ASN1C

ASN.1 Compiler
Version 6.8
Installation Guide

Copyright Notice

Author Contact Information

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

Table of Contents

Change History

Date	Author	Description
March 2015	EM	Initial release of 6.8 documentation.

Installing the ASN1C Compiler Software

Version 6.x of the ASN1C compiler software is packaged in two or three separate distribution units, depending on how ASN1C is licensed:

- A System Development Kit (SDK) unit that contains the compiler and development libraries.
- One or more run-time deployment units containing optimized binary libraries for deployment of a finished application.
- A floating license server application (available for Windows and various UNIXes).

The floating license server application is only provided for customers who have purchased a floating license configuration of the compiler.

The following sections describe installation instructions for Windows and UNIX versions of the ASN1C distribution files. The final section describes techniques for users of 5.x-era versions to port code to 6.x-era versions of ASN1C.

Some basic familiarity with Windows and UNIX commands is assumed in the following sections. In particular, it is assumed that users understand how to install programs and unzip archives, so we will spend most of the text discussing how to configure and test the applications once the base installation is complete.

Windows Installations

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 `acv6NNw32sdk.exe` or `acv6NNx64sdk.exe`. (Throughout this document, `w32` is freely interchangeable with `x64`; we will use `w32` by convention.) NN is replaced with the minor and patch release version numbers, as in `acv680w32sdk.exe`.

The format of the runtime library filename is `rtLv6NNw32TYP.exe`. L is replaced with *p* (C/C++), *j* (Java), or *s* (C#). NN retains the same meaning as in the SDK kits. TYP is replaced with a three-letter code indicating the type of the kit: limited or unlimited, binary or source, and the encoding rule set. For example, `rtjv680w32ubb.exe` indicates a Java binary runtime kit with unlimited redistribution rights (*u* for unlimited; *b* for binary; *b* for BER/CER/DER).

The floating license package is versioned separately from ASN1C and is distributed in an executable self-extracting installation file (like other Windows kits). The package is named `rlmsvrw32.exe`.

Installing the RLM Floating License Server

Objective Systems uses the Reprise License Manager for license checking in the ASN1C SDK. Licenses come in both floating and node-locked configurations. Users who request a floating license may host the license server

remotely with Objective Systems or locally on their own machine. This section details a local installation of the RLM license server. Node-locked licenses are provided by Objective Systems, and their installation is covered in future sections.

The server can be started from a command line or run as a service. It requires a license file (provided by Objective Systems, usually called `server.lic`) to run. Users should not confuse this license file with the license provided for node-locked kits, or the corresponding `client.lic` file used to tell ASN1C to communicate with the license server.

The `rlm` command is invoked like this (command-line options can be in any order):

```
% rlm [-c license_file] [-dlog [+]logfile]
      [-nows] [-ws port] [-x [rlmdown|rlmremove]]
      [-install_service] [-service_name sname]
```

The `-c license_file` option specifies which license file to use. This option overrides the setting of the `RLM_LICENSE` environment variable. The `license_file` parameter can be a directory containing license files, all of which will be processed.

The `-dlog logfile` specifies the pathname for the server debug log. If `logfile` is preceded by the `+` character, the log file will be appended to; otherwise it is overwritten. All ISV servers will write their output to the same log file specified in the `-dlog` option.

The `-nows` and `-ws port` options control the operation of the embedded Web Server. The `-nows` option instructs the RLM server not to start the embedded web server. The `-ws port` option instructs the RLM server to use `port` as the port number for the web server. If the RLM server cannot bind the web server port (5054 by default), it will exit.

The `-x [rlmdown | rlmremove]` option controls whether the `rlmdown` and/or `rlmremove` commands will be processed by the server. Specifying `-x` by itself will disable both commands. Specifying either command name after the `-x` will disable just that command.

The `-install_service` and `-service_name sname` options are used to run the RLM server as a service under Windows, as in the following example:

```
rlm -install_service -dlog [+]logfile [-service_name sname] <rlm runtime args>
```

The `<rlm runtime args>` are those specified above; when the service starts, it will pass these additional arguments to the RLM server.

