C++11 Mapping for Operations on Local Types

Previous Next

An operation on a local interface or a local class is mapped to a pure virtual member function with the same name. The mapping of operation parameters to C++ is identical to the Client-Side Mapping for these parameters:

- in parameters are passed by value (for simple types such as integers), or by const reference (for other types)
- out parameters are passed by reference
- return values are mapped to return values in C++

However, unlike the Client-Side mapping, there is no trailing `Ice::Context` parameter in the mapped member functions.

The type of a parameter can be a local interface or class. Such a parameter is passed as a `std::shared_ptr<`*`mapped C++ class`*`>` for regular parameters (const reference for in parameters, reference for out parameters); a "delegate" interface parameter, mapped to a `std::function` in C++, is passed by value if it's an in parameter, and by reference if it's an out parameter.

A `LocalObject` parameter is mapped to a parameter of type `std::shared_ptr<void>`. You can also create constructed types (such as local sequences and local dictionaries) with local types.

For example:

Slice

```
module M
{
    local interface L; // forward declared
    local sequence<L> LSeq;
    local interface L
    {
        string op(int n, string s, LocalObject any, out int m, out string t, out LSeq newLSeq);
    }
}
```

is mapped to:

C++

```
namespace M
{
    class L;
    using LSeq = std::vector<std::shared_ptr<L>>;
    class L
    {
    public:
        virtual ~L() = default;
        virtual std::string op(int n, const std::string&, const std::shared_ptr<void>& any,
                               int& m, std::string& t, LSeq& newLSeq) = 0;
    };
}
```

Back to Top ^

Previous Next