art - Feature #2334

Another candidate for cetlib?

12/20/2011 11:08 AM - Rob Kutschke

Status:	Rejected	**Start date:**	12/20/2011
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:	Application	**Estimated time:**	0.00 hour
Target version:		**Spent time:**	0.00 hour
Scope:	Internal	**SSI Package:**	cetlib
Experiment:	Mu2e		

Description

I have used the following template a number of times recently. I use it as a handy notation to print the contents of a vector (often in a debugging context):

The template allows me to:

```
  std::vector<T> v; // where T has operator<<()
       // fill v.
  std:: cout << v << endl;
```

The fragment is:

```
 template <class T>
  inline std::ostream& operator<<(std::ostream& ost,
                                  typename std::vector<T> const& v ){
    for ( typename std::vector<T>::const_iterator i=v.begin(), e=v.end();
          i != e; ++i ){
      ost << *i;
    }
    return ost;
  }
```

Is this generally a safe practice?
If so, is it a candidate for inclusion in cetlib in header named, perhaps, cet/VectorPrinter.h ?

History

#1 - 12/20/2011 01:15 PM - Walter E Brown

- Category set to Application

- Status changed from New to Feedback

- Assignee set to Walter E Brown

The usual one-liner for something like this avoids any need for an explicit streaming operator:

```
#include <algorithm>
#include <iterator>

std::copy( v.begin(), v.end(), std::output_iterator<T>(std::cout, " ") );
```

Please give this a try and see if you still feed the need for the template you describe.

#2 - 12/20/2011 03:36 PM - Rob Kutschke

Thanks - I will add that to my tool kit. But it does not address the particular issue I have been working on these recent days.

In the use case I was doing today, I had a small message reporting class template

```
template &lt;class T&gt;
  void mark ( ... many arguments
              T const& t) {
```

```
cout << " " // lots of printout from arguments
           <<  t
           << endl;

}
```

where T at various times has been a built in type, a user written class or a std::vector of either.

#3 - 01/09/2012 09:24 AM - Walter E Brown

As pointed out during the stakeholders' meeting on 2012-01-06, the proposed operator << function would, when called, not be found via either ordinary name lookup or argument-dependent name lookup without some kind of using.

While we can likely design something (perhaps along the lines of MessageFacility's user syntax) for the requested purpose, let's first consider whether and to what degree the MessageFacility library itself might serve.

#4 - 08/02/2013 02:16 PM - Christopher Green

- Description updated

- Status changed from Feedback to Rejected

- Assignee deleted (Walter E Brown)

- Scope set to Internal

- Experiment Mu2e added

- SSI Package cetlib added

Upon consideration, we do not believe this is sufficiently safe to make generally available due to argument-dependent lookup issues. Happily, C++2011 allows for several one liners to provide the same (actually more) functionality:

```
for (auto const & x : v) std::cout << x << ", ";
```

```
std::copy(v.cbegin(), v.cend(), std::ostream_iterator(std::cout, ", "));
```

or

```
cet::copy_all(v, std::ostream_iterator(std::cout, ", "));
```