A complete command line might look like this:

```
rlm -install_service -service_name rlm-xyz -dlog
   c:\logs\server.log -c c:\licenses\xyz.lic
```

This installs the RLM server as a service under the name `rlm-xyz`. When started via the Services control panel or at boot time, RLM will be passed the `-c c:\licenses\xyz.lic` argument, and it will write its debug log information to the file `c:\logs\server.log`.

Installed RLM services are also deleted with the `rlm` program. Services must be stopped using the Windows Service control panel application before they can be deleted. Deleting a service only removes it from the Windows service database; it does not uninstall the `rlm` executable or associated license file(s).

The service can be removed by calling the executable like this:

```
rlm -delete_service [-service_name sname]
```

where `sname` is an optional name for the installed service. If not specified, service_name defaults to `rlm`. If `service_name` contains embedded whitespace, it must be enclosed in double quotes.

ASN1C Windows SDK Installation

There are three types of SDK installations: floating, node-locked, and evaluation.

An ASN1C distribution that uses the floating license server works by contacting the server to obtain a license key. A license file (`client.lic`) that facilitates communication with the floating license server is provided by Objective Systems.

An ASN1C node-locked distribution works by reading the contents of a license file (`asn1c.lic`) supplied by Objective Systems. This file contains details to validate the installation to ensure it is running on a permitted host.

Evaluation licenses are provided as a key to users who download the software in the format of `NNNN-NNNN-NNNN-NNNN`.

In all three cases, the basic installation is the same: double-click the executable installation package downloaded from the Objective Systems website and follow the setup program installation instructions.

If a license file was provided (either for the floating or node-locked configuration), this file should be downloaded and copied into the ASN1C installation's `bin` subfolder. By default, this will be `c:\acv6NN\bin`. The license may also be copied to a folder pointed to by the `RLM_LICENSE` environment variable if desired.

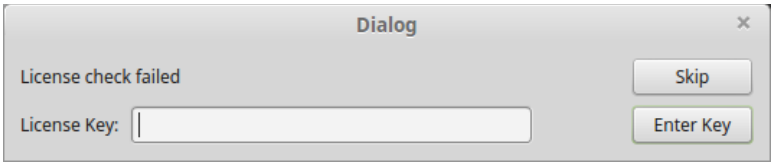
In the evaluation installation, the license key may be applied using the command line or by entering it into the ASN.1 Editor.

To use the command line, open the command prompt application and execute the following commands:

```
C:\> cd \acv6NN\bin
C:\acv6NN\bin> asn1c -lickey NNNN-NNNN-NNNN-NNNN -nousage
```

If no errors are reported, you can test the license by executing `asn1c` without any command-line arguments; a usage display will appear.

To use the ASN.1 Editor, navigate to the ASN1C application from the Start Menu or double-click the application icon on the Desktop. (This was installed by default by the setup program.) Once started, the editor will open up the previously saved project or a blank screen. If the product is unlicensed, you will be prompted to enter the license key, as in the following dialog:



The license key `NNNN-NNNN-NNNN-NNNN` can be entered in the *License key* field and applied. (If a text license was provided, the license must be applied as described above: it should not be pasted into this dialog box.)

To test whether the license key was applied, you can repeat the process described in the command-prompt or open a sample project file included with the ASN1C installation kit.

Finally, you may wish to modify the system-wide `PATH` environment variable in order to run ASN1C from any location.

ASN1C Run-time Deployment Kit Installation

The run-time deployment kits packages come in two varieties: limited (node-locked) and unlimited. Both are installed the same way. Run-time deployment kits contain libraries optimized for space and performance.

Installation proceeds in the same way as the SDK installation: double-click the executable package and follow the setup installation program instructions. By default the root installation folder will be the same. The kit contains optimized libraries in the `lib_opt` folders. The sample programs provided in the SDK link against libraries in the `lib` folder, so users may find it easier to rename this folder to `lib_nonopt` and rename the `lib_opt` folder to `lib`.

