Command-Line Parsing and Initialization

Previous Next

On this page:

Parsing Command Line Options

When you initialize the Ice run time by calling `initialize`, you can pass the application's arguments to the initialization call.

In most language mappings, this argument vector is an *in-out* parameter. In C++, for example, `argc` is passed as a *reference* to an `int`:

C++11

```
std::shared_ptr<Ice::Communicator> initialize(int& argc, const char* argv[], ...other parameters...);
```

C++98

```
Ice::CommunicatorPtr initialize(int& argc, const char* argv[], ...other parameters...);
```

`initialize` parses the argument vector and initializes the new communicator's properties accordingly. It also removes all arguments that set Ice properties from the provided argument vector. For example, assume we invoke a C++ server as:

```
./server --myoption --Ice.Config=config -x a --Ice.Trace.Network=3 -y opt file
```

Initially, `argc` has the value `9`, and `argv` has ten elements: the first nine elements contain the program name and the arguments, and the final element, `argv[argc]`, contains a null pointer (as required by the C++ standard). When `Ice::initialize` returns, `argc` has the value `7` and `argv` contains the following elements:

```
./server
--myoption
-x
a
-y
opt
file
0             # Terminating null pointer
```

This means that you should initialize the Ice run time before you parse the command line for your application-specific arguments. That way, the Ice-related options are stripped from the argument vector for you so you do not need to explicitly skip them.

`initialize` provides the same argument-property parsing and stripping in all language mappings.

If you use the Application helper class, the `run` member function or method is passed an argument vector with the Ice-related options already stripped. The same is true for the `runWithSession` member function or method called by the `Glacier2::Application` helper class.

Back to Top ^

The `Ice.ProgramName` Property

For C++, Objective-C, Python, and Ruby, `initialize` sets the `Ice.ProgramName` property to the name of the current program (`argv[0]`). In C#, `initialize` sets `Ice.ProgramName` to the value of `System.AppDomain.CurrentDomain.FriendlyName`. Your application code can read this property and use it for activities such as logging diagnostic or trace messages.

Even though `Ice.ProgramName` is initialized for you, you can still override its value from a configuration file or by setting the property on the command line.

For Java, the program name is not supplied as part of the argument vector — if you want to use the `Ice.ProgramName` property in your application, you must set it before initializing a communicator.

Back to Top ^

See Also

- Using Configuration Files
- Reading Properties
- Communicator Initialization
- Glacier2 Application Class

Previous

Next