PowerShell ABC's - N is for Numbers

Joe Pruitt, 2009-08-01

Welcome to this addition of the PowerShell ABC's where you'll find 26 posts detailing a component of the PowerShell scripting language, one letter at a time. Today's letter is the letter "N" and for this letter I'm going to discuss one of the core types of objects you'll likely be dealing with: **Numbers**.

PowerShell supports all of the basic .NET numeric types and performs conversions to and from the different types as needed. The types, along with the PowerShell type names are in the following table

.NET Full Type Name	PowerShell Short Type Name	Example
System.Int32	[int]	1
System.Int64	[long]	4294967296
System.Double	[double]	1.2
System.Decimal	[decimal]	1d

Default Type Determiniation

If you don't specify a type for a literal, the system will figure out the best type and way to represent the number.

Integers (System.Int32) will be used by default and if the literal value is too large to fit into a 32-bit integer, then a long 64-bit integer (System.Int64) will be used. If it's too large for a 64 bit integer, or if it contains a decimal point, then a double (System.Double) will be used.

The System.Single type is supported but is typically not used as it provides no advantages over the System.Double type.

There is one condition where you will want to specify the type and that would be for the decimal type (System.Decimal). This is done by placing a "d" after the number with no whitespace in between.

Multiplier Suffixes

The PowerShell team was kind enough to add some special suffixes for numbers to make managing larger numbers easier. The following table lists the special powers of two modifiers supported by PowerShell

Multiplier Suffix	Multiplication Factor	Example	Equivalent Value	.NET Type
kb	1024	1kb	1024	System.Int32
		1.1kb	1126.4	System.Double
mb	1024*1024	1mb	1048576	System.Int32
		1.1mb	1153433.6	System.Double
gb	1024*1024*1024	1gb	1073741824	System.Int32
		1.1gb	1181116006.4	System.Double
tb*	1024*1024*1024*1024	1tb	1099511627776	System.Int64
		1.1tb	1209462790553.6	System.Double
pb*	1024*1024*1024*1024*1024	1pb	1125899906842624	System.Int64
		1.1pb	1.23848989752689E+15	System.Double

*Note, that tb (terabyte) and pb (petabyte) were added in PowerShell v2.

Don't forget the hexidecimals

The last type of number literals are hexidecimals. Hexidecimal numbers are in base-16 and thus use the 16 distinct symbols [0-9] and [A-F] to represent the values 0-15 respectively. PowerShell follows the same notation as C, C++, C#, and many other languages in that you prefix the number with the sequence Zero-X ("0x") and allowing the letters A-F (in either case) as extra digits. 0x10 -> 16, 0xF -> 16, and so on.

F5 Networks, Inc. | 401 Elliot Avenue West, Seattle, WA 98119 | 888-882-4447 | f5.com

F5 Networks, Inc.	F5 Networks	F5 Networks Ltd.	F5 Networks
Corporate Headquarters	Asia-Pacific	Europe/Middle-East/Africa	Japan K.K.
info@f5.com	apacinfo@f5.com	emeainfo@f5.com	f5j-info@f5.com