In a floating configuration with limited run-time deployment, users may need to use a binary license (`rtkey.dat`) supplied by Objective Systems. The binary license contains host name information for the limited deployment; the host names may also be embedded within the client license (`client.lic`); in this case, the binary license is not needed. See Deploying a Run-time Library with a Binary License File for more details.

Unlimited deployment run-time libraries do not need any additional license files.

Testing the C or C++ Runtime Components

The default C and C++ run-time libraries for Windows were built with Microsoft Visual Studio 2010. Included with Visual Studio 2010 are libraries built for Visual Studio 2008 and 2012 and libraries built for Cygwin and MinGW. Add-on packages are available for Microsoft Visual Studio 2005 and 2013, Microsoft Visual C++ v6.0,

and Borland C++ (version 5.5). If you have purchased 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`, `sample_per`, `sample_json`, `sample_xer` and `sample_xml` subdirectories.

For example, we will assume that you installed ASN1C for C/C++. To test the default Visual Studio 2010 BER C++ encode/decode capabilities, open a Visual Studio 2010 command prompt and execute the following commands:

```
C:\> cd \acv6NN\c\sample_ber\employee
C:\acv6NN\c\sample_ber\employee> nmake
                                        [...]
C:\acv6NN\c\sample_ber\employee> writer
                                        [...]
C:\acv6NN\c\sample_ber\employee> reader
```

The output from running these commands has been stripped for sake of clarity.

When executing the `writer` program, a file called `message.dat` will be created; its contents will be dumped to the screen as well. The `reader` program reads in `message.dat` and prints the formatted contents to the screen.

Testing the PER libraries is analogous. The employee sample is stored in `acv6NN\c\sample_per\employee`. Of special note in the PER samples is the ability to read or write *aligned* or *unaligned* PER messages through the use of command-line switches:

```
C:\acv6NN\c\sample_per\employee> writer -u
                                        [...]
C:\acv6NN\c\sample_per\employee> reader -u
                                        [...]
```

By default, the sample programs will encode using aligned PER. Using the `-u` switch will turn on unaligned encodings. Use `-a` to force aligned encodings.

Testing XER encoding and decoding proceeds in the same way.

UNIX Installations

We treat all UNIX-derived operating systems the same for the purposes of this section. Users of commercial UNIXes, Linux, and MacOS X should be able to install the software following roughly the same steps. Each platform has its own idiosyncrasies, so installation procedures will vary.

Installing the RLM Floating License Server

The RLM server is packaged as a `.tar.gz` archive for Linux, Linux-64, and MacOS X. As in the Windows installation, users will need a license file to run the server (usually called `server.lic`). The server can be run manually from the command line or at boot time.

Command-line options are detailed in the section Installing the RLM Floating License Server. They are identical between Windows and UNIX installations.

Running the RLM server at boot time usually requires the addition of an init script. Examples are provided in the README provided with the RLM distribution package. Users are likewise encouraged to consult platform-specific documentation to determine how best to set up boot time services, since configurations between various platforms may differ considerably.

Objective Systems License Server Installation

For other UNIXes (AIX, HP-UX, and Solaris), Objective Systems has produced a custom floating license server. More details may be found in the `README.txt` file that accompanies the installation package.

Like the RLM server, the OSys license server is packaged in a `.tar.gz` archive that may be unpacked anywhere on the system. It can be run as a background process or by writing an init script for execution at boot time. The server must be licensed in order to run; the `osyslic.txt` file is provided for users who need this server.

The server may be run from the command line like this:

```
./licServer -url http://[host IP]
```

The `host IP` is the IP address of the host that has been licensed to run the server.

ASN1C UNIX SDK Installation

The SDK kits are distributed as `.tar.gz` archives and may be unpacked anywhere on the system. Common UNIX convention is to install commercial products in the `/opt` directory hierarchy. The naming conventions for these kits are the same as those described in the Windows Installations section.

Kits delivered for Linux, Linux-64, and MacOS X use the Reprise License Manager and so are licensed in a manner similar to the Windows kits. An RLM license file (usually `asn1c.lic` or `client.lic`) is provided for node-locked or floating development kits, while an evaluation license key will be provided for customers who are evaluating.

