abstract: Header `<boost/iterator/iterator_traits.hpp>` provides the ability to access an iterator’s associated types using MPL-compatible metafunctions.

Overview

`std::iterator_traits` provides access to five associated types of any iterator: its `value_type`, `reference`, `pointer`, `iterator_category`, and `difference_type`. Unfortunately, such a “multi-valued” traits template can be difficult to use in a metaprogramming context. `<boost/iterator/iterator_traits.hpp>` provides access to these types using a standard metafunctions.

Summary

Header `<boost/iterator/iterator_traits.hpp>`:

```cpp
template <class Iterator>
struct iterator_value
{
    typedef typename
        std::iterator_traits<Iterator>::value_type
        type;
};

template <class Iterator>
struct iterator_reference
{
    typedef typename
        std::iterator_traits<Iterator>::reference
        type;
};

template <class Iterator>
struct iterator_pointer
{
    typedef typename
        std::iterator_traits<Iterator>::pointer
        type;
};
```
template <class Iterator>
struct iterator_difference
{
    typedef typename
detail::iterator_traits<Iterator>::difference_type
type;
};

template <class Iterator>
struct iterator_category
{
    typedef typename
detail::iterator_traits<Iterator>::iterator_category
type;
};

---

**Broken Compiler Notes**

Because of workarounds in Boost, you may find that these metafunctions actually work better than the facilities provided by your compiler's standard library.

On compilers that don’t support partial specialization, such as Microsoft Visual C++ 6.0 or 7.0, you may need to manually invoke `BOOST_BROKEN_COMPILER_TYPE_TRAITS_SPECIALIZATION` on the `value_type` of pointers that are passed to these metafunctions.

Because of bugs in the implementation of GCC-2.9x, the name of `iterator_category` is changed to `iterator_category_` on that compiler. A macro, `BOOST_ITERATOR_CATEGORY`, that expands to either `iterator_category` or `iterator_category_`, as appropriate to the platform, is provided for portability.