Initialization in Ruby

Previous Next

Every Ice-based application needs to initialize the Ice run time, and this initialization returns an `Ice::Communicator` object.

A `Communicator` is a local Ruby object that represents an instance of the Ice run time. Most Ice-based applications create and use a single `Communicator` object, although it is possible and occasionally desirable to have multiple `Communicator` objects in the same application.

You initialize the Ice run time by calling `Ice::initialize`, for example:

Ruby

```
require 'Ice'

communicator = Ice::initialize(ARGV)
```

`Ice::initialize` accepts the argument list that is passed to the program by the operating system. The function scans the argument list for any command-line options that are relevant to the Ice run time; any such options are removed from the argument list so, when `Ice::initialize` returns, the only options and arguments remaining are those that concern your application. If anything goes wrong during initialization, `initialize` throws an exception.

Before leaving your program, you must call `Communicator.destroy`. The `destroy` method is responsible for finalizing the Ice run time. In particular, `destroy` ensures that any outstanding threads are joined with and reclaims a number of operating system resources, such as file descriptors and memory. Never allow your program to terminate without calling `destroy` first.

The general shape of our Ice Ruby application is therefore:

Ruby

```
require 'Ice'
begin
    communicator = Ice::initialize(ARGV)
    ...
ensure
    if defined? communicator and communicator != nil
        communicator.destroy()
    end
end
```

Back to Top ^

See Also

- Communicators
- Communicator Initialization
- Application Helper Class

Previous Next