Kits delivered for other UNIXes use Objective Systems' custom license manager and use a file called `osyslic.txt`.

Please note that the ASN.1 editor is not included with all UNIX kits since it depends on the Qt libraries. Objective Systems compiles these libraries manually for each platform to ensure that the GUI is portable. On systems where this is not possible (AIX, HP-UX, and Solaris), no GUI is provided.

Unpacking the SDK

In order to unpack the kit, follow these directions:

1. Copy the installation kit to the top-level directory in which the compiler should be installed.
2. Unzip the package using the GNU unzip tool:

   ```
   $ gunzip <installation kit>
   ```

3. Unpackage the files using tar:

   ```
   $ tar xf <installation kit>
   ```

On systems equipped with GNU tar, the final two steps may be combined into a single command:

```
$ tar xzf <installation kit>
```

Unpacking the kit will result in a new directory hierarchy rooted at `asn1c-v6NN`. Users may wish to add the `bin` subdirectory to the system-wide `PATH` in order to run ASN1C from any location.

On systems that do not have Qt installed, it may be necessary to modify the `LD_LIBRARY_PATH` (or its equivalent) so that the shared libraries provided by Objective Systems will be loaded when running the editor.

Installing the SDK License

The license provided by Objective Systems comes in one of three varieties: an RLM license file, a license activation key, or an Objective Systems license file. The RLM license file and the activation key are used on Linux, Linux-64, and MacOS X systems. Other UNIXes use the Objective Systems license file (`osyslic.txt`).

The RLM license file should be copied to the `bin` subdirectory of the ASN1C installation. The `RLM_LICENSE` environment variable can also point to the directory in which the license file is located.

The license activation key is used predominantly for evaluation versions of the software. The activation key is a 16-digit number formatted as `NNNN-NNNN-NNNN-NNNN`. ASN1C can be activated from the command-line or from the ASN.1 editor.

The command to activate the license is:

```
$ ./asn1c -lickey NNNN-NNNN-NNNN-NNNN -nousage
```

No output will appear on the screen if a successful activation occurred. The license key may also be entered in the editor in the same way described in ASN1C Windows SDK Installation. The editor will need to be run from the command-line if a desktop launcher was not manually created:

```
$ ./acgui
```

In other UNIXes like AIX, HP-UX, and Solaris, Objective Systems provides the `osyslic.txt` file to allow ASN1C to run. This file should be copied into the `bin` subdirectory of the ASN1C installation. It may also be copied to a directory pointed to by the `OSLICDIR` environment variable.

To test whether the license was properly installed, run ASN1C from the command line or start the GUI:

```
$ ./asn1c
$ ./acgui
```

If a usage message appears in the first case, or the GUI reports no license errors in the second case, the installation was successful.

ASN1C UNIX Run-time Deployment Kit Installation

ASN1C run-time deployment kits come in two varieties: limited (node-locked) and unlimited. Both are installed the same way. The installation procedures are analogous to those used for the UNIX SDK.

Run-time deployment kits come with libraries that have been optimized for performance and space. These libraries are included in the `lib_opt` directories included in the specific language subdirectories of the installation.

Limited deployments may require the use of a binary license (`rtkey.dat`). See Deploying a Run-time Library with a Binary License File for more details for how to deploy the library in this case.

Testing the C or C++ Run-time Components

The C and 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: `lib` and `platform.mk`.

By default, these are set to point at standard compiler 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 change from using libraries compiled with CC:

```
rm lib
rm platform.mk
ln -s libCC lib
ln -s platform.CC platform.mk
```

Soft links are used to preserve a common build infrastructure. Users are free to modify the platform-specific definitions as needed to customize compilations for their application builds.

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).

Assuming that ASN1C was installed in `/opt/asn1c-v680`, testing the C BER employee sample program would look like this:

```
~$ cd /opt/asn1c-v680/c/sample_ber/employee
```

```
/opt/asn1c-v680/c/sample_ber/employee$ make
                                       [...]
/opt/asn1c-v680/c/sample_ber/employee$ ./writer
                                       [...]
/opt/asn1c-v680/c/sample_ber/employee$ ./reader
```

The output generated by running these programs has been stripped for sake of clarity. The `writer` application creates a file called `message.dat` and dumps the output to the terminal. The reader application loads the file and prints it out in a brace-formatted display.

