



Microsoft .NET Framework 3.5, Application Development Foundation Exam 70-536

About this Exam

This exam is designed to measure your knowledge of .NET development fundamentals and is not tied to a particular version of .NET. Since the exam is now applicable to both Microsoft .NET Framework 2.0 and Microsoft .NET Framework 3.5 certification tracks, we have changed the name of the exam. Formerly TS: Microsoft .NET Framework 2.0 – Application Development Foundation, Exam 70-536 is now called TS: Microsoft .NET Framework, Application Development Foundation. If you are a .NET 2.0 developer, you do not need to learn .NET 3.5 to pass Exam 70-536; conversely, if you are a .NET 3.5 developer, you do not need to review .NET 2.0 to pass the exam



Audience Profile

Candidates for this exam work on a team in a medium-sized or large development environment that uses Microsoft Visual Studio .NET 2003 Enterprise Developer, Microsoft Visual Studio 2005, or Microsoft Visual Studio 2008. Candidates should have at least two to three years of experience developing Web-based, Windows-based, or distributed applications by using the Microsoft .NET Framework 1.0, the .NET Framework 1.1, the .NET Framework 2.0, or the .NET Framework 3.5. Candidates should have a working knowledge of Microsoft Visual Studio 2005 or Visual Studio 2008

How long is this course?

2KO International presents this course as a [five day full time course](#). We occasionally run this course as a part time course over 5 Saturdays (based on demand)

Where does the course take place?

At our training centre in Greenpoint, Cape Town, South Africa.
Or we can run it in-house at your business, anywhere in the world (requires individualised quotation).



We aim to offer training at the best prices – if you get a cheaper quote than us, send it to us (official quotes only) and we will beat it. Simple!



Skills Being Measured

This exam measures your ability to accomplish the technical tasks listed below. The percentages indicate the relative weight of each major topic area on the exam.

Developing applications that use system types and collections (15 percent)

- Manage data in a .NET Framework application by using .NET Framework system types. May include but is not limited to: Value types; Nullable type; Reference types; Attributes; Generic types; Exception classes; Boxing and UnBoxing ; TypeForwardedToAttribute class
- Manage a group of associated data in a .NET Framework application by using collections
May include but is not limited to: ArrayList class; Collection interfaces; Iterators; Hashtable class; CollectionBase class and ReadOnlyCollectionBase class; DictionaryBase class and DictionaryEntry class; Comparer class; Queue class; SortedList class; BitArray class; Stack class
- Improve type safety and application performance in a .NET Framework application by using generic collections
May include but is not limited to: Collection.Generic interfaces; Generic Dictionary; Generic Comparer class and Generic EqualityComparer class; Generic KeyValuePair structure; Generic List class, Generic List.Enumerator structure, and Generic SortedList class; Generic Queue class and Generic Queue.Enumerator structure; Generic SortedDictionary class; Generic LinkedList; Generic Stack class and Generic Stack.Enumerator structure
- Manage data in a .NET Framework application by using specialized collections
May include but is not limited to: Specialized String classes; Specialized Dictionary; Named collections; CollectionsUtil; BitVector32 structure and BitVector32.Section structure
- Implement .NET Framework interfaces to cause components to comply with standard contracts.
May include but is not limited to: IComparable interface; IDisposable interface; IConvertible interface; ICloneable interface; IEquatable interface; IFormattable interface
- Control interactions between .NET Framework application components by using events and delegates.
May include but is not limited to: Delegate class; EventArgs class; EventHandler delegates

Implementing service processes, threading, and application domains in a .NET Framework application (11 percent)

