

# Client-Side Slice-to-Ruby Mapping

The client-side Slice-to-Ruby mapping defines how Slice data types are translated to Ruby types, and how clients invoke operations, pass parameters, and handle errors. Much of the Ruby mapping is intuitive. For example, Slice sequences map to Ruby arrays, so there is essentially nothing new you have to learn in order to use Slice sequences in Ruby.

The Ruby API to the Ice run time is fully thread-safe. Obviously, you must still synchronize access to data from different threads. For example, if you have two threads sharing a sequence, you cannot safely have one thread insert into the sequence while another thread is iterating over the sequence. However, you only need to concern yourself with concurrent access to your own data — the Ice run time itself is fully thread safe, and none of the Ice API calls require you to acquire or release a lock before you safely can make the call.

Much of what appears in this chapter is reference material. We suggest that you skim the material on the initial reading and refer back to specific sections as needed. However, we recommend that you read at least the mappings for [exceptions](#), [interfaces](#), and [operations](#) in detail because these sections cover how to call operations from a client, pass parameters, and handle exceptions.



In order to use the Ruby mapping, you should need no more than the Slice definition of your application and knowledge of the Ruby mapping rules. In particular, looking through the generated code in order to discern how to use the Ruby mapping is likely to be inefficient, due to the amount of detail. Of course, occasionally, you may want to refer to the generated code to confirm a detail of the mapping, but we recommend that you otherwise use the material presented here to see how to write your client-side code.



## The Ice Module

All of the APIs for the Ice run time are nested in the `Ice` module, to avoid clashes with definitions for other libraries or applications. Some of the contents of the `Ice` module are generated from Slice definitions; other parts of the `Ice` module provide special-purpose definitions that do not have a corresponding Slice definition. We will incrementally cover the contents of the `Ice` module throughout the remainder of the manual.

A Ruby application can load the Ice run time using the `require` statement:

```
require 'Ice'
```

If the statement executes without error, the Ice run time is loaded and available for use. You can determine the version of the Ice run time you have just loaded by calling the `stringVersion` function:

```
icever = Ice::stringVersion()
```

## Topics

- [Ruby Mapping for Identifiers](#)
- [Ruby Mapping for Modules](#)
- [Ruby Mapping for Built-In Types](#)
- [Ruby Mapping for Enumerations](#)
- [Ruby Mapping for Structures](#)
- [Ruby Mapping for Sequences](#)
- [Ruby Mapping for Dictionaries](#)
- [Ruby Mapping for Constants](#)
- [Ruby Mapping for Exceptions](#)
- [Ruby Mapping for Interfaces](#)
- [Ruby Mapping for Operations](#)
- [Ruby Mapping for Classes](#)
- [Code Generation in Ruby](#)
- [The main Program in Ruby](#)
- [Using Slice Checksums in Ruby](#)
- [Example of a File System Client in Ruby](#)