Introducing wide number support and strict radix checks

2026-01-20 - Comments (0) - Announcements

We are upgrading PowerView’s PRACTICE scripting language to support wide numbers, with the current limit of **64 bits** being increased to **2080 bits**.

At the same time, we are introducing **strict radix** checking for numerical constants. This means, that numerical constants must have a radix identifier: `0x` prefix for hexadecimal or a `.` postfix for decimal numbers.

These changes could influence the behavior of some scripts.

The changes will be activated by default starting with PowerView software version **2026.02**, by changing the `SETUP.RADIX` default setting from `Hex` (hereafter called "previous system") to `WideStrict` ("new system")

We strongly recommend checking your scripts in advance regarding compatibility with the new number system. Using the command `SETUP.RADIX Hex` to keep backward compatibility beyond this time is possible, but not recommended.

Wide Number Support

In the previous system, with default setting `SETUP.RADIX Hex`, all numbers are treated as signed **64-bit** values. In the new system, `SETUP.RADIX WideStrict` is used as default setting, PowerView accepts up to **2080-bit** wide numbers. Future versions may support wider numbers as needed.

The introduction of wide numbers has some side effects that may affect script compatibility.

1. Comparing Hexadecimal Values with Negative Numbers

In the previous system, numbers are treated as signed numbers and thus if the most significant bit (63) is set, the number is interpreted as negative number. Therefore, `0xFFFFFFFFFFFFFFFF` equals `-1`. Unsigned 64-bit values are thus not possible in PRACTICE.

In the wide number system introduced in R.2026.02, all numbers are signed, and the sign will not change on arithmetic overflows. Comparisons of a positive number and a negative number will always return `FALSE`.

Example	SETUP.RADIX Hex	SETUP.RADIX Wide*
PRINT 0xFFFFFFFFFFFFFFFF==-1.	TRUE	FALSE
PRINT 0xFFFFFFFFFFFFFFFE==-2.	TRUE	FALSE
...	TRUE	FALSE
PRINT 0x8000000000000001==-9223372036854775807.	TRUE	FALSE
PRINT 0x8000000000000000==-9223372036854775808.	ERROR	TRUE
PRINT -0x1==-1.	TRUE	TRUE

Note

We use the notation **SETUP.RADIX Wide*** here as a wildcard to represent all commands beginning with **SETUP.RADIX Wide**, for example **SETUP.RADIX WideHex** or **SETUP.RADIX WideDecimal**.

Unsigned values may need to be converted to signed values in scripts. For 8-, 16- or 32-bit numbers conversion Powerview functions already exist. A new Powerview function is now available for 64-bit numbers.

Width	Correct hexadecimal to signed value conversion	Notes
Byte	PRINT CONVert.SignedByte(0xFF)==-1	required independent of SETUP.RADIX
Word	PRINT CONVert.SignedWord(0xFFFF)==-1	required independent of SETUP.RADIX

Long	PRINT CONVert.SignedLong(0xFFFFFFFF)==-1	required independent of SETUP.RADIX
Quad	PRINT CONVert.SignedQuad(0xFFFFFFFFFFFFFFFF)==-1	new requirement for **SETUP.RADIX Wide***; introduced in rev. 155331

2. Shifting Left

In the previous system, the maximum shift value is limited to 63. If the most significant bit (63) is set after the shift, the resulting number is negative.

In the new wide number system, the maximum shift value is not limited. Digits beyond the maximum number width are cut off. The sign of the number will not be affected by this operation.

Example	SETUP.RADIX Hex/Decimal	SETUP.RADIX Wide*	Notes
PRINT 0x0000000000000002<<62.	8000000000000000 (8 + 15 zeros)	8000000000000000 (8 + 15 zeros)	
PRINT 0x0000000000000002<<63.	0	10000000000000000 (1 + 16 zeros)	
PRINT 0x0000000000000001<<63.	8000000000000000 (8 + 15 zeros)	8000000000000000 (8 + 15 zeros)	
PRINT 0x0000000000000001<<64.	8000000000000000 (8 + 15 zeros)	10000000000000000 (1 + 16 zeros)	Due to the shift limitation in Hex/Decimal mode, this is effectively a 63-bit shift.
PRINT 0x0000000000000001<<2080.	8000000000000000 (8 + 15 zeros)	0	All bits shifted beyond most significant bit and cut off.

3. Sign-Extension of Bit Fields

Some scripts make use of the 64-bit signed integer behavior in order to sign-extend a value of a bit field. In the new wide number system, this sign-extension “trick” is not possible, because the number sign is not affected by shift operations.

Example: sign extend 3-bit field	SETUP.RADIX Hex/Decimal	SETUP.RADIX Wide*	Notes
PRINT (0x06<<61.)>>61.	0FFFFFFFFFFFFFFE (=-2)	6	No sign extension in Wide modes
PRINT (0x06<<2077.)>>2077.	0	6	No sign extension in Wide modes

In order to write backwards-compatible scripts, use the functions `CONVert.SignedBITS()` or the function `CONVert.SignedQuad()`, which are dedicated to perform sign-extensions of bit fields:

Example: sign extend 3-bit field	Backwards-compatible method	Notes
PRINT (0x06<<61.)>>61.	PRINT CONVert.SignedQUAD(0x06<<61.)>>61.	Function introduced in rev. 155331
PRINT (0x06<<61.)>>61.	PRINT CONVert.SignedBITS(0x06, 3., 0.)	Function introduced in rev. 183018

4. Operations that Rely on 64-bit Integer Overflows

In the previous number system, all arithmetic operations perform modulo 2^64 calculation.

In the new number system, all arithmetic operations that produce results beyond the size limit are unpredictable.

Example: sign extend 3-bit field	SETUP.RADIX Hex	SETUP.RADIX Wide*
PRINT 0xFFFFFFFFFFFFFFFE+0x3	1	10000000000000001
PRINT 0x7FFFFFFFFFFFFFFF*0x5	7FFFFFFFFFFFFFFB	27FFFFFFFFFFFFFFB

In order to write backwards-compatible scripts, use the function `CONVert.SignedQUAD()`:

Example: sign extend 3-bit field	Backwards-compatible method
PRINT 0xFFFFFFFFFFFFFFFE+0x3	PRINT CONVert.SignedQUAD(0xFFFFFFFFFFFFFFFE+0x3)
PRINT 0x7FFFFFFFFFFFFFFF*0x5	PRINT CONVert.SignedQUAD(0x7FFFFFFFFFFFFFFF*0x5)

5. Converting Negative Values to Unsigned Hexadecimal for Output

Some of the PowerView output commands will, by default, produce undecorated output. The commands are `PRINT`, `WRITE`, `APPEND`, `INTERCOM.PipeWRITE`, `PRinTer.PRINT` and `STOP`. If a negative hexadecimal value is passed to one of these commands, the command will implicitly output these values as unsigned hexadecimals.

Examples of implicit and explicit hexadecimal conversions	Output format
PRINT -0x123	implicit unsigned hexadecimal output
PRINT %Hex -0x123	**explicit unsigned hexadecimal output**
PRINT %HexS -0x123	explicit signed hexadecimal output
PRINT -456.	**decimal value without specified format is not changed**
PRINT %Hex -456.	explicit unsigned hexadecimal output of decimal value

In the previous system, the conversion to unsigned hexadecimal was performed as the two’s complement of a 64-bit number.

In the new wide number system, the two’s complement is also performed, but the resulting number width is the smallest multiple of 64, that is wide enough to hold the resulting value.

Here is a comparison of previous and new output:

Examples of implicit unsigned hexadecimal output	SETUP.RADIX Hex/Decimal	SETUP.RADIX Wide*
PRINT -0x7	0FFFFFFFFFFFFFFF9	0FFFFFFFFFFFFFFF9
PRINT -0xFFFFFFFFFFFFFFFF	1	0FFFFFFFFFFFFFFFF0000000000000001

In order to achieve consistent output, there are several options available:

5.1. Use Format Option %DECOrated with Above Commands

Note

Please note that doing so will hexadecimal numbers with prefix “0x” and end decimal numbers with suffix “.”

Below table shows how `%DECOrated` changes the output, if `SETUP.RADIX Wide*` is used. While the examples are made using print, the same is true for all commands mentioned above.

Command (with SETUP.RADIX Wide*)	Output
PRINT 5.	5
PRINT %DECOrated 5	**5**
PRINT -0x7	0FFFFFFFFFFFFFF9
PRINT %DECOrated -0x7	**-0x7**

Note

`ECHO` can be used as replacement for `PRINT %DECOrated`

5.2 Use Commands or Functions For Advanced Formatting

In order to achieve consistent output, scripts should make use of output commands with advanced formatting. PowerView provides several commands that allow printf-like formatting.

Function	Simple formatting	Advanced formatting
Write to message window	PRINT [%<format>] <value>	PRINTF "<format_string>" <values>
Write to file	WRITE [%<format>] <value>	WRITEF "<format_string>" <values>
Print to Printer	PRinTer.PRINT [%<format>] <value>	PRinTer.PRINTF "<format_string>" <values>
Write to Intercom Pipe	**INTERCOM.PipeWRITE [%<format>] <value>**	**INTERCOM.PipeWRITEF "<format_string>" <values>**
Append to file	Append to file	Not available, use STOP %DECOrated
Stop Script Execution with message	**STOP [%<format>] <value>**	**Not available, use STOP %DECOrated**
Write to Macro	-	SPRINTF <macro> "<format_string>" <values>

5.3 Use Commands with Advanced Formatting with Advanced Number Width, or FORMAT.HEX()

Use commands with advanced formatting for standard number widths (8, 16, 32, 64 bit), or the `FORMAT.HEX()` function for arbitrary lengths:

Function	Command
Print Hex value as unsigned 32-bit hex	PRINTF “0x%016lx” -0x123
Print Hex value as unsigned 64-bit hex	PRINTF “0x%016llx” -0x123
Format Hex value as unsigned hex with 37. digits	PRINT FORMAT.HEX(37., -0x123)

Strict Radix Identifier Requirement

Overview and Time Plan

In order align to PowerView’s numeric constants with modern programming languages like C or Python, the default radix for numeric constants without an explicit radix identifier will be changed in three phases, as detailed in the table below. The change will be performed in parallel to the introduction of the wide number system.

The change will be performed in three phases, accompanied by changing the `SETUP.RADIX` defaults:

Phase	SETUP.RADIX default	Behavioral changes	Change due to
1	Hex -> WideStrictWarn	Explicit radix identifier required in command line. A warning is printed during script execution.	2026/02
2	WideStrictWarn -> WideStrict	PowerView throws error for numerical constants without explicit radix identifier	2026/09
3	WideStrict -> WideDecimal	Numerical constants without explicit radix identifier are interpreted as decimal value	