EnLIGHTTened HARC RM Perl Dependences

Version 1.9

EnLIGHTTened Project
CCT at LSU

Jon MacLaren

August 7, 2007
# Contents

1 Introduction 3

2 Prerequisites 3
   2.1 OpenSSL 3
      2.1.1 Linux 3
      2.1.2 AIX, using cc not gcc 4
   2.2 Xerces-C 4
      2.2.1 AIX with cc, not gcc 5

3 Getting Started with CPAN 5

4 Cryptography Modules 6
   4.1 Net::SSLeay 6
   4.2 Crypt::SSLeay 6
      4.2.1 Recent problems 6

5 POE and components 7
   5.1 POE Framework 7
   5.2 POE Components 7

6 XML::Xerces 8
   6.1 AIX, cc not gcc 9

7 Odds and ends 9
   7.1 AIX, cc not gcc 10

8 Testing 10
1 Introduction

This document describes the installation of the Perl modules which are needed to support the running of a Resource Manager. The set of modules required does not vary from RM to RM.

The RMs should be run using Perl 5.8.x. I haven't tested the RMs on anything older than 5.8.6 for a long time now. Note that if you are using AIX, with the vendor compiler, you must use 5.8.8 (due to type problems in CORE/reentr.inc).

All of the modules required are available through CPAN. Outside of this, there are only two pieces of software required, openssl and xerces-c.

2 Prerequisites

When building the following pieces of software, you must use the same compiler as was used to build your perl. This means that on platforms with a vendor compiler and gcc that usually the vendor compiler is the right one to use. (This also means that on such platforms, your Globus or DVT OpenSSL will not work properly.)

Note that on many 64-bit systems that also support 32-bit executables, that Perl will be 32-bit.

If you are using Linux, you may be able to pick up RPMs containing some/all of the modules listed below. I have not tried this approach, nor gotten any reports of people doing this. If you've tried this, I'd be interested to hear about it, at mailto:maclaren@cct.lsu.edu One list of RPMs can be found at http://dag.wieers.com/rpm/packages/—they are indexed by replacing the double-colons in the package name with hyphens, and prefixing “perl”, e.g. for Net::SSLeay, look for perl-Net-SSLeay.

2.1 OpenSSL

HARC RMs use X509 Certificates for security. For the cryptography perl modules (Crypt::SSLeay and Net::SSLeay) to work correctly, the need to be built against a version of OpenSSL 0.9.7 (API changes mean that these modules do not work with either 0.9.6 or 0.9.8). OpenSSL 0.9.7l and OpenSSL 0.9.7m both work.

It is possible to use a previously installed version of OpenSSL. However, the header (or include) files must also be present. And also see the constraints above.

2.1.1 Linux

When you are building the perl cryptography modules, you may get the following message:

    @gprel relocation against dynamic symbol

(Some variants of this message include an explicit message stating you need ”-fPIC”.) If so, you will need to build your own OpenSSL. Download the latest version of OpenSSL 0.9.7, e.g. 0.9.7m. Do the ./config step as you would normally, specifying any prefix flags as per the instructions, but put ”-fPIC” on the end, e.g.:
Then do make, make tests, make install as usual. This will build a version of the library which you can correctly build against.

2.1.2 AIX, using cc not gcc

There is a problem with fips-1.0 during make install, but this can be gotten around easily using gmake instead of make. For example:

```bash
./Configure --prefix=/work/default/maclaren/openssl-install/openssl-0.9.7m --openssldir=/work/default/maclaren/openssl-install/openssl-0.9.7m aix-cc
gmake
gmake test
gmake install
```

2.2 Xerces-C

Later we'll install XML::Xerces, which is currently at version 2.7.0. This is based on xerces-c 2.7.0. Instructions for building from source are given here. However, you may be able to find RPMs for this, e.g. from http://dag.wieers.com/rpm/packages/xerces-c/. Note that you will need to install both xerces-c and xerces-c-devel RPMs; please stick to version 2.7.0.

The installation for your platform will vary, but for a 32-bit Linux installation, I did the following:

1. `curl -O http://www.apache.org/dist/xml/xerces-c/source/xerces-c-src_2_7_0.tar.gz`
2. `gunzip -c xerces-c-src_2_7_0.tar.gz | tar xf -`
3. `export XERCESCROOT=$HOME/xerces-c-src_2_7_0`
4. `cd $XERCESCROOT/src/xercesc/`
5. `./runConfigure -p linux -c gcc -x g+ -b 32+`
6. `gmake`

There is no need to install xerces-c system-wide, but you can if you wish. Whether you do or not, you will need to make sure that the directory where the library finishes up is part of your dynamic library search path. This is controlled by the environment variable LD_LIBRARY_PATH on many systems, but it's LIBPATH on AIX, e.g.

```bash
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$XERCESCROOT/lib
```

You will need to make sure that this is the case, both while installing XML::Xerces below, and when installing/running the resource manager. When installing XML::Xerces, the variable XERCESCROOT must also be set.
2.2.1 AIX with cc, not gcc

For the configuration, I used:

```shell
./runConfigure -p aix -c xlc_r -x xlc_r -b 32
```

Using `xlc_r` should be the same as using `cc_r`, but if you use `cc_r`, then there are a couple of odd string library calls missing. I am not sure why. Using `xlc_r` will give you a lot of warnings about duplicate symbols, but this is not a real problem (according to the mailing lists).

