art - Bug #11039

What do we need to do to make --print-available-modules and --module-description work?

12/01/2015 12:23 AM - Christopher Backhouse

Status:	Closed	**Start date:**	12/01/2015
Priority:	Normal	**Due date:**	
Assignee:	Kyle Knoepfel	**% Done:**	100%
Category:		**Estimated time:**	2.00 hours
Target version:	1.17.05	**Spent time:**	1.75 hour
Occurs In:		**Experiment:**	NOvA
Scope:	Internal	**SSI Package:**	art

Description

The --print-available-modules and --module-description switches don't work well with our SRT build.

nova --print-available-modules gives me a large number of errors like this:

```
In: getModuleType
---- Configuration BEGIN
  Library specificaton "": does not correspond to any library in LD_LIBRARY_PATH of type "module"
---- Configuration END

In: getFilePath
---- Configuration BEGIN
  Library specificaton "": does not correspond to any library in LD_LIBRARY_PATH of type "module"
---- Configuration END

In: getDescription
---- Configuration BEGIN
  Library specificaton "": does not correspond to any library in LD_LIBRARY_PATH of type "module"
---- Configuration END
```

before providing the module list. Built in modules have Type and Source location, but all of our modules have [error] and [not found] for those.

The result for nova --module-description is consistent, error messages and blank fields. Nor does it show anything in the Allowed configuration block for a module I converted over to fhicl::Table.

Interestingly services seem to fare better, they get correct Type and Location fields. I haven't tried converting a service configuration yet.

What are the requirements for these features to work? Do we need to set environment variables or have the module libraries named according to a certain convention?

This is all for art 1.17.03

Associated revisions

Revision a3946f22 - 12/09/2015 10:46 AM - Kyle Knoepfel

Implement fix for issue #11039

Revision 77e3c61b - 12/14/2015 02:11 PM - Kyle Knoepfel

Implement fix for issue #11039

Revision 81189539 - 12/21/2015 02:32 PM - Kyle Knoepfel

Implement fix for issue #11039

History

#1 - 12/01/2015 08:05 AM - Kyle Knoepfel

- Status changed from New to Assigned

- Assignee set to Kyle Knoepfel

#2 - 12/01/2015 08:45 AM - Kyle Knoepfel

- Tracker changed from Support to Bug

- Estimated time set to 2.00 h

Short answer: NOvA uses a library-naming scheme that does not support plugin-ambiguity resolution.

Full answer: Consider a module whose .so is named:

- libdir1_MyModule_module.so

To enable this module in an art job, a user has two configuration options:

```
module_type: "MyModule"         # or
module_type: "dir1/MyModule"
```

The first specification is the "short spec", and the second is the "long spec" or full spec.

Using the long spec is essential for the following situation. Consider the source code paths for two modules:

- dir1/dir2/MyModule_module.cc
- dir1/dir3/MyModule_module.cc

Following the encouraged naming pattern, the .so's for these modules is:

- libdir1_dir2_MyModule_module.so
- libdir1_dir3_MyModule_module.so

This way, users can specify

```
module_type: "dir1/dir2/MyModule"  # or
module_type: "dir1/dir3/MyModule"
```

in order to disambiguate the modules. It would not be possible otherwise, and art, in fact, throws an exception if there is an ambiguity that cannot be resolved. NOvA, however, does not follow the naming pattern as mentioned above -- thus, no long spec is available.

Whenever the --print-available-modules/--module-description facilities are called, the long spec is used -- thus the unhelpful printout you are seeing. For services, the long spec is never used by art, and the --print-available-services/--service-description facilities therefore use the short spec, which is why you saw no problems with the services.

Bottom line: Even though NOvA does not name .so's in a way that allows for ambiguity resolution, there's no reason why the --print-available-modules/--module-description facility should require the long specification. This will be adjusted.

#3 - 12/01/2015 10:44 AM - Christopher Backhouse

OK, so short answer is we should sit tight? Though I thought there were not going to be any more 1.17 releases? so it could be a while before we're able to use this.

