ASN1C

ASN.1 Compiler
Version 6.0
Installation Guide

Objective Systems, August 2006

Author's Contact Information:

Comments, suggestions, and inquiries regarding this document may be submitted via electronic mail to info@obj-sys.com.

Installing the ASN1C Compiler Software

Version 6.0x of the ASN1C compiler software is packaged in two separate distribution units:

- A *System Development Kit* (SDK) unit that contains the compiler and development libraries, and
- One or more run-time deployment units containing optimized binary libraries for deployment of a finished application.

The following sections contain installation instructions for Windows and UNIX versions of the ASN1C distribution files. This procedure is the same for all of the different configurations.

The last section describes techniques to help port code generated by earlier versions of the compiler to version 6.0x.

Microsoft Windows Distribution

The Microsoft Windows version of ASN1C is distributed either on a CD and/or electronically over the Internet. The distribution files are self-extracting, executable setup files. The format of the filename of the SDK unit is as follows:

```
ac<L>v60xw32sdk.exe
```

where *<L>* would be replaced with a single-letter language code. Possible values are *p* for C/C++, *j* for Java, or *s* for C#.

Run-time deployment packages have the following format:

```
rt<L>v60xw32<TYP>.exe
```

where *<L>* would be replaced with a single-letter language code and *<TYP>* would be replaced with a code for the package type. Possible values for the language code are *p* for C/C++, *j* for Java, or *s* for C#. The type code would contain information on whether the library is licensed per-host (limited) or unlimited, whether source code is included or not, and what encoding rules are supported.

For example: the following would be an ASN1C v6.00 run-time deployment kit for Java:

```
rtjv600w32ubb.exe
```

In this case, the *ubb* on the end stands for unlimited, binary, BER/DER.

ASN1C System Development Kit (SDK) Installation

The procedure to install the ASN1C compiler and run-time libraries is as follows:

1. Double-click the SDK installation program kit executable filename. This is the filename in the format described above.

2. Follow the setup program instructions.

3. You should have received a node-locked license file (*osyslic.txt*) to enable the compiler to run on a given node. Copy this file into one of the following locations:

a. One of the directories specified in your PATH environment variable, or

b. In a different directory and create a new environment variable name OSLICDIR that points to this directory, or

c. Into the same directory as the ASN1C compiler executable file (asn1c.exe)

If installation was successful, you should have both a graphical user interface (GUI) compilation wizard available as well as a command line version of the compiler. The GUI wizard can be tested by starting the application and entering the data requested in each of the dialog prompts. The command-line version of the compiler can be tested as follows:

1. Open an MS-DOS or other command shell window.

2. Change directory (cd) to the compiler root directory. The default directory in the setup script is **c:\acv<version>** where **<version>** is the version number of the compiler. For example, **c:\acv600** is the default root directory for version 6.00 of the compiler.

3. Enter **.\bin\asn1c** from the command line prompt. You should see a usage display of compiler command line options (these options are discussed later). This indicates the compiler is properly installed and working.

If you get a message indicating the license file could not be found, please review the procedure in step 3 above to make sure it is installed in the proper location.

You should include the <target>\bin (ex. c:\acv600\bin) directory in your PATH environment variable in order to run the compiler from anywhere.

ASN1C Run-time Deployment Kit Installation

The deployment run-time packages can be installed at any time after the SDK is installed. They are not necessary for basic program development. They should be used when an application is ready to be deployed. To install, do the following:

1. Double-click the installation program kit executable filename. This is the filename in the format described above.

2. Follow the setup program instructions. Note that the root directory for the installation should be the same as was specified for installation of the SDK package described earlier.

The result will be additional library subdirectories of the form *lib_opt* added to the directory hierarchy. These contain the optimized libraries. To link with these libraries, either the makefile(s) or project files used to link the application must be changed, or the subdirectories must be renamed. For example, the existing *lib* subdirectory could be renamed *lib_nonopt* and *libopt* could then be renamed to *lib*.

