Type IDs

Each user-defined Slice type has an internal type identifier, known as its *type ID*. The type ID is simply the fully-qualified name of each type. For example, the type ID of the `Child` interface in the preceding example is `::Family::Children::Child`. All type IDs for user-defined types start with a leading `::`, so the type ID of the `Family` module is `::Family` (not `Family`). In general, a type ID is formed by starting with the global scope (`::`) and forming the fully-qualified name of a type by appending each module name in which the type is nested, and ending with the name of the type itself; the components of the type ID are separated by `::`.

The type ID of a proxy is formed by appending a `*` to the type ID of an interface or class. For example, the type ID of a `Child` proxy is `::Family::Children::Child*`.

The type ID of the Slice `Object` type is `::Ice::Object` and the type ID of an `Object` proxy is `::Ice::Object*`.

The type IDs for the remaining built-in types, such as `int`, `bool`, and so on, are the same as the corresponding keyword. For example, the type ID of `int` is `int`, and the type ID of `string` is `string`.

Type IDs are used internally by the Ice run time as a unique identifier for each type. For example, when an exception is raised, the marshaled form of the exception that is returned to the client is preceded by its type ID on the wire. The client-side run time first reads the type ID and, based on that, unmarshals the remainder of the data as appropriate for the type of the exception.

Type IDs are also used by the `ice_isA` operation.

See Also

- ice_isA