Using art/LArSoft services outside art

DUNE software and computing

David Adams
BNL

September 21, 2015

Updated 23Sep2015 10:00 EDT

Introduction

Art and LArSoft provide many services

- TFileService – manage root files and objects
- RandomNumberGenerator – generate randoms, manage seeds
- Geometry – access detector geometry info
- DetectorProperties – detector info
- LArProperties – LAr info
- TimeService – Timing info
- And many more…

Not all user code will be in art modules

- How can code outside art modules make use of the existing services?
- I start by trying to make direct use of these services without any modification of the art or LArSoft code

Why use services outside art?

Motivation to use services outside art framework

- Convenient to reuse the services outside framework
  - E.g. in Root scripts, user main program for data analysis, event display, …
    - Including component tests
  - Avoid need to duplicate the effort that went into writing the art services
- User code using services can work inside and outside framework
  - So we don't have to branch for different interfaces depending on context
  - And so avoid ugly code and extra coding (see following page)
  - Reduce the need for testing in multiple environments
- New user services can be written as art services
  - Easily take advantage of fcl configuration
  - They will also work in art

Ugly code

Like to avoid code like this for creating histograms:

```
#ifdef ARTFW
#include “art/Framework/Services/Optional/TFileService.h”
#include “art/Framework/Services/Registry/Servicehandle.h”
#else
#include “TFile.h”
#endif
.
.
.
#ifdef ARTFW
  art::ServiceHandle<TFileService> pfs;
  TH1* ph1 = pfs->make<TH1F>(“hist1”, “My hist”, 50, 0, 100);
#else
  Tfile::Open(“myfile.root”, “CREATE”);
  TH1* ph = new TH1F(“hist1”, “My hist”, 50, 0, 100);
#endif
.
.
.
```

Or introducing another layer which hides such code.

Can we do it?

Art developers were not encouraging

- "Art services are intended to be used in the art framework"

But also not strongly discouraging

- Didn't say not to try and provided some guidance
- Discussion scheduled for the next art developers meeting

So I went ahead and gave it a try

- First goal was a simple main configuring, loading and using services
- Success!
  - I was able to load and use the art TFileService and
  - Next loaded and used the RandomNumberGenerator
    - Then Geometry, DetectorProperties, LArProperties and TimeService
- Code is a bit ugly and so I hid it in a class ArtServiceHelper
  - Repository: https://github.com/dladams/art_extensions
    - Code is in AXService with example in test/services
    - See some of the code on following pages
  - Loads system services where required
  - So far, only supports a single thread

ArtServiceHelper.h (1/3)

```
...
class ArtServiceHelper {

public:

  typedef std::vector<std::string> NameList;
  typedef std::map<std::string, std::string> ConfigurationMap;

  // Return the one instance of this (singleton) class.
  static ArtServiceHelper& instance();

  // Close the one instance of this class.
  // Services are not longer available.
  // The current instance of this class and all services are deleted.
  // For TFileService, the files are written and renamed.
  // The load status is set to 3.
  static void close();

  // Dtor.
  ~ArtServiceHelper() = default;
```

Singleton

Call this to close services.
Needed to write objects and close and rename files for TFileService

ArtServiceHelper.h (2/3)

```
// Add a service.
//   name - Name of the service, e.g. "TFileService"
//   sval - if not isFile, configuration string for the service, e.g. for
//     TFileService:
//          service_type: "TFileService" fileName: "test.root"
//     Note this is just the contents, not the full named block.
//     if isFile, base file name. Path to locate file is $FHICL_FILE_PATH.
// Configuration format is the same as that found in the services block
// of an fcl file.
// Returns 0 for success.
int addService(std::string name, std::string sval ="", bool isFile =false);
```

The string sval can either be the name of an fcl file or the fcl parameters for the service

```
// Load the services, i.e. make them available for use via
// art::ServiceHandle.
// Returns the status: 1 for success, 2 for failure.
int loadServices();
```

Call this after all services are added.
They are now accessible with ServiceHandle

ArtServiceHelper.h (3/3)

```
  // Return the names of added services.
  NameList serviceNames() const;

  // Return the configuration string for a service.
  std::string serviceConfiguration(std::string name) const;

  // Return the full configuration string.
  std::string fullServiceConfiguration() const;

  // Return the service status.
  //   0 - not loaded
  //   1 - services loaded and available
  //   2 - service load failed
  //   3 - service helper is closed
  int serviceStatus() const;

  // Display the contents and status of a service helper.
  void print(std::ostream& out =std::cout) const;

private:
...
};
```

Methods to view the information about added services.