Isn't a short spec equivalent to a long spec in the case that the library happens not to be in any directory? Imagine we just prefixed "a_" to all our library names, regardless of what directory the source came from.

#4 - 12/01/2015 04:14 PM - Rob Kutschke

A reminder about a possibly related issue. About a month ago I submitted issue #10410 which requests a command lineoption to change the behaviour of module disambiguation to first match wins.

#5 - 12/01/2015 04:20 PM - Christopher Backhouse

I'd always assumed, without really thinking about it, that the behaviour was "multiple matches = error". But SRT test releases rely on first-match-in-LD_LIBRARY_PATH behaviour and definitely work reliably, so it sounds like the premise of your issue (and all the replies to it) is wrong?

#6 - 12/02/2015 08:56 AM - Kyle Knoepfel

It is the stated policy that art continues its feature development on top of a 1.18 branch. However, that does not preclude us from back-porting feature requests to 1.17, which we have already done. Also, I would characterize this issue as a bug, and we typically implement and release bug fixes in patch versions of the same minor series (1.17, in this case).

As for "first-match-in-LD_LIBRARY_PATH wins" (which is a tangential discussion), that is typically true for any .so. However, for plugins (i.e. modules, services, sources, etc.), art explicitly loads them through its plugin manager, which throws an exception if a given library spec can be matched to more than one .so file. For example, consider the two libraries:

- dir1/libMyModule_module.so
- dir2/libMyModule_module.so

The library spec for both is 'MyModule'. If dir1 and dir2 are both on LD_LIBRARY_PATH, art (more specifically cetlib::LibraryManager) will throw an exception.

If you are seeing behavior contrary to this, please file a bug report giving a specific example so we can recreate it. Last I checked, there were no duplicate library specifications in novasoft (based on the list obtained running nova --print-available-modules. So unless you added a module whose library name clashed with another, where both directories were included on LD_LIBRARY_PATH, I would not expect you to see any exceptions being thrown.

As Rob already mentioned, there is a feature request for allowing the first .so name match to be loaded.

#7 - 12/02/2015 01:37 PM - Christopher Backhouse

OK, I'd be happy to see this fixed in a bugfix 1.17 release.

Whenever a user adds a package to their test release the modules in that package appear twice, as:

$SRT_PRIVATE_CONTEXT/lib/$SRT_SUBDIR/libMyModule.so and $SRT_PUBLIC_CONTEXT/lib/$SRT_SUBDIR/libMyModule.so

where $LD_LIBRARY_PATH is $SRT_PRIVATE_CONTEXT:$SRT_PUBLIC_CONTEXT:...

There's no error in this case, just first module wins (which is exactly what we want).

Perhaps this is OK because the dir1/dir2 part in your example is identical (ie they're located in the same place relative to the element of $LD_LIBRARY_PATH they're under).

#8 - 12/09/2015 11:00 AM - Kyle Knoepfel

- Status changed from Assigned to Resolved

- % Done changed from 0 to 100

The fix has been implemented with commit art:a3946f2. There has been an additional request to include updates to fhiclcpp in a future 1.17 release. We would likely fold this bug fix in when that happens.

Regarding the first-match-in-LD_LIBRARY_PATH behavior. You are correct, Chris; I was mistaken. Indeed, the first match for a given .so wins. The disadvantage to not using the recommend library-naming scheme (as alluded to previously) is that if a user specifies a certain module, they may not realize that they will be getting a module from an LD_LIBRARY_PATH directory that is sooner in the list than what they intend (if the modules have the same base name). This undesirable situation can be avoided by using the recommend library-naming pattern, as the LibraryManager is then able to detect an ambiguity, throw an exception and ask the user to resolve the ambiguity via a full specification.

#9 - 12/17/2015 01:51 PM - Kyle Knoepfel

- Target version set to 1.17.05

- SSI Package art added

- SSI Package deleted ()

#10 - 12/23/2015 08:53 AM - Kyle Knoepfel

- Status changed from Resolved to Closed