ECMA/TC39/00/9

Common Language Infrastructure (CLI)

Introduction and Class Library Factorization

Hewlett-Packard

Intel

Microsoft

Outline

- What is the CLI?
- Factoring the Base Class Libraries
- Categories
- Packages by Category
- Questions and Answers

Overview of the CLI

- A file format
- A common type system
- An extensible metadata system
- An intermediate language
- Access to the underlying platform
- A factored base class library

File Format

- Based on COFF
- Uses existing extension mechanism
- Code represented as MSIL instructions
- Metadata stored in read-only area
- EAT / IAT for access to platform only
- Methods include a descriptive header
  - Stack frame size
  - Types of local variables and parameters
  - Pinned variable information
  - Exception handler table

Common Type System

- Spans large number of languages
- Object-oriented in flavor
- Supports procedural and functional languages, too
- Includes value types (“structs”), pointers, and by-reference values
- Subset for wide reach
  - Common Language Specification (CLS)

Metadata System

- Self-description for assemblies (components)
  - Includes referenced assemblies
  - Allows crypto-strong names
  - Records version information
  - Security boundary
- Self-description for types
  - Name and defining assembly
  - Member information (fields, methods, etc.)
- Extensible through custom attributes
- Stored in file along with code

Intermediate Language

- Simple stack machine model
- Typeless opcodes (`add`, not `add.int32`)
  - Signed and unsigned via opcode, not type
  - Rich set of conversion operations
- Verifiable subset
- Tail calls, virtual dispatch, call via function pointer, exception handling (two-pass)
- Typed variable argument lists, dynamically typed pointers
- Objects, vectors, and strings are built-in
  - As are 32- and 64-bit integers and floats, and 32/64-bit agnostic integers

Access to Platform

- Metadata describes managed and unmanaged interface
- Marshaling is automatic for many types
- Custom marshaling can be specified
- Platform-specific transformations are possible (ANSI <-> Unicode, etc.)
- Platform-specific calling conventions can be specified

Factored Class Library

- Designed for cross-language use
  - Adheres to the CLS rules
- Factored to allow minimal footprint and minimal hardware requirements
- Intended to be platform-neutral
- Three layers: kernel, basic language, additional functionality
- Methodology and details follow….

Outline

- What is the CLI?
- Factoring the Base Class Libraries
- Categories
- Packages by Category
- Questions and Answers

Goals

- Factored Class Library
  - Size constraints (RAM, ROM, Flash)
  - Computational constraints (FPU, 64bit support)
  - Feature requirements
- Factored Execution Environment
  - Minimal base is always present
  - File format independent of factorization
  - Library factorization is the driver
- Standardization allows …
  - **… vendors** to specify what's available
  - **… developers** to specify requirements

Methodology

- Define ***Kernel***
  - Fixes file format
  - Minimal functionality and hardware
  - Hand-picked classes and methods
- Define ***Basic Language***
  - Minimal hardware support required
  - Most common language features
    - Features required for C# with minimal hardware support
  - Depends on classes defined in ***Kernel***
- Package each advanced function separately
  - Implemented a la cart by runtime vendors
  - Required a la cart by developers

Defining a Package

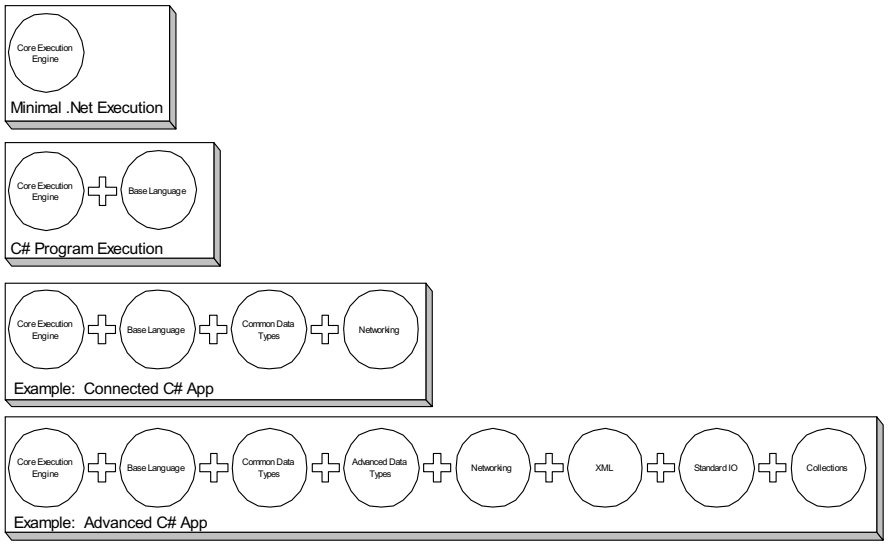
- Choose the classes
  - A class can only be in one package
  - Minimize and specify dependencies on packages
  - Base class in package or one it depends on
- ***Basic Language*** depends on the ***Kernel*** package
- All other packages depend on both ***Kernel*** and ***Basic Language***
- Compute the missing methods
  - Check it makes sense, new dependencies
  - Interfaces may be in another package
    - Methods will exist, just can't cast to interface

