. This is the path to the SDK includes and libraries. By default this is set to `/usr/local/NuDesign/MultiProtocol/PCx86` or `/usr/local/NuDesign/MultiProtocol/PCx86_64`. Alter this if the SDK was installed in a different location.

In addition, depending on configuration options, you may need to define the following.

- **OPENSSLINCDIR**. This is the path to OpenSSL development include files.

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 be specified on any `make` command line.

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

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, NDAgDE.
- A static library archive (.a extension)
- A standalone executable (no extension) with the same name as the project.
- A standalone remote agent executable⁶ (no extension) with the same name as the project with the “REM” suffix added.

The standalone agent produced by a project requires the <project name>.XNV configuration file produced by the code generator be present in the same directory as executable. The file is normally located in the `/EXE` directory of the project. Copy the file to the directory in which the executable is being executed. For example into `/Debug`.

The standalone agent also requires that it be run at a `root` privilege level if it is configured to open UDP port 161 (SNMP), TCP port 80 (HTTP) or TCP port 443 (HTTPS).

⁶ This project creates a remote (process) sub agent that adheres to the NDT SubAgent protocol. With minor modification to this sub project, you can also produce an AgentX compliant remote agent.

Executables

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

NDAgDE

Synopsis: NDAgDE [options]

Description:

This is the evaluation version of the NuDesign Multi Protocol SNMP/HTTP/CLI (master) agent. The master agent is IPv4/IPv6 compatible. If IPv6 is determined to be running on the host, then the SNMP agent is set to use IPv4 & IPv6 concurrently. The executable 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.0`⁷
- o `libNDWebsLinux.so.0`⁸
- o `libNDSubAgE`⁹.so.0
- o `libNDAgentX`¹⁰.so.0
- o `libNDSProxy.so.0`
- o `libNDMibHX.so.0`
- o `libNDVacm.so.0`
- o `libNDNVol.so.0`
- o `libNDSAgent.so.0`
- o `libNDSUdp.so.0`
- o `libNDSocket.so.0`
- o `libNDSocketLinux.so.0`
- o `libNDSnmp.so.0`
- o `libNDSCrypt.so.0`
- o `libNDSsi.so.0`¹¹
- o `libNDMibHE.so.0`
- o `libNDSys.so.0`
- o `libNDSysLinux.so.0`

⁷ Included only with the HTTP option

⁸ “

⁹ Optional (use) NDT SubAgent management library

¹⁰ Optional (use)) AgentX conforming management library

¹¹ Included only with the HTTP option

Additionally, the following system shared object library is required. This library should be available with the Linux distribution. If not, it is available for download from the internet.

- o `libcrypto.so`
- o `libssl.so` (optional)

Lastly the sub agent library MIB II is available to load into the master agent.

- o `libNDMIB2.so.0`¹²
- o `libNDMIB2-6.so.0`¹³

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

This executable needs to be run with `root` privileges.

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. (default option)
- 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.

¹² `libNDMIB2.so.0` is an implementation based on RFC1213.

¹³ `libNDMIB2-6.so.0` is an implementation based on newer RFCs and also includes Ipv6 specific information. It is recommended the master agent load only this one or `libNDMIB2.so` at once.

This document contains confidential and proprietary information. Reproduction and or disclosure through any means is prohibited unless expressed, written consent of authorized representative of NuDesign Technologies Inc. is obtained.

-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.

Enabling SSL / HTTPS

If your license includes a web server, then there are a few requirements before the web server will enable secure web services.

First, the system specific OpenSSL library `libssl.so` must be on the library load path.

Secondly, the following three certificate files must exist and be in the directory where the daemon is executing from:

- `server.pem`
- `certs/cacert.pem`
- `certs/cakey.pem`

As a convenience, we provide these files as “self signed” certificates. You **SHOULD NOT** release your product with these certificates. They are provided **FOR TESTING ONLY**.

Note: When using the self signed certificates, most web browsers will warn you about the certificate and you must accept the certificate to continue to use HTTPS.

There are a variety of resources on the Internet as to how to “self sign” and certificates in general. The following link <http://www.openssl.org/docs/HOWTO/certificates.txt> and <http://www.openssl.org/docs/HOWTO/keys.txt> are OpenSSL documents relating to certificates and keys.

