**Using Conversion Operators**

## Example

This is an example of an explicit conversion operator. This operator converts from the type [Byte](https://msdn.microsoft.com/en-us/library/system.byte.aspx) to a value type called Digit. Because not all bytes can be converted to a digit, the conversion is explicit, meaning that a cast must be used, as shown in the Main method.

|  |
| --- |
| struct Digit  {  byte value;  public Digit(byte value) //constructor  {  if (value > 9)  {  throw new System.ArgumentException();  }  this.value = value;  }  public static explicit operator Digit(byte b) // explicit byte to digit conversion operator  {  Digit d = new Digit(b); // explicit conversion  System.Console.WriteLine("Conversion occurred.");  return d;  }  }  class TestExplicitConversion  {  static void Main()  {  try  {  byte b = 3;  Digit d = (Digit)b; // explicit conversion  }  catch (System.Exception e)  {  System.Console.WriteLine("{0} Exception caught.", e);  }  }  }  // Output: Conversion occurred. |

## Example

This example demonstrates an implicit conversion operator by defining a conversion operator that undoes what the previous example did: it converts from a value class called Digit to the integral [Byte](https://msdn.microsoft.com/en-us/library/system.byte.aspx) type. Because any digit can be converted to a [Byte](https://msdn.microsoft.com/en-us/library/system.byte.aspx), there's no need to force users to be explicit about the conversion.

|  |
| --- |
| struct Digit  {  byte value;  public Digit(byte value) //constructor  {  if (value > 9)  {  throw new System.ArgumentException();  }  this.value = value;  }  public static implicit operator byte(Digit d) // implicit digit to byte conversion operator  {  System.Console.WriteLine("conversion occurred");  return d.value; // implicit conversion  }  }  class TestImplicitConversion  {  static void Main()  {  Digit d = new Digit(3);  byte b = d; // implicit conversion -- no cast needed  }  }  // Output: Conversion occurred. |