The PER and XER samples can be built analogously. Of special note in the PER samples is the ability to read or write *aligned* or *unaligned* PER messages through the use of command-line switches:

```
/opt/asn1c-v680/c/sample_per/employee$ ./writer -u
                                       [...]
/opt/asn1c-v680/c/sample_per/employee$ ./reader -u
                                       [...]
```

By default, the sample programs will encode using aligned PER. Using the `–u` switch will turn on unaligned encodings. Use `-a` to force aligned encodings.

Compiling and Linking Generated Code

ASN1C comes with a variety of options to assist with compiling and linking generated code. Of principal interest are the `-genmake` and `-vsproj` options (for generating makefiles and Visual Studio projects, respectively). ASN1C can also generate Ant build files for Java using the `-genant` switch.

The generated makefiles may not integrate with users' build systems, so please note the following:

1. The include paths should include `rtsrc`, `rtxsrc` (if developing with C/C++; this directory is not used for C#), and the runtime source directories needed for their rulesets (`rtbersrc`, `rtpersrc`, `rtxersrc`, `rtxmlsrc`).
2. The library paths should point at the `lib` or `lib_opt` directory.

When developing using Java, the class path must include `asn1rt.jar` in order to properly locate compiled classes. This can be specified on the command-line by using the `-cp` or `-classpath` switches or else by modifying the system-wide `CLASSPATH` environment variable.

Users who have purchased an unlimited runtime library must link against this library in order to remove license-checking from their application. Users who are linking dynamically do not need to relink their applications, but may instead replace the shared libraries (DLLs or .so files) with those contained in the unlimited redistribution package.

ASN1C Licensing Procedures

It is not unusual during the course of application development, deployment, and maintenance that customers find it necessary to update and or upgrade their licenses. This may happen as you add new deployment hosts, upgrade the software, or transition between different platforms. The procedures described in the following sections should be applied if needed to ensure that the SDK and run-time libraries work properly.

While the format differs slightly, each supported language embeds the run-time key in generated source files. The key is checked by the node-locked runtime libraries as the program executes. Unlimited runtime libraries do not check the key at all.

Because the key is embedded in generated sources, it is recommended that users regenerate their source code when receiving a new license. If this is not possible or is undesirable, a binary license may be generated and used (as described in following sections).

Updating an SDK License

Users may be supplied with a new SDK license file in several circumstances:

1. When switching from an evaluation license to a permanent one.
2. When upgrading from an older version of the software to a newer version.
3. When migrating from one license style (e.g., `osyslic.txt`) to another (e.g., `asn1c.lic`).
4. When adding new SDK or run-time deployment hosts.

When updating the license, users should take care to remove the old license files and adjust the `RLM_LICENSE` or `OSLICDIR` environment variables if needed.

After updating the license, users are advised to regenerate their code and recompile their applications so that the new license information is properly embedded. This is the easiest way to ensure that the application will run properly on the target hosts.

Deploying a Run-time Library with a Binary License File

When it is impossible or undesirable to regenerate code, users can generate a binary license file for use with their deployment libraries. (This is only needed with per-host deployments; unlimited runtime libraries do not check the license, but users must link against these libraries in order for their applications to run properly.)

To do this, ASN1C can be run with the `-genlic` option, which generates a file called `rtkey.dat`. This file must be installed with the user's application in order for it to execute.

For the C/C++ run-time an environment variable named ACLICFILE must be defined to point to the file. This environment variable must point at the fully-qualified name of the file; for example:

```
~$ export ACLICFILE=/opt/asn1c-v680/rtkey.dat
```

For the Java run-time the ACLICFILE environment variable also works. With Java two other options are to place the file anywhere in the application's class path or to place the file into the asn1rt.jar file.

For the C# run-time the ACLICFILE environment variable also works. With C# another option is to place the file anywhere in the system's PATH (i.e., into any of the folders indicated by the PATH environment variable).