- Implement, install, and control a service.
May include but is not limited to: Inherit from ServiceBase class; ServiceController class and ServiceControllerPermission class; ServiceInstaller and ServiceProcessInstaller class; SessionChangeDescription structure and SessionChangeReason enumeration
- Develop multithreaded .NET applications.
May include but is not limited to: Thread class; ThreadPool class; ThreadStart delegate, ParameterizedThreadStart delegate, and SynchronizationContext class; Timeout class, Timer class, TimerCallback delegate, WaitCallback delegate, WaitHandle class, and WaitOrTimerCallback delegate; ThreadExceptionEventArgs class and ThreadExceptionHandler class; ThreadState enumeration and ThreadPriority enumeration; ReaderWriterLock



class; AutoResetEvent class and ManualResetEvent class; IAsyncResult interface and ICancelableAsyncResult interface (refer System Namespace); EventWaitHandle class, RegisterWaitHandle class, SendOrPostCallback delegate and IOCompletionCallback delegate; Interlocked class, NativeOverlapped structure and Overlapped class; ExecutionContext class, HostExecutionContext class, HostExecutionContextManager class, and ContextCallback delegate; LockCookie structure, Monitor class, Mutex class, and Semaphore class

- Create a unit of isolation for common language runtime within a .NET Framework application by using application domains.
May include but is not limited to: Create an application domain; Unload an application domain; Configure an application domain; Retrieve setup information from an application domain; Load assemblies into an application domain

Embedding configuration, diagnostic, management, and installation features into a .NET Framework application (14 percent)

- Embed configuration management functionality into a .NET Framework application.
May include but is not limited to: Configuration class and ConfigurationManager class; ConfigurationSettings class, ConfigurationElement class, ConfigurationElementCollection class and ConfigurationElementProperty class; Implement IConfigurationSectionHandler interface; ConfigurationSection class, ConfigurationSectionCollection class, ConfigurationSectionGroup class and ConfigurationSectionGroupCollection class; Implement ISettingsProviderService interface; Implement IApplicationSettingsProvider interface; ConfigurationValidationBase class; Implement IConfigurationSystem interface
- Create a custom Microsoft Windows Installer for .NET components by using the System.Configuration.Install namespace, and configure .NET Framework applications by using configuration files, environment variables, and the .NET Framework Configuration tool (Mscorcfg.msc).
May include but is not limited to: Installer class; Configure which runtime version a .NET Framework application should use; Configure where the runtime should search for an assembly; Configure the location of an assembly and which version of the assembly to use; Direct the runtime to use the DEVPATH environment variable when searching for assemblies; AssemblyInstaller class; ComponentInstaller class; Configure a .NET Framework application by using the .NET Framework Configuration tool (Mscorcfg.msc); ManagedInstallerClass; InstallContext class; InstallerCollection class; Implement IManagedInstaller interface; InstallEventHandler delegate; Configure concurrent garbage collection; Register remote objects by using configuration files
- Manage an event log by using the System.Diagnostics namespace.
May include but is not limited to: Write to an event log; Read from an event log; Create a new event log
- Manage system processes and monitor the performance of a .NET application by using the diagnostics functionality of the .NET Framework.
May include but is not limited to: Get a list of all running processes; Retrieve information about the current process; Get a list of all modules loaded by a process; PerformanceCounter class, PerformanceCounterCategory and CounterCreationData class; Start a process both by using and by not using command-line arguments; StackTrace class; StackFrame class



- Debug and trace a .NET Framework application by using the System.Diagnostics namespace.
May include but is not limited to: Debug class; Debugger class; Trace class, CorrelationManager class; TraceListener class; TraceSource class; TraceSwitch class; XmlWriterTraceListener class; DelimitedListTraceListener class and EventlogTraceListener class; Debugger attributes
- Embed management information and events into a .NET Framework application.
May include but is not limited to: Retrieve a collection of Management objects by using the ManagementObjectSearcher class and its derived classes; ManagementQuery class; Subscribe to management events by using the ManagementEventWatcher class

Implementing serialization and input/output functionality in a .NET Framework application (18 percent)