3 Getting Started with CPAN

One hint before you start. If you’re doing this as root, you might want to do a `umask 022` before you start.

It’s best to do this as root. If you can’t, you can install the Perl modules under the account where the Resource Manager will be run. In this case, follow the instructions at:


Note that you should also make sure that you set up the environment variable `PERL5LIB` to point to the location of the libraries before you install anything. Otherwise CPAN may try to install some modules multiple times.

All of the packages are available through CPAN. The first step is to run the perl CPAN shell, and configure it if you haven’t done this before. Type:

```perl
perl -MCPAN -e shell
```

It’s up to you how you configure CPAN, itself. Personally, I usually change `/usr/bin/make` to `/usr/bin/gmake`.

You will probably wish to update CPAN, particularly if you can’t recall commands using the Up-Arrow key; it should improve the process somewhat. To do this, at the prompt, type:

Then, I installed Bundle::CPAN, to update CPAN, as suggested by typing:

```bash
cpan> install Bundle::CPAN
```

This can install a lot of things. You should now quit CPAN, and restart. There will probably be extra config items to put in. (If you are non-root, you should be careful to set “--install_base” correctly.) Once you’ve added these, you will need to type:

```bash
cpan> o conf commit
```

Now you’re ready to install the individual Perl modules.
4 Cryptography Modules

These are the hardest things to get working.

4.1 Net::SSLeay

At the CPAN prompt, type:

```
cpan> install Net::SSLeay
```

You will be asked to supply the location of your OpenSSL installation.

This module should build and test alright. However, if you’ve got a firewall that prevents general web access, then the network tests will fail. As there is no easy way to turn off the network tests, you may need to force the installation as follows:

```
cpan> install Net::SSLeay
```

But you must check that it builds and does the non-network tests correctly first!

If you get problems building the module, check the notes above in Section 2.1, and maybe try building your own OpenSSL.

4.2 Crypt::SSLeay

At the CPAN prompt, type:

```
cpan> install Crypt::SSLeay
```

This will again ask you for the location of your OpenSSL installation. This module is kinder with its tests, and will ask if you wish to perform the non-network tests. If you don’t have free access to the web from your machine, you should say no.

4.2.1 Recent problems

The last time I tried to install Crypt::SSLeay, there was a silly problem with the include path, causing the SWIG parts of the module not to compile. If the first compile errors complain about not being able to locate the OpenSSL header files, then do the following:

```
cpan> look Crypt::SSLeay
```

This will drop you into a shell in the build area. Edit the (generated) Makefile (which contains all your other stuff, like installation dirs), and find the line reading something like this:

```
INC = -I/work/maclaren/openssl-install/openssl-0.9.7m/include/openssl
```
Note that this definition appears higher up in the Makefile in a commented out section—please ensure you edit the correct line (the one that does not start with ‘#’).

Now remove the trailing "openssl":

```
INC = -I/work/maclaren/openssl-install/openssl-0.9.7m/include
```

Return to the shell, and do:

```
make
make test
make install
```

Although some tests are skipped, those that run should all work.

## 5 POE and components

POE (Perl Object Environment) is quite a large module. It is “a framework for creating event-driven multitasking programs in Perl”. For more information, see:

```
http://poe.perl.org/?What_POE_Is
```

### 5.1 POE Framework

To install this, type:

```
cpan> install POE
```

This may need to install a number of other components which it depends upon. When asked if you want to add these to the list of things to install, you should say yes (the default).

I have had one problem with the POE tests, on AIX, but have seen this reported for OpenBSD also. The symptom is that:

```
t/30_loops/20_poll/wheel_sf_unix
```

fails 9 out of 12 tests. This is a test of UNIX sockets. This does not affect the HARC RM, which does not use these (it uses TCP sockets). In this case, you can force the installation.