Lastly, the web server must be configured to enable the secure port. This is the default configuration. (See **SecureMode**, documented below)

Enabling Web Support for Remote Sub Agents

If your license includes a web server and any of the subagents that are going to be managed by the master agent are remote and derived from a xAgentBuilder project, then an additional step is required to provide web support for these

This document contains confidential and proprietary information. Reproduction and or disclosure through any means is prohibited unless expressed, written consent of authorized representative of NuDesign Technologies Inc. is obtained.

agents. An auxiliary object definitions file must be located in the master agent's executable directory. xAgentBuilder creates the necessary file for each sub project when the project is generated. This will be a file with the file extension **.xmi**. The file is placed in the xAgentBuilder project `/web/<project name>` directory and will have the name `<project name>`, under the `/<project name>xmi`. Just copy this file to the master agent's executable directory and restart the master agent.

If you have multiple remote sub agents that implement multiple MIBs, then it is permissible to consolidate the files if you wish. The file name isn't important as long as the file extension remains **.xmi**.

Due to a slightly different registration process on AgentX remote sub agents, a modification step should be made. Therefore for AgentX, if there is only one type of remote subagent, then just copying the XMI file for the subagent is also sufficient. If however, multiple types of remote agents (E.g. one class of remote implements Garage-MIB and another class implements, say MIB2) are to be supported, then the user must make a slight modification to the corresponding XMI files.

For every file XMI file, the user must add the following section:

```
[Module]
Name=ND-GARAGE-MIB
Subtree=1.3.6.1.4.1.4761.99.11
```

Where:

[Module]	identifies the section associated with the follow info.
Name=	is the MIB module name that is associated with the information in the XMI file.
Subtree=	A dot notational OID of the registration point for the remote sub agent. Generally this is one sub OID shorter than the shortest OID in the XMI file. This OID would be created by removing the least significant sub OID. You can confirm the registration point by inspecting the value of sysORID in the sysORTable table corresponding to the registration.

A XMI modified for AgentX is still compatible with the format of XMI files required by the NDT Sub Agent handler.

Configuration:

NDAgDE 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
3=NDMIB2-6

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

[NDHost]
Load=0
Path=/lib/libNDHost.so.0
Web=${NDWEBROOT}/NDHost

[NDMIB2-6]
Load=1
Path=/lib/libNDMIB2-6.so.0
Web=${NDWEBROOT}/NDMIB2-2
```

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. The macro `$(NDWEBROOT)` may be used in the path to specify a path relative to the “Root” web path specified in `ndagd.xnv`.

(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
EnableLocalSecurity=1
SecurePort=443
SecureMode=3

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

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

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

[XCLI]
ExePathAndName=cli
ConfigPath=/ndxcli
Timeout=2

[SAMaster]
Port=10001

[AXMaster]
IP=4
Port=705
```

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 .
- SecurePort=** specifies the port secure web services (https) are provided on. If not provided, the default is 443.
- Secure Mode=** specifies the web service ports to open. When the value is:
- 1: implies open only the web service port controlled by "Port=".
 - 2: implies open only the secure web service port controlled by "SecurePort=".
 - 3: implies open both ports.
- If not provided, the default is '3'.
- EnableLocalSecurity=** when set to '1' enables a user name and password challenge to be issued on the local host if user management is enabled. Default is no entry, meaning local challenges are not issued.
- 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 invalidated 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)`
- ContextName=** is the V3 context name the web/cli server uses when accessing MIB objects. Default is "".
- [CLI VACM]** identifies the VACM section associated with the CLI interface. 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 invalidated by administrative changes to the V3 configuration of the agent.
- (the following items apply to both sections)
- 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/cli 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/cli server uses when accessing MIB objects. Default is `noAuthNoPriv(1)`. It could also take the values `authNoPriv(2)` or `authPriv(3)`
- ContextName=** is the V3 context name the web/cli server uses when accessing MIB objects. Default is "".
- [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.
- [XCLI]** identifies the section associated with the optional XCLI command line component.
- ExePathAndName=** is the path and filename of the xcli command line console that may be invoked from a SSI web sequence.
- ConfigPath=** is the path to the configuration repository for the xcli command line console.
- Timeout=** is the time in seconds that a xcli command line console is allowed to run to process a CLI command sequence.
- [SAMaster]** identifies the section associated with the optional NDT Sub Agent configuration component.
- IP=** (option) specification for IP version handling. Values may be 4 or 6, with 6 being the default when not specified. A value of 6 enables the communication socket for IPv4 and IPv6.
- Port=** is the UDP port number on which the NDT Sub Agent handler responds to requests. If this specification doesn't exist or is set to zero (0), then the library `libNDSubAg.so` is not loaded at start up.
- [AXMaster]** identifies the section associated with the AgentX configuration component.
- IP=** (option) specification for IP version handling. Values may be 4 or 6, with 6 being the default when not specified. A value of 6 enables the communication socket for IPv4 and IPv6.
- Port=** is the TCP port number on which the AgentX handler responds to requests. If this specification doesn't exist or is set to zero (0), then the library `libNDAgentX.so` is not loaded at start up.