- Serialize or deserialize an object or an object graph by using runtime serialization techniques.
May include but is not limited to: Serialization interfaces; Serialization attributes; SerializationEntry structure and SerializationInfo class; ObjectManager class; Formatter class, FormatterConverter class, and FormatterServices class; StreamingContext structure
- Control the serialization of an object into XML format by using the System.Xml.Serialization namespace.
May include but is not limited to: Serialize and deserialize objects into XML format by using the XmlSerializer class; Control serialization by using serialization attributes; Implement XML serialization interfaces to provide custom formatting for XML serialization; Delegates and event handlers provided by the System.Xml.Serialization namespace
- Implement custom serialization formatting by using the Serialization Formatter classes.
May include but is not limited to: SoapFormatter; BinaryFormatter class
- Access files and folders by using the File System classes.
May include but is not limited to: File class and FileInfo class; Directory class and DirectoryInfo class; DriveInfo class and DriveType enumeration; FileSystemInfo class and FileSystemWatcher class; Path class; ErrorEventArgs class and ErrorHandler delegate; RenamedEventArgs class and RenamedEventHandler delegate
- Manage byte streams by using Stream classes.
May include but is not limited to: FileStream class; Stream Class (NOT Readers and Writer classes, as they are separate objectives); MemoryStream class; BufferedStream class
- Manage .NET Framework application data by using Reader and Writer classes.
May include but is not limited to: StringReader class and StringWriter class; TextReader class and TextWriter class; StreamReader class and StreamWriter class; BinaryReader class and BinaryWriter class
- Compress or decompress stream information in a .NET Framework application and improve the security of application data by using isolated storage.
May include but is not limited to: IsolatedStorageFile class; IsolatedStorageFileStream class; DeflateStream class; GZipStream class



Improving the security of .NET Framework applications by using the .NET Framework security features (20 percent)

- Implement code access security to improve the security of a .NET Framework application.
May include but is not limited to: SecurityManager class; CodeAccessPermission class; Modify the Code Access Security Policy at machine, user, and enterprise policy level by using the Caspol tool; PermissionSet class, NamedPermissionSet class, and PermissionSetCollection class; Standard Security interfaces
- Implement access control by using the System.Security.AccessControl classes.
May include but is not limited to: DirectorySecurity class, FileSecurity class, FileSystemSecurity class, and RegistrySecurity class; AccessRule class; AuthorizationRule class and AuthorizationRuleCollection class; CommonAce class, CommonAcl class, CompoundAce class, GeneralAce class, and GeneralAcl class; AuditRule class; MutexSecurity class, ObjectSecurity class, and SemaphoreSecurity class
- Implement a custom authentication scheme by using the System.Security.Authentication classes.
May include but is not limited to: Authentication algorithms and SSL protocols
- Encrypt, decrypt, and hash data by using the System.Security.Cryptography classes.
May include but is not limited to: DES class and DESCryptoServiceProvider class; HashAlgorithm class; DSA class and DSACryptoServiceProvider class; SHA1 class and SHA1CryptoServiceProvider class; TripleDES and TripleDESCryptoServiceProvider class; MD5 class and MD5CryptoServiceProvider class; RSA class and RSACryptoServiceProvider class; RandomNumberGenerator class; CryptoStream class; CryptoConfig class; RC2 class and RC2CryptoServiceProvider class; AssymmetricAlgorithm class; ProtectedData class and ProtectedMemory class; RijndaelManaged class and RijndaelManagedTransform class; CspParameters class; CryptoAPITransform class; Hash-Based Message Authentication Code (HMAC)
- Control permissions for resources by using the System.Security.Permission classes.
May include but is not limited to: SecurityPermission class; PrincipalPermission class; FileIOPermission class; StrongNameIdentityPermission class; UIPermission class; UriIdentityPermission class; PublisherIdentityPermission class; GaclIdentityPermission class; FileDialogPermission class; DataProtectionPermission class; EnvironmentPermission class; IUnrestrictedPermission interface; RegistryPermission class; IsolatedStorageFilePermission class; KeyContainerPermission class; ReflectionPermission class; StorePermission class; SiteIdentityPermission class; ZoneIdentityPermission class
- Control code privileges by using System.Security.Policy classes. May include but is not limited to: ApplicationSecurityInfo class and ApplicationSecurityManager class; ApplicationTrust class and ApplicationTrustCollection class; Evidence and PermissionRequestEvidence class; CodeGroup class, FileCodeGroup class, FirstMatchCodeGroup class, NetCodeGroup class, and UnionCodeGroup class; Condition classes; PolicyLevel and PolicyStatement class; IApplicationTrustManager interface, IMembershipCondition interface, and IIdentityPermissionFactory interface
- Access and modify identity information by using the System.Security.Principal classes.