### 5.2 POE Components

We now install the following POE components.
cpan> install POE::Component::Client::Keepalive

cpan> install POE::Component::Client::HTTP

cpan> install POE::Component::Server::HTTPServer

cpan> install POE::Component::Server::SimpleHTTP

Note that you will be asked two questions when installing the SimpleHTTP package, namely:

SSL support? Answer yes, which will install POE::Component::SSLify

Prefork support? Unless you are on AIX, answer yes, which will install IPC::Shareable

It is possible, when installing install POE::Component::Client::Keepalive for one of the tests to fail, due to a timing problem. Specifically, you may see the following (reported with Version 0.1000):

```
t/09_timeout...........ok 1/6
    t/09_timeout...........NOK 5/6# Failed test 'second connection failed'
    # at t/09_timeout.t line 75.
    Use of uninitialized value in numeric eq (==) at t/09_timeout.t line 76.
    # Failed test 'second connection request timed out'
    # at t/09_timeout.t line 76.
    t/09_timeout...........NOK 6/6# Looks like you failed 2 tests of 6.
    t/09_timeout...........dubious
        Test returned status 2 (wstat 512, 0x200)
    DIED. FAILED tests 5-6
        Failed 2/6 tests, 66.67% okay
```

The source of the test contains the following comment:

```
# TODO - The 0.01 second timeout assumes it will give the component
# enough time to create a wheel but not establish a connection.
# This is a bold assumption, and it may lead to false failures.
```

If this is the only failure when testing this module, you should be safe to force the installation.

6 XML::Xerces

Before installing this from CPAN, you should make sure that either:

- XERCESCROOT is set as in Section 2.2, if you built from source, and did not install xerces-c system-wide; or,

- you set XERCES_INCLUDE and XERCES_LIB if you installed xerces-c after building, or if you installed xerces-c from RPMs.

In both cases, you should also ensure that your dynamic library search path (e.g. LD_LIBRARY_PATH) includes the directory containing to xerces-c library.

Next, start CPAN, and type:
6.1 AIX, cc not gcc

The installation will fail during compilation because of a type mismatch in Xerces.cpp (int * vs. bool *), currently at line 1901. When you get this error do:

```perl
sv_vsetpvfn(perl_get_sv("@", TRUE), fmt, strlen(fmt), &args, Null(SV**), 0, Null(bool*));
```

becomes:

```perl
sv_vsetpvfn(perl_get_sv("@", TRUE), fmt, strlen(fmt), &args, Null(SV**), 0, Null(int_*));
```

Now, before running this again, there is one more change to make. Edit the Makefile, and alter the LDDLFLAGS definition. Insert "-lm" and "-lC_r" in front of the "-lc_r", e.g.

```plaintext
LDDLFLAGS = -bhalt:4 -bM:SRE -bI:$(PERL_INC)/perl.exp -bE:$(BASEEXT).exp -bnoentry -lpthreads -lm -lC_r -lc_r
```

Return to the shell, and do:

```bash
  gmake
gmake test
gmake install
```

All the tests should work (a few may be skipped, though).

7 Odds and ends

There are a few remaining modules which are required.

```bash
  cpan> install HTML::SimpleLinkExtor
  cpan> install Date::Manip
  cpan> install Math::Round
  cpan> install LWP::Parallel::UserAgent
  cpan> install Data::UUID
```
7.1 AIX, cc not gcc

The AIX C compiler doesn’t like the (slightly naughty) C++ style comments used in UUID.h. When the install command fails, do:

```
cpan> look Data::UUID
```

Edit UUID.h, and change the C++ style comments (currently on lines 8 and 96 only) to C-style comments. Still in the shell do:

```
gmake
gmake test
gmake install
```

8 Testing

To test that the Perl modules have been installed, you can run the test program test-perl-install. If the environment variable HARC is set to the top-level directory of the CVS area (the ‘negotiation’ directory), then the script is at:

```
$HARC/rm-service/scripts/test-perl-install
```

First, if your Perl is not located at /usr/bin/perl, then edit the script and change the line reading:

```
PERL=/usr/bin/perl
```

Next, run the program by following the following instructions:

```
cd $HARC/rm-service
chmod 755 ./scripts/test-perl-install
./scripts/test-perl-install
```

If the PERL5LIB environment variable is setup correctly, and you’ve got everything installed, you should see the following:

```
2007-05-25 18:05:56 rm-server: Reading acceptor-mapfile
2007-05-25 18:05:56 rm-server: Done reading acceptor-mapfile
2007-05-25 18:05:56 rm-server: Reading grid-mapfile
2007-05-25 18:05:56 rm-server: Done reading grid-mapfile
2007-05-25 18:05:56 rm-server: SCBatch->new called
2007-05-25 18:05:56 rm-server: CHECK_INTERVAL remains at default 30
2007-05-25 18:05:56 rm-server: DELAY_INTERVAL remains at default 30
2007-05-25 18:05:56 rm-server: ABORT_INTERVAL remains at default 150
2007-05-25 18:05:56 rm-server: Starting up on port 9393
2007-05-25 18:05:56 rm-server: Running POE kernel
```
In the case that port 9393 is not free on your machine, this will be followed by an error:

    POE::Component::Server::SimpleHTTP tried 5 times to create a Wheel and is giving up...
    at /Library/Perl/5.8.6/POE/Component/Server/SimpleHTTP.pm line 331.

But in either case, if you get this far, then the Perl libraries are all in place. We can do further tests, but this requires further configuration, an X509 credential, and a working set of HARC Acceptors (see the RM Configuration and Installation Guide for details). If the RM is running, you can type <CTRL>-C to stop it.