

Client-Side Slice-to-C-Sharp Mapping

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

The C# 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 C# mapping, you should need no more than the Slice definition of your application and knowledge of the C# mapping rules. In particular, looking through the generated code in order to discern how to use the C# 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 Namespace

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

Topics

- [C-Sharp Mapping for Identifiers](#)
- [C-Sharp Mapping for Modules](#)
- [C-Sharp Mapping for Built-In Types](#)
- [C-Sharp Mapping for Enumerations](#)
- [C-Sharp Mapping for Structures](#)
- [C-Sharp Mapping for Sequences](#)
- [C-Sharp Mapping for Dictionaries](#)
- [C-Sharp Collection Comparison](#)
- [C-Sharp Mapping for Constants](#)
- [C-Sharp Mapping for Exceptions](#)
- [C-Sharp Mapping for Interfaces](#)
- [C-Sharp Mapping for Operations](#)
- [C-Sharp Mapping for Classes](#)
- [C-Sharp Mapping for Optional Values](#)
- [Serializable Objects in C-Sharp](#)
- [C-Sharp Specific Metadata Directives](#)
- [Asynchronous Method Invocation \(AMI\) in C-Sharp](#)
- [slice2cs Command-Line Options](#)
- [Using Slice Checksums in C-Sharp](#)
- [Example of a File System Client in C-Sharp](#)