Dynamic V3 SNMP Engine ID and USM User Creation

This pre-configuration phase is conducted in response to the agent finding the file "**NDMPDynConfig.xnv**" in the same directory as the executable, at startup. Since the contents of this file may contain sensitive information, after processing, the file will be erased, so any backup copy you've created becomes the only record of this information. The intention of this facility is to assist an administrator in the initial deployment of a master agent.

A sample file's contents can be viewed at [this link](#).

This processing has two components: **EngineID** generation and **USM UserTable** creation.

V3 SNMP EngineID Generation

EngineID generation is based on the value of '**EngineIdGenMode**' in **NDMPDynConfig.xnv** and has seven different values, implying different modes of creation:

Gen Mode	Function
0	(default) do not change current engine id.
1	Use the current IPv4 address of the first non-loop back interface found. E.g. 80:00:12:99:01:C0:A8:00:01 (given an IP address of 192.168.0.1)
2	Use the current IPv6 address of the first non-loop back interface found. (not implemented in this version.)
3	Use the current MAC address of the first non-loop back interface found. E.g. 80:00:12:99:03:01:02:03:04:05:06 (given a MAC address of 01:02:03:04:05:06)
4	Use the current text provided. E.g. 80:00:12:99:04:12:99:04:6E:64:74:2D:69:6E:63:2E:63:6F:6D (given EngineIdUseText below)
5	Use the current octets provided. See EngineIdUseOctets below E.g. 80:00:12:99:05:7f:00:00:01:ab:cd:ef:bc (given EngineIdUseOctets below)
6	Use the current octets provided, but do not insert the method into the resulting engine id. See EngineIdUseOctets below E.g. 00:00:12:99:7f:00:00:01:ab:cd:ef:bc (given EngineIdUseOctets below)

This document contains confidential and proprietary information. Reproduction and or disclosure through any means is prohibited unless expressed, written consent of authorized representative of NuDesign Technologies Inc. is obtained.

Note: With methods 1 to 5, the method number is inserted into the **EngineId** after the **EngineIdPrefix**.

An example copy of **NDMPDynConfig.xnv** can be found in the **doc** directory of the SDK.

USM Table Initialization

The USM User Table initialization allows an initial USM table to be put in place and any adjustments to related tables, such as the VACM tables, will also be made.

There are only two modes of operation of this function. To do nothing, leaving all tables as is. Alternately, it will replace the USM table with the information supplied in this XNV file, making any necessary changes to other tables, such as the VACM tables. You will note the data supplied for the USM User Table will look like:

```
4=shades SHA(3) DES(2) "shadesauth" "shadespriv" "grpAll"
```

includes an optional field ("**grpAll**" in this case) which is used while setting up the VACM tables.

It is important to note that if you elect to install a new USM User Table, any previous table information will be lost.

