Client-Side Slice-to-Objective-C Mapping

Previous Next

The client-side Slice-to-Objective-C mapping defines how Slice data types are translated to Objective-C types, and how clients invoke operations, pass parameters, and handle errors. Much of the Objective-C mapping is intuitive. For example, Slice dictionaries map to Cocoa framework dictionaries, so there is little new you have to learn in order to use Slice dictionaries in Objective-C.

The Objective-C mapping is thread-safe. For example, you can concurrently invoke operations on an object from different threads without the risk of race conditions or corrupting data structures in the Ice run time, but you must still synchronize access to application 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 Objective-C mapping, you should need no more than the Slice definition of your application and knowledge of the Objective-C mapping rules. In particular, looking through the generated code in order to discern how to use the Objective-C mapping is likely to be confusing because the generated code is not necessarily meant for human consumption and, occasionally, contains various cryptic constructs to deal with mapping internals. 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.

ICE, ICEMX, GLACIER2, ICESTORM, ICEGRID Prefixes

All of the APIs for the Ice run time are prefixed by `ICE`, to avoid clashes with definitions for other libraries or applications. Parts of the Ice API are generated from Slice definitions; other parts provide special-purpose definitions that do not have a corresponding Slice definition. Regardless of the way they are defined, the `ICE` prefix universally applies to all entry points in the Ice run time. We will incrementally cover the contents of the Ice API throughout the remainder of the manual.

The APIs for IceMX, Glacier2, IceStorm and IceGrid are also prefixed with respectively ICEMX, GLACIER2, ICESTORM and ICEGRID.

Topics

Previous Next