98-372
MTA: Microsoft .NET Fundamentals

Audience

Microsoft Technology Associate

The Microsoft Technology Associate (MTA) is Microsoft's newest suite of technology certification exams that validate fundamental knowledge needed to begin building a career using Microsoft technologies.

Successful candidates earn MTA certificates as well as access to benefits on the Microsoft Certification member site.

This program:

- is targeted primarily at students attending high schools or two-year colleges.
- provides an appropriate entry point to a future career in technology.
- assumes some hands-on experience or training but does not assume on-the-job experience.

Target Audience

Candidates for this exam are seeking to prove knowledge and skills on Microsoft .NET fundamentals. Before taking this exam, candidates should have a solid foundational knowledge of the topics outlined in this preparation guide. It is recommended that candidates be familiar with the concepts of and have hands-on experience with the technologies described here either by taking relevant training courses or by working with tutorials and samples available on MSDN and in Microsoft Visual Studio. Candidates are expected to have some experience with a .NET language such as C# or Microsoft Visual Basic .NET.

Candidates for this exam are in the process of expanding their knowledge and skills in the following areas:

- .NET namespace and class organizations
- core .NET knowledge
- managed code theory
- memory management in .NET
- language parity

Objective Domain

1. **Understanding .NET Framework Concepts**
   - 1.1. Understand basic application settings.
     This objective may include but is not limited to: understanding app.config and web.config
   - 1.2. Understand events and event handling in the .NET Framework.
     This objective may include but is not limited to: understanding the event-driven programming model and event handlers, raising events, and implementing delegates
   - 1.3. Understand structured exception handling in the .NET Framework.
     This objective may include but is not limited to: understanding error handling concepts, exceptions, and exception types

2. **Understanding Namespaces and Classes in the .NET Framework**
   - 2.1. Understand .NET class hierarchies.
     This objective may include but is not limited to: understanding system classes, classifications of classes, and logical organization of classes
   - 2.2. Understand Object Oriented Concepts in the .NET Framework.
     This objective may include but is not limited to: understanding how inheritance works in .NET, polymorphism, and interfaces
   - 2.3. Understand .NET namespaces.
     This objective may include but is not limited to: understanding the reason for namespaces, the organization of namespaces in the class library, and how to use namespaces in an application
   - 2.4. Understand and create class libraries.
     This objective may include but is not limited to: understanding the logical grouping of classes and the logic behind class libraries (what they are, why they are important)

2.5. Understand and use different data types in the .NET Framework.
This objective may include but is not limited to: understanding intrinsic data types, values types, reference types, boxing, unboxing, and .NET collection classes

2.6. Understand generics.
This objective may include but is not limited to: understanding generics infrastructure, generic interfaces, generic delegates, contravariant and covariant generic type arguments, generic methods, verifiability, and constraints

3. Understanding .NET Code Compilation

3.1. Understand the fundamentals of Microsoft Intermediate Language (MSIL) and Common Language Infrastructure (CLI).
This objective may include but is not limited to: understanding what MSIL is, what the CLI is, how source code is compiled into MSIL, and how code is Just-in-Time (JIT) compiled

3.2. Understand the use of strong naming.
This objective may include but is not limited to: understanding why strong naming is used, how to sign assemblies to support strong naming, and Global Assembly Cache (GAC)

3.3. Understand version control.
This objective may include but is not limited to: understanding how .NET applications are versioned and how to run different versions on the same computer

3.4. Understand assemblies and metadata.
This objective may include but is not limited to: understanding .NET assemblies, shared assemblies, and metadata in .NET

4. Understanding I/O Classes in the .NET Framework

4.1. Understand .NET file classes.
This objective may include but is not limited to: understanding read/write file classes and stream readers and writers

4.2. Understand console I/O.
This objective may include but is not limited to: understanding System.Console classes for input and output

4.3. Understand XML classes in the .NET Framework.
This objective may include but is not limited to: understanding XMLReader, XMLWriter, and XML Schemas

5. Understanding Security

5.1. Understand the System Security namespace.
This objective may include but is not limited to: understanding permissions and what cryptography is

5.2. Understand authentication and authorization.
This objective may include but is not limited to: understanding code access security, access

control, and policies

6. Understanding .NET Languages

6.1. Understand language interoperability.
This objective may include but is not limited to: calling code written in one language from another language, understanding .NET language parity

6.2. Understand type safety.
This objective may include but is not limited to: understanding memory type safety, type safety in classes, strong types, and security policies

7. Understanding Memory Management

7.1. Understand resource allocation.
This objective may include but is not limited to: understanding garbage collection and memory allocation, understanding stack versus heap

7.2. Understand the difference between managed and unmanaged applications.
This objective may include but is not limited to: understanding why managed code is called managed code, understanding the differences between coding in managed versus unmanaged code