May include but is not limited to: GenericIdentity class and GenericPrincipal class; WindowsIdentity class and WindowsPrincipal class; NTAccount class and SecurityIdentifier class; IIdentity interface and IPrincipal interface; WindowsImpersonationContext class; IdentityReference class and IdentityReferenceCollection class

Implementing interoperability, reflection, and mailing functionality in a .NET Framework application (11 percent)

- Expose COM components to the .NET Framework and .NET Framework components to COM.
May include but is not limited to: Import a type library as an assembly; Create COM types in managed code; Compile an interop project; Deploy an interop application; Qualify .NET types for interoperation; Apply Interop attributes, such as the ComVisibleAttribute class; Package an assembly for COM; Deploy an application for COM access.
- Call unmanaged DLL functions within a .NET Framework application, and control the marshalling of data in a .NET Framework application.
May include but is not limited to: Platform Invoke; Create a class to hold DLL functions; Create prototypes in managed code; Call a DLL function; Call a DLL function in special cases, such as passing structures and implementing callback functions; Create a new Exception class and map it to an HRESULT; Default marshalling behavior; Marshal data with Platform Invoke; Marshal data with COM Interop; MarshalAsAttribute class and Marshal class
- Implement reflection functionality in a .NET Framework application, and create metadata, Microsoft intermediate language (MSIL), and a PE file by using the System.Reflection.Emit namespace.
May include but is not limited to: Assembly class; Assembly Attributes; Info classes; Binder class and BindingFlags; MethodBase class and MethodBody class; Builder classes
- Send electronic mail to a Simple Mail Transfer Protocol (SMTP) server for delivery from a .NET Framework application.
May include but is not limited to: MailMessage class; MailAddress class and MailAddressCollection class; SmtpClient class, SmtpPermission class, and SmtpPermissionAttribute class; Attachment class, AttachmentBase class, and AttachmentCollection class; SmtpException class, SmtpFailedRecipientException class, and SmtpFailedRecipientsException class; SendCompletedEventHandler delegate; LinkedResource class and LinkedResourceCollection class; AlternateView class and AlternateViewCollection class



Implementing globalization, drawing, and text manipulation functionality in a .NET Framework application (11 percent)

- Format data based on culture information.
May include but is not limited to: Access culture and region information within a .NET Framework application; Format date and time values based on the culture; Format number values based on the culture; Perform culture-sensitive string comparison; Build a custom culture class based on existing culture and region classes.
- Enhance the user interface of a .NET Framework application by using the System.Drawing namespace.
May include but is not limited to: Enhance the user interface of a .NET Framework application by using brushes, pens, colors, and fonts; Enhance the user interface of a .NET Framework application by using graphics, images, bitmaps, and icons; Enhance the user interface of a .NET Framework application by using shapes and sizes.
- Enhance the text handling capabilities of a .NET Framework application, and search, modify, and control text within a .NET Framework application by using regular expressions.
May include but is not limited to: StringBuilder class; Regex class; Match class and MatchCollection class; Group class and GroupCollection class; Encode text by using Encoding classes.; Decode text by using Decoding classes.; Capture class and CaptureCollection class