ECMA: Technical Report TR/89 - Common Language Infrastructure (CLI) - Common Generics

return to the Ecma Standards

Table 1: Data sheet for Common Language Infrastructure (CLI) - Common Generics

Title	Common Language Infrastructure (CLI) - Common Generics
Acronym	
Version	2
Series	TR
Document Number	TR/89
Release Date	June 2006
About Specification	
Download	https://www.ecma-international.org/publications/files/ECMA-TR/ECMA%20TR-089.pdf

Note: The following is an excerpt from the official ECMA description. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Overview

The CLI standard libraries (ISO/IEC 23271) provide a collection of common types that can be used by multiple languages. With the addition of generics to the CLI, the standard libraries have been extended to include a number of common generic types, in particular, collections. However, at present, these libraries do not include many simple generic types found in a number of different languages. Any language which uses these common types must implement them rather than deferring to the CLI library, thereby reducing language interoperability. This proposal addresses this issue by providing a number of these common types.

Generic tuples (product types) are standard in a number of languages: C++ (`template Pair`), Ada, Haskell, and Standard ML (SML). However, languages differ in the number of predefined tuple sizes supported by their standard libraries; e.g., C++ provides just one (Pair) while Haskell provides eight (sizes 2 to 9) and SML allows any size of tuple. This proposal provides nine (sizes 2 to 10).

Generic programming encourages "higher order" programming where generic functions (methods) take function (delegate) type arguments that have generic types. Examples include Ada's with and generic constraints, and function arguments in Haskell and SML. In the CLI, function values are provided in the form of delegates, so this proposal defines standard generic delegate types for functions (which return a value) and procedures (which do not).

Another two types that occur in a number of languages are an optional type, which either contains a value of some other type or an indication that such a value is not present; and an either type, which holds a value of one of two possible types and an indication of which one is present. This

proposal provides both of these. (Note: The optional type is similar to, but different from, the type `System.Nullable`.)

Finally, in existing generic languages, a need has been found for a filler type to be used when a particular generic parameter is not required for a particular use of the generic type. A standard one-value type is often provided for this purpose, often called `Unit` or `Void` . This proposal includes such a type.