Documentation Installation

Up-to-date documentation is always available online at: http://www.obj-sys.com/docs/documents.shtml

If a CD was purchased with the software, all applicable PDF document files for a particular configuration of the software will be available on the CD.

Compiling and Linking ASN1C Generated Code

When building code generated by the ASN1C compiler, you will need to have one or more run-time source directories in your include path to compile the generated code with a C or C++ compiler. The run-time source directories are *rtsrc* (common), *rtbersrc* (BER/DER), *rtpersrc* (PER), and *rtxersrc* (XER). To link, you will need the *lib* or *lib_opt* subdirectory in your library path. This is where all of the library files are located.

Testing the C or C++ Run-time Components

The default C or C++ run-time libraries for Windows were built with Microsoft Visual C++ V6.0. Other libraries are available that have been built with the Borland C++ compiler (v5.5) and with the Microsoft Visual C++ v7.1 and v8.0 (.NET) compilers. If you have the version of the product that includes run-time source code, you can rebuild the run-time libraries using any ANSI-standard C or C++ compiler.

You can verify operation of any of the different run-time libraries by executing the sample programs. These can be found in the *sample_ber*, *sample_der*, and (optionally) the *sample_per* or *sample_xer* subdirectories.

For example, we will assume that you installed ASN1C for C/C++. To test the BER C++ encode/decode capabilities, do the following:

1. Change directory (cd) to the **.\cpp\sample_ber\employee** subdirectory. Execute the **nmake** command to build the writer and reader sample programs. **nmake** is utility program that comes with Visual C++. It may also be necessary to execute a Microsoft batch file named **VCVARS32.BAT** to set the path information so that the **nmake** utility can be found. (Important note: this assumes you are using Microsoft Visual C++ on your PC. Some PC specific include and library directories in the makefile may need to be changed to get the samples to work on your system. See the README.txt file for further details).

2. Execute the **writer.exe** program to encode the sample record. The results of the encoding will be dumped to the screen and saved in a file called *message.dat*.

3. Execute the **reader.exe** program to read and decode the contents of the *message.dat* file. This program will read the encoded record into memory, decode it, and then print the contents of the generated structure variable to standard output.

Testing PER is similar:

1. Change directory (cd) to the **.\sample_per\employee** subdirectory. Execute the **nmake** command to build the writer and reader sample programs.

2. Execute the **writer.exe** program to encode the sample record. The **–a** switch can be used to encode a record using aligned PER. The **–u** switch encodes a record using unaligned PER. The results of the encoding will be dumped to the screen and saved in a file called *message.dat*.

3. Execute the **reader.exe** program to read and decode the contents of the *message.dat* file. The **–a** or **–u** switch must be the same as that specified when the writer program was executed. This program will read the encoded record into memory, decode it, and then print the contents of the generated structure variable to standard output.

This test can be repeated for XER as well by going to the **sample_xer\employee** subdirectory and repeating the above sequence of steps.

Per-host License Deployment Issues

If you purchased run-time libraries that allow for unlimited redistribution, then all run-time license checking would have been removed from these libraries and all that must be done to deploy is to make certain that the code is linked with these libraries.

If per-host run-time licensing was selected, then there are two choices as to how applications are deployed:

1. The run-time license information can be directly compiled into the application and then the application deployed without any external files, or

2. An external binary license file can be deployed with the application to allow it to run on the licensed hosts.

The first choice is done by default every time ASN.1 source files are generated and the resulting code compiled and linked into an application. Information from the *osyslic.txt* file is transferred to a C header file called *rtkey.h* and this information is included in the generated code. This allows the application to run on all licensed hosts.

The second option is applicable mainly in situations when a finished application is to be run on different hosts than were originally licensed. This can happen if host names are changed, or additional hosts are added at a later date. It might not be practical in these situations to rebuild the application in this case. The alternative is therefore available to create a binary license file and deploy it with the application to allow it to run on the newly licensed hosts. The procedure to do this is as follows:

1. Use ASN1C to generate an *rtkey.dat* file by issuing the following command:

   ```
   asn1c -genlic
   ```

   The *rtkey.dat* file will be created in the directory where you issued the command.