When replacing a binary license file, users are advised to purge the host system of old licenses and to adjust the `ACLICFILE` environment variable as needed.

Deploying a Run-time Library with a Text License File

Starting with ASN1C v6.7.2 the run-time libraries can also be deployed with a text license file named osyslic.txt that contains the ASN1C run-time key and would be furnished by Objective Systems.

For the C/C++ run-time an environment variable named ACLICFILE must be defined to point to the file. This environment variable must point at the fully-qualified name of the file; for example:

```
~$ export ACLICFILE=/opt/asn1c-v672/osyslic.txt
```

For the Java run-time the ACLICFILE environment variable also works. With Java two other options are to place the file anywhere in the application's class path or to place the file into the asn1rt.jar file.

For the C# run-time the ACLICFILE environment variable also works. With C# another option is to place the file anywhere in the system's PATH (i.e., into any of the folders indicated by the PATH environment variable).

When replacing a text license file, users are advised to purge the host system of old licenses and to adjust the `ACLICFILE` environment variable as needed.

Further Documentation

Up-to-date documentation on ASN1C's SDK and runtime kits is always available on Objective Systems' website at http://www.obj-sys.com/support.php. Historical documentation and change logs are also available for users of older versions.

Users who purchased a CD distribution kit will find all applicable PDFs for their version of ASN1C on that CD.

Backwards Compatibility

This section contains notes for users considering upgrading to a new version.

Version 6.0.x

ASN1C underwent significant changes between version 5.8x and 6.0x, including some that affect code generation. These changes will affect users who are attempting an upgrade: the two versions are not API compatible.

Many of the changes can be procedurally applied. To help assist users in this effort, we have included the `rtport.pl` script in newer kits to reduce some of the labor associated with an upgrade. We have also included the `asn1compat.h` file for C/C++ users; this can be included in your sources to help assist a transition between versions.

Version 6.1.x

Version 6.1.x introduced changes to Java and C# code generation to help help reduce the use of system resources in enumeration-heavy specifications. These changes are mostly transparent to users, but those who are working directly with enumerated types will need to access those fields using special accessor methods; in most cases, all that is needed is to add parentheses after the identifier. These changes are discussed in finer detail in the Java and C# User's Manuals.

Version 6.3.x

Version 6.3.x introduced a modification to Java code generation so that it would embed the run time key in generated code instead of requiring users to execute a script (`setkey.bat` or `setkey.sh`) as in previous versions. Users upgrading to version 6.3 from 6.2 will need to regenerate their code in order to properly use the new library.

Version 6.5.x

In version 6.5, Objective Systems adopted the use of the Reprise License Manager for Windows, Linux, and MacOS X installations. Users who are upgrading from earlier versions will need to acquire a new license file from Objective Systems in order to validate their installations. For all other platforms, the legacy license format (`osyslic.txt`) has been retained.

Version 6.6.x

Beginning with version 6.6, we adopted the use of sized types in our C/C++ runtime libraries. These changes will introduce API and ABI incompatibilities on systems where sized types do not alias the same types used previously. For example, if a function's parameters included `OSUINT32`, and `size_t` is defined to be an unsigned 32-bit integer, it is possible that no incompatibilities would arise. However, if `size_t` were defined as an unsigned 64-bit integer, incompatibilities would likely occur.

Our usual policy is to recommend that users synchronize generated code with the runtime libraries, rather than mixing and matching.

Version 6.7.x

The changes to size types in 6.6.x continued in 6.7, and our recommendations for upgrading remain the same.

Version 6.8.x

In version 6.8.x, Objective Systems made a minor change that will affect ABI compatibility in generated code for users who are upgrading from previous versions.

In particular, we changed the signatures of our generated copy functions in C/C++ to return an integer status instead of returning void. This change was adopted to signal when a copy function failed due, for example, to an invalid memory allocation.

We expect the impact of such a change to be rather minimal, but users distributing dynamic libraries should be advised that the exported symbols will probably change.

Support

Questions related to installation and support for ASN1C may be directed via email to support@obj-sys.com.