# art - Feature #11156

## Suggestions to update the fhicl documentation

12/15/2015 06:33 PM - Rob Kutschke

| | | | |
|---|---|---|---|
| **Status:** | Accepted | **Start date:** | 12/15/2015 |
| **Priority:** | Low | **Due date:** | |
| **Assignee:** | | **% Done:** | 0% |
| **Category:** | | **Estimated time:** | 0.00 hour |
| **Target version:** | | **Spent time:** | 0.00 hour |
| **Scope:** | Internal | **SSI Package:** | |
| **Experiment:** | Mu2e | | |

**Description**

I just read the latest fhicl document: https://cdcvs.fnal.gov/redmine/attachments/download/29136/quick_start_v3.pdf

It's really good. I do have a few comments.

1. Page 7. A common rookie mistake is misuse of member notation. Can you find a way to make two items more visible so that they are not missed by the casual reader:
    1. Member notation must start from outer-most scope # The point that you make on page 11 illustrated below:

```
foo : {
   t1 :  1
   t2 : @local::foo.t1  // error because initial defintiion of foo is not yet complete
}
```

2. Page 13. I believe that arbitrary horizontal white space is permitted between the #include and the filename. The statement that the only allowed format is exactly one space was true in the initial c++ binding but was subsequently relaxed.
3. Page 13. Expand on FHCIL_FILE_PATH.

```
export FHICL_FILE_PATH=a:b
#include "foo.fcl"
```

   will look for a/foo.fcl and b/foo.fcl and will accept the first match. It does NOT look for: a/*/foo.fcl or b/*/foo.fcl, where * means an arbitrarily deep path fragment.
   Can you also discuss when there is a prohibition on absolute paths and give a use case to motivate that restriction.
4. Page 14. What does the following do?

```
foo : [ 0, 1, 2 ]
@erase::foo[1]
```

5. Page 14. Provide use cases for protect_ignore and protect_error. I can describe the Mu2e use case if you wish.
6. The document would benefit from showing a "best practices" example that makes use of many of the high end features. We might be able to build an example using the toyExperiment package. Maybe this does not belong in the document but elsewhere, perhaps in the art workbook? If so this document should link to it. My own notion of best practice is to use nested definitions in order to provide single points of maintenance: when an expert tweaks the default parameter set for their module, everyone else should get that automatically withouth having to edit their own fcl files.

### History

**#1 - 12/16/2015 08:01 AM - Kyle Knoepfel**

Thanks, Rob, for the suggestions. Below, I include some comments/questions I got from Gianluca a while ago:

I think this will be invaluable documentation. Provided that I can convince people to read it, that is.
I have a few questions...

4.2) it might be worth anticipating that this simple behaviour can be
influenced by some operators at 10.2.

5.2.1) I would stress that case sensitivity is still the rule for "true" and

"false".

5.2.2) what is it expected to happen assigning "infinity" to types that don't support it (e.g., C++ short)? undefined? unsupported?

5.2.4) are string not separated by spaces automatically concatenated? "What "' about '" this "" perversion?"

8) can values in the prologs be modified, rather than overridden, inside and or outside the prolog? In the example: "a.b: 12"

9) is a comment allowed at the end of an #include line? I believe I had problems with it in the past...

10.1) @erase does not apply to list elements, right? "$a^3$: @erase"

10.2.2) I expected a1 to completely disappear, and that the showed result would be generated by having "a1.b: @erase"

10.2.3) the choice of "a.b.c: 37" might be confusing, since you are trying to assign the same value as already in. But my question is: what happens with "d.b.c: 43"? does @local::, @table and friends preserve the binding operator? (I believe they do...)

#### #2 - 12/21/2015 11:21 AM - Kyle Knoepfel

*- Status changed from New to Accepted*