2. Copy the *rtkey.dat* file onto the computers on which you want to run your application.

3. Set the *ACLICFILE* environment on these computers to point at the full path to the *rtkey.dat* file.

In the case of Java, the procedure is different. In this case, the deployed *asn1rt.jar* file must contain up-to-date license information. This is done by executing the *setkey.bat* script for Windows or *setkey.sh* script for Linux/UNIX in the Java subdirectory after the license file for ASN1C is installed. This must be done whenever the *osyslic.txt* license file is updated.

UNIX Distribution

Installation of the Linux / UNIX version of ASN1C is similar to the Windows version except that the distribution files are packaged as gzipped tar files. The format is the same as Windows except that the extension is *.tar.gz* instead of *.exe.*

To install, do the following:

1. Copy the distribution file <distfile> to the top-level directory where the compiler is to be installed.

2. Unzip using the GNU unzip tool:

   ```
   gunzip <distfile>
   ```

3. Untar the file using the following command:

   ```
   tar xvf <distfile>
   ```

   This will create a directory tree structure with 'asn1c-v<version>' as the root.

4. You should have received a node-locked license file (*osyslic.txt*) to enable the compiler on a given node. Copy this file into one of the following locations:

   a. One of the directories specified in your PATH environment variable, or

   b. In a different directory and create a new environment variable name OSLICDIR that points to this directory, or

   c. Into the same directory as the ASN1C compiler executable file (asn1c.exe)

To test if the compiler installation was successful, do the following:

1. Change directory (cd) to the compiler root directory. The default directory in the setup script is **./asn1c-v<version>** where **<version>** is the version number of the compiler. For example, **./asn1c-v600** is the default root directory for version 6.00 of the compiler.

2. Enter **./bin/asn1c** from the command line prompt. You should see a usage display of compiler command line options (these options are discussed later). This indicates the compiler is properly installed and working.

After installing the compiler, you can modify your operating environment to access the compiler executable files from anywhere. To do this, you will need to add the path to the compiler to your PATH environment variable or add a link to it from a standard binary directory (such as /opt/bin). This is optional and not required for any of the run-time sample programs to work. They are all set up to use relative directory paths to access the compiler and libraries.

Deployment run-time packages should be installed by repeating steps 1 through 3 above. The packages should be unpacked in the same base root directory as the original SDK files. This will cause *lib_opt* subdirectories to be added to the various C and C++ directories in the installation.

Testing the C or C++ Run-time Components

The basic C or C++ run-time libraries for UNIX are typically built with the GNU gcc/g++ compiler and/or the standard native compiler provided by the manufacturer of a particular type of UNIX (for example, aCC for HP-UX). Two symbolic links are used within the c or cpp subdirectory to select the version of the run-time libraries to be used. They are as follows:

- lib
- platform.mk

By default, these are set to point at the GNU gcc/g++ version of the run-time libraries for a particular platform. This is easily changed by deleting the links and setting them to point at another run-time library. For example, on Solaris, to use the native compiler libraries one would set lib -> libCC and platform.mk -> platform.CC:

```
ln –s libCC lib
ln –s platform.CC platform.mk
```

You can verify operation of any of the different run-time kits by executing the sample programs. These can be found in the different sample directories (*sample_ber*, *sample_der*, *sample_per* and/or *sample_xer* depending on what run-time kits were installed).

To test the encode/decode capabilities for any of the encoding rules, execute the sample programs as described in the section on Windows installation.

Backwards Compatibility

ASN1C has undergone significant changes for release 6.0, including some that affect code generation. Since several type names have changed, two files have been packaged with the kit to help in porting code generated with prior versions: `rtport.pl` and `asn1compat.h`.

The former will process a specified file and rename the types contained in it. The latter contains macro definitions that convert old names to the new naming scheme.