Contents of Sample NDMPDynConfig.xnv

```
; *****  
;  
; Configuration file to dynamically create engineid and  
; corresponding USM table.  
;;  
; The v3CommunityTable is modified if any entries are found to contain the  
; the old engine id.  Entries are updated with the new engine id.  
;;  
; This file is erased on completion.  Make a backup if you need it.  
;;  
;  
; *****  
;;  
; EngineIdGenMode  
; =====  
;;  
; integer value.  Valid values: 0 - 6.  
;;  
; Where:  
;;  
; 0 - (default) do not change current engine id.  
;;  
; 1 - Use the current IPv4 address of the first non-loop back interface found.
```

This document contains confidential and proprietary information. Reproduction and or disclosure through any means is prohibited unless expressed, written consent of authorized representative of NuDesign Technologies Inc. is obtained.

```
;;
; E.g. 80:00:12:99:01:C0:A8:00:01 (given an IP address of 192.168.0.1)
;;
; 2 - Use the current IPv6 address of the first non-loop back interface found.
; ( not implemented at this time.)
;;
; 3 - Use the current MAC address of the first non-loop back interface found.
;;
; E.g. 80:00:12:99:03:01:02:03:04:05:06
; (given a MAC address of 01:02:03:04:05:06)
;;
; 4 - Use the current text provided.
;;
; E.g. 80:00:12:99:04:12:99:04:6E:64:74:2D:69:6E:63:2E:63:6F:6D
; (given EngineIdUseText below)
;;
; 5 - Use the current octets provided. See EngineIdUseOctets below
;;
; E.g. 80:00:12:99:05:7f:00:00:01:ab:cd:ef:bc (given EngineIdUseOctets below)
;;
; 6 - Use the current octets provided, but do not insert the method into the
; resulting engine id. See EngineIdUseOctets below
;;
; E.g. 00:00:12:99:12:7f:00:00:01:ab:cd:ef:bc (given EngineIdUseOctets below)
;;
; Note: With methods 1 to 5, the method number is inserted into the engine id
; after the EngineIdPrefix.
;;

; EngineIdPrefix
; =====
;;
; 4 octet string, specifying first four octets of the engine id.
;;
; The first bit of the first octet will be set to '1' if the generation mode
; has a value 1 - 5.
;;
; default prefix: "00:00:12:99", implying the default after first bit set:
; "80:00:12:99" for methods 1 - 5.
;;

; USMGenMode
; =====
;;
; integer value. Valid values: 0 - 1.
;;
; Where:
;;
; 0 - do not change USM table
;;
; 1 - (default) create table from provided USM table. Note:
; this option will delete existing entries in the table.
;;

; SysNameMode
; =====
;;
; Optional: integer value. Valid value: 1
;;
; When set to '1' the service will be configured to use the current host name
; as the value for sysName.0. Any other value or the absence of the parameter
```

This document contains confidential and proprietary information. Reproduction and or disclosure through any means is prohibited unless expressed, written consent of authorized representative of NuDesign Technologies Inc. is obtained.

```
; will leave the setting of this mode and hence, sysName.0 to the current (or
default) mechanisms.
```

```
;;
```

```
[DynConfig]
EngineIdGenMode=3
EngineIdPrefix=00:00:12:99
EngineIdUseOctets=7f:00:00:01:ab:cd:ef:bc
EngineIdUseText="ndt-inc.com"
USMGenMode=1
SysNameMode=1
```

```
;;
```

```
; The last column in the table below is used to create a row in the
; vacmSecurityToGroupTable for each USM entry
```

```
;
```

```
[DynUsmUserTable]
```

```
;
```

```
;index=UserName AuthProtocol PrivProtocol AuthPassword PrivPassword [GroupName:
default=grpReadOnly]
```

```
1=md5 MD5(2) none(1) "md5auth" "" "grpAll"
```

```
2=sha SHA(3) none(1) "shaauth" "" "grpAll"
```

```
3=md5des MD5(2) DES(2) "md5desauth" "md5despriv" "grpAll"
```

```
4=shades SHA(3) DES(2) "shadesauth" "shadespriv" "grpAll"
```

```
5=public none(1) none(1) "" "" "grpReadOnly"
```

```
6=shaaes SHA(3) AES128(4) "shaaesauth" "shaaespriv" "grpAll"
```

```
7=md5aes MD5(2) AES128(4) "md5aesauth" "md5aespriv" "grpAll"
```