Languages and Packages

- C#
  - Requires ***Kernel***, ***Basic Language***, and ***Extended Numerics***
- ECMAScript
  - Requires above plus ***Reflection***
- ISO C++
  - Requires ***Kernel***, ***Basic Language***, ***Extended Numerics***, and ***NonCLS***

Scenario-based System Design

Scenario	***Required Packages***
Minimal	Kernel
C# Program	Kernel, Basic Language
Ex: Connected C# Application	Kernel, Basic Language, Common DT, Networking
Ex: Connected XML C# Application	Kernel, Basic Language, Common DT, Advanced DT, Networking, XML, IO, Collections



Outline

- What is the CLI?
- Factoring the Base Class Libraries
- Categories
- Packages by Category
- Questions and Answers

Categories of Packages

- Classes grouped into packages
- Packages grouped into five categories
  - For ease of discussion only

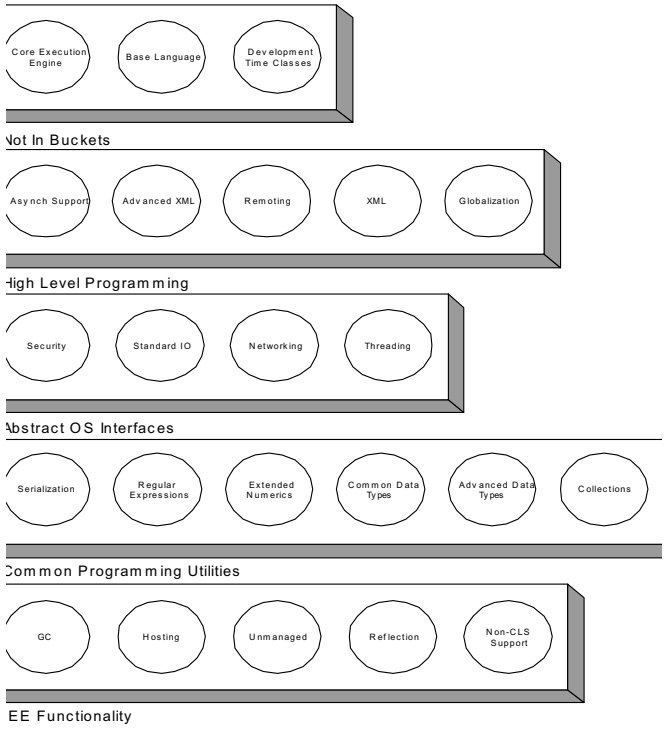
Miscellaneous

High Level Programming

Abstract OS Interfaces

Common Programming Utilities

EE Functionality



The Five Categories (1 – 3)

- Abstract OS Interface
  - Platform-independent operating system functionality
- Common Programming Library
  - Classes that support common programming patterns
- High-Level Programming
  - Programming patterns for the 2000s: XML, remote objects, asynchronous computing

The Five Categories (4 – 5)

- EE Functionality
  - Revealing underlying operations to programming languages
- Miscellaneous
  - ***Kernel***, ***Basic Language***, and support for developers

Outline

- What is the CLI?
- Factoring the Base Class Libraries
- Categories
- Packages by Category
- Questions and Answers

Abstract OS Interface

183 Classes and interfaces

- Networking (60)
  - System.Net.*
- Security (60)
  - System.IsolatedStorage, System.Security, …
- Standard I/O (32)
  - System.Console, System.IO, System.Text, …
- Threading (31)
  - System.Threading, …

Common Programming Lib.

118 Classes and interfaces

- Common Data Types (5)
  - System.DateTime, System.Text.StringBuilder, etc.
- Advanced Data Types (11)
  - System.BitConverter, System.URI, …
- Collections (27)
  - System.Collections
- Extended Numerics (6)
  - System.Decimal, System.Double, etc.
- Regular Expressions (8)
  - System.Text.RegularExpressions.*
- Serialization (61)
  - System.Runtime.Serialization.*, etc.

High-Level Programming

188 Classes and interfaces

- Asynchronous Programming (2)
  - System.AsyncCallback, System.IAsyncResult
- Globalization (39)
  - System.Globalization.*, System.Resources.*, etc.
- Remoting (88)
  - System.Runtime.Remoting.*
- XML (54)
  - System.Xml.* (parsing and generation)
- Advanced XML (5)
  - System.Xml.Xsl.*, System.Xml.XPath.*

EE Functionality

96 Classes and interfaces

- GC (2)
  - System.WeakReference, System.WeakReferenceException
- Hosting (3)
  - System.OperatingSystem, etc.
- NonCLS (3)
  - System.ArgIterator, etc.
- Reflection (62)
  - System.Reflection.*, etc.
- Unmanaged (26)
  - System.Runtime.InteropServices, etc.

Miscellaneous

107 Classes and interfaces

- Kernel (66)
  - 1, 2, and 4 byte integers, arrays, string, object, etc.
- Basic Language Support (17)
  - System.EventHandler, System.IFormattable, System.Type, etc.
- Development Time (24)
  - System.Diagnostics.*, System.Runtime.CompilerServices.*

Outline

- What is the CLI?
- Factoring the Base Class Libraries
- Categories
- Packages by Category
- Questions and Answers