test_LArProperties.cxx (1/2)

```
...
  cout << myname << line << endl;
  cout << myname << "Fetch art service helper." << endl;
  ArtServiceHelper& ash = ArtServiceHelper::instance();

  cout << myname << line << endl;
  cout << myname << "Add the LArProperties service." << endl;
  scfg = "prodsingle_dune35t.fcl";
  cout << myname << "Configuration: " << scfg << endl;
  assert( ash.addService("LArProperties", scfg, true) == 0 );

  cout << myname << line << endl;
  cout << myname << "Add the DatabaseUtil service." << endl;
  scfg = "DBHostName: \"fnalpgsdev.fnal.gov\" DBName: \"dune_dev\" DBUser:
\"dune_reader\" PassFileName: \".lpswd\" Port: 5438 ShouldConnect: false
TableName: \"main_run\" ToughErrorTreatment: false";
  cout << myname << "Configuration: " << scfg << endl;
  assert( ash.addService("DatabaseUtil", scfg) == 0 );

  cout << myname << line << endl;
  cout << myname << "Print the services." << endl;
  ash.print();
```

Add service with fcl.

Add service with string.

test_LArProperties.cxx (2/2)

```
  cout << myname << line << endl;
  cout << myname << "Load the services." << endl;
  assert( ash.loadServices() == 1 );

  cout << myname << line << endl;
  cout << myname << "Get LArProperties service." << endl;
  art::ServiceHandle<util::LArProperties> plarsrv;

  cout << myname << line << endl;
  cout << myname << "Use LArProperties service." << endl;
  cout << myname << "    LAr Density: " << plarsrv->Density() << endl;
  cout << myname << "  DriftVelocity: " << plarsrv->DriftVelocity() << endl;

  cout << myname << line << endl;
  cout << myname << "Close services." << endl;
...
```

Load services.

Access service.

Use service.

Close services.

Testing status

The table lists the services that have been tested thus far

- The services required to use each are also listed
  - Those marked with asterisks (*) require system services that are are loaded automatically

Service	Required services	Status
TFileService	TriggerNamesService*	Tests OK
RandomNumberGenerator	CurrentModule*	Tests OK
Geometry	ExprGeoHelper	Tests OK
DetectorProperties	TimeService, DatabaseUtil	Tests OK
LArProperties	DatabaseUtil	Tests OK
TimeService	DatabaseUtil	Tests OK

Plans

I plan to continue testing services as interest dictates

- See art_extensions/test/services/test_*.cxx for the services tested
- Happy to add any additional services that are of interest

I would like to move ArtServiceHelper down our SW stack

- User → DUNE → LArSoft → art
- Modify as required for final destination while retaining functionality
- Ask this group to support move to DUNE, LArSoft or art
- I am happy to do this or relinquish control to others

Further development

- ~~Provide option to take service configurations from an fcl file~~ – Done
- Add support for multiple threads
  - Do art, LArSoft have this now?
- Provide option for users to provide module-like context for services

Split services and geometry

An alternative is "split services"

- I.e. designing service so that the non-art code is in one part and the art interface is provided in a separate piece
- The users outside the art framework can use only the first part and get the desired functionality
- LArSoft Geometry service is already this way—the non-art part is in the base class GeometryCore
- Examples
  - For use of geometry service with ArtServiceHelper, see art_extensions/test/services/test_Geometry.cxx
  - For direct use of GeometryCore, see dune_extensions/test/DXGeometry/test_geometry.cxx and dune_extensions/test/DXGeometry/draw_geometry.cxx
    - The latter is a simple event display
- Splitting services would require significant effort for existing services
  - There would be the temptation to not bother with this and leave users where they are now for many services
  - Probably good to keep ArtServiceHelper for those cases where art, LArSoft or user does not make the effort to split a service

Conclusions

It *is* possible to use existing art services outside art framework

- Demonstrated in standalone programs
- Same presumably true for Root scripts
- No changes made to art or LArSoft code

ArtServiceHelper

- Hides the code required to load the services
- Provides a simple, intuitive interface to configure services
- Service access is via the art ServiceHandle (as in art modules)
- Like to move this or equivalent to DUNE, LArSoft or art
- Ideas presented for future development

Other approaches

- Split services is one but would require effort for every shared service

Extra: fcldump

During the above work, I wanted a nice dump of fcl contents

One can use "`export ART_DEBUG_CONFIG=1`"

- But output is not very pretty or compact
- Each variable is on a separate line with the full path to the file where that variable is last set
  - Useful for debugging
- Also this show the full configuration including the fields added by art
  - Not just the fields in the fcl files

I created `fcldump` which has a nicer (IMHO) format

- Unlike the previous, this does not include
  - the source information for each variable or
  - the fields added by art
- Included in art_extensions
- Build that package and use "`fcldump -h`" for help
- I would also like to push this up the DUNE/LarSoft/art chain