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. (Status: deprecated)
sha	authNoPriv (SHA), V3 read-write USM User with access to all implemented MIB objects. Auth password: shaauth , privacy password: none. (Status: deprecated)
md5des	authPriv (MD5, DES), V3 read-write USM User with access to all implemented MIB objects. Auth password: md5desauth , privacy password: md5despriv .
shades	authPriv (SHA, DES), V3 read-write USM User with access to all implemented MIB objects. Auth password: shadesauth , privacy password: shadespriv .
md5aes	authPriv (MD5, AES128), V3 read-write USM User with access to all implemented MIB objects. Auth password: md5aesauth , privacy password: md5aespriv .
shaaes	authPriv (SHA, AES128), V3 read-write USM User with access to all implemented MIB objects. Auth password: shaaesauth , privacy password: shaaespriv .
md5nopriv	authNoPriv (MD5), V3 read-write USM User with access to all implemented MIB objects. Auth password: md5noprivauth , privacy password: none.
shanopriv	authNoPriv (SHA), V3 read-write USM User with access to all implemented MIB objects. Auth password: shanoprivauth , privacy password: none.

This document contains confidential and proprietary information. Reproduction and or disclosure through any means is prohibited unless expressed, written consent of authorized representative of NuDesign Technologies Inc. is obtained.

NDAgClient

Synopsis: `NDAgClient`

Description:

This application provides a daemon maintenance console interface to the daemon that is currently running. In addition to the console interface documented in the section ‘Console Interface’, one additional console command is available. This is the ‘**exit**’ command that causes this client application to exit.

There are two modes of operation for this application. When the application is started with no additional command line information, the application presents an interactive console interface similar to that which is available when the agent is run as a console application.

When the application is started with additional information on the command line, the additional information is presented to the daemon and executed. When a response is received from the daemon, the application outputs the daemon’s response to `stdout` and exits.

Example 1:

```
NDAgClient get sysDescr.0
```

The above causes the ‘`get sysDescr.0`’ to be executed by the daemon. The response should be something like:

```
sysDescr = NuDesign Multiprotocol Agent
```

Redirecting `stdout` from this is a little “tricky”, since the rest of the command line is presented to the daemon to execute. However the solution is simple, have the client command line execute in it’s own shell then redirect the output of that shell however you want.

Example 2:

```
(NDAgClient get sysDescr.0) > tmp.out
```

The above still causes the ‘`get sysDescr.0`’ to be executed by the daemon, however using this syntax, it is run in a child shell, with the end of the command line being the ‘0’. The (shell) response is redirected to `tmp.out`. The contents of `tmp.out` should be something like:

```
sysDescr = NuDesign Multiprotocol Agent for Linux
```

In this mode of operation, you can use the client from a script to perform some set of operations repetitively. Also note that in this mode the 'exit' command is implied, so this is no need to use it.

Note: 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 will also happen when the agent is being run from the console. This application (and agent in console mode) uses a file "lock" on a temporary file to accomplish this. 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) then the user may be required to remove the lock file if it was not removed by the terminated session.

NDAgClient should be run with `root` privileges.

Console Interface

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.
AgentXPort	Shows the current TCP port on which AgentX requests are being processed.
agents	Displays, suspends or resumes the SNMP or HTTP agents.
agparams	Displays current agent operations parameters.
evaltime	Displays evaluation time remaining.
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.)

This document contains confidential and proprietary information. Reproduction and or disclosure through any means is prohibited unless expressed, written consent of authorized representative of NuDesign Technologies Inc. is obtained.

`snmp` Display or manage SNMP trace facility.
`tget` Performs a get of a table. E.g. `tget RFC1213-MIB.ifTable`.
`vacm` Display or manage the view access control table.
`version` Display daemon version.
`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
<code>exit</code>	Exits <code>NDAgClient</code> , but leaves the daemon executing.
<code>!<command></code>	Executes the provided <code>command</code> in the daemon's context. E.g. <code>!ls -l</code> There is a limitation in this mechanism in that the output from the requested application must go to <code>stdout</code> .

Limitations

The evaluation version of this product has several limitations.

- The evaluation agent service will function for a period of 30 days after the installation of the SDK. If the agent service is running when the time expires, it will shutdown automatically. If you attempt to start the executable after that time, a message will appear indicating the expiry time has been reached.
- Standalone agents produced with the evaluation SDK will operate for 30 days after the installation of the SDK. After that time, executables produced with it will exit shortly after starting.
- The evaluation agent service does not perform the functionality of a SNMP V3 Proxy.
- The ./MIB implementation is not provided with the evaluation.

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