Differences between the .NET interface and the .NET Framework interface:

- .NET Framework only works on Microsoft Windows; .NET is cross-platform.

- The .NET Framework interface works on both Unicode and Classic editions of Dyalog; the .NET interface only works on Unicode editions of Dyalog.

- On Microsoft Windows it is not currently possible to use both the .NET interface and the .NET Framework interface at the same time. This means that, on Windows, the .NET interface has to be explicitly enabled. This is done by setting DYALOG_NETCORE=1 (the default is 0 for backwards compatibility). On Linux and macOS, DYALOG_NETCORE defaults to 1, so needs to be explicitly set to 0 in the (unlikely) event that you want to disable the .NET interface.

- The .NET interface automatically converts Tuples (https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/builtin-types/value-tuples) to and from nested arrays.

- The .NET interface automatically "associates" extension methods (https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/extension-methods) with relevant classes.
    - When the .NET interface loads an assembly, it identifies all relevant extension methods and add them to its internal cache of members. These extension methods will then appear as any other method in the relevant objects.

- The .NET interface handles `]USING` differently to the way the .NET Framework interface does. If the assembly specification in a `]USING` statement does not end in "**.dll**" then the assembly is loaded by name (not by filename). This means that .NET can use its built-in binding mechanisms to determine from where to load the assembly. This is useful as in many cases it removes the need to know where an assembly is located. [implementation-wise we call Assembly.Load() not Assembly.LoadFrom()]

NOTE: Features that are in the .NET Framework Interface Guide but are not included in the .NET Interface Guide should not be assumed to work in .NET.