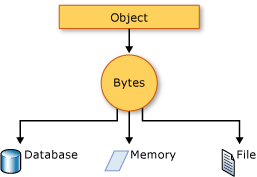
**Serialization (C# and Visual Basic)**

**Visual Studio 2010**

Serialization is the process of converting an object into a stream of bytes in order to persist it to memory, a database, or a file. Its main purpose is to save the state of an object in order to be able to recreate it when needed. The reverse process is called deserialization.

[**How Serialization Works**](javascript:void(0))

This illustration shows the overall process of serialization.



The object is serialized to a stream, which carries not just the data, but information about the object's type, such as its version, culture, and assembly name. From that stream, it can be stored in a database, a file, or memory.

**Uses for Serialization**

Serialization allows the developer to save the state of an object and recreate it as needed, providing storage of objects as well as data exchange. Through serialization, a developer can perform actions like sending the object to a remote application by means of a Web Service, passing an object from one domain to another, passing an object through a firewall as an XML string, or maintaining security or user-specific information across applications.

**Making an Object Serializable**

To serialize an object, you need the object to be serialized, a stream to contain the serialized object, and a [Formatter](http://msdn.microsoft.com/en-us/library/system.runtime.serialization.formatter.aspx). [System.Runtime.Serialization](http://msdn.microsoft.com/en-us/library/system.runtime.serialization.aspx) contains the classes necessary for serializing and deserializing objects.

Apply the [SerializableAttribute](http://msdn.microsoft.com/en-us/library/system.serializableattribute.aspx) attribute to a type to indicate that instances of this type can be serialized. A [SerializationException](http://msdn.microsoft.com/en-us/library/system.runtime.serialization.serializationexception.aspx) exception is thrown if you attempt to serialize but the type does not have the [SerializableAttribute](http://msdn.microsoft.com/en-us/library/system.serializableattribute.aspx) attribute.

If you do not want a field within your class to be serializable, apply the [NonSerializedAttribute](http://msdn.microsoft.com/en-us/library/system.nonserializedattribute.aspx) attribute. If a field of a serializable type contains a pointer, a handle, or some other data structure that is specific to a particular environment, and the field cannot be meaningfully reconstituted in a different environment, then you may want to make it nonserializable.

If a serialized class contains references to objects of other classes that are marked [SerializableAttribute](http://msdn.microsoft.com/en-us/library/system.serializableattribute.aspx), those objects will also be serialized.

[**Binary and XML Serialization**](javascript:void(0))

Either binary or XML serialization can be used. In binary serialization, all members, even those that are read-only, are serialized, and performance is enhanced. XML serialization provides more readable code, as well as greater flexibility of object sharing and usage for interoperability purposes.

**Binary Serialization**

Binary serialization uses binary encoding to produce compact serialization for uses such as storage or socket-based network streams.

**XML Serialization**

XML serialization serializes the public fields and properties of an object, or the parameters and return values of methods, into an XML stream that conforms to a specific XML Schema definition language (XSD) document. XML serialization results in strongly typed classes with public properties and fields that are converted to XML. [System.Xml.Serialization](http://msdn.microsoft.com/en-us/library/system.xml.serialization.aspx) contains the classes necessary for serializing and deserializing XML.

You can apply attributes to classes and class members in order to control the way the [XmlSerializer](http://msdn.microsoft.com/en-us/library/system.xml.serialization.xmlserializer.aspx) serializes or deserializes an instance of the class.

**SOAP Serialization**

XML serialization can also be used to serialize objects into XML streams that conform to the SOAP specification. SOAP is a protocol based on XML, designed specifically to transport procedure calls using XML. As with regular XML serialization, attributes can be used to control the literal-style SOAP messages generated by an XML Web service.

[**Basic and Custom Serialization**](javascript:void(0))

Serialization can be performed in two ways, basic and custom. Basic serialization uses the .NET Framework to automatically serialize the object.

**Basic Serialization**

The only requirement in basic serialization is that the object has the [SerializableAttribute](http://msdn.microsoft.com/en-us/library/system.serializableattribute.aspx) attribute applied. The [NonSerializedAttribute](http://msdn.microsoft.com/en-us/library/system.nonserializedattribute.aspx) can be used to keep specific fields from being serialized.

When you use basic serialization, the versioning of objects may create problems, in which case custom serialization may be preferable. Basic serialization is the easiest way to perform serialization, but it does not provide much control over the process.

**Custom Serialization**

In custom serialization, you can specify exactly which objects will be serialized and how it will be done. The class must be marked [SerializableAttribute](http://msdn.microsoft.com/en-us/library/system.serializableattribute.aspx) and implement the [ISerializable](http://msdn.microsoft.com/en-us/library/system.runtime.serialization.iserializable.aspx) interface.

If you want your object to be deserialized in a custom manner as well, you must use a custom constructor.

[**Designer Serialization**](javascript:void(0))

Designer serialization is a special form of serialization that involves the kind of object persistence usually associated with development tools. Designer serialization is the process of converting an object graph into a source file that can later be used to recover the object graph. A source file can contain code, markup, or even SQL table information. For more information, see [Designer Serialization Overview](http://msdn.microsoft.com/en-us/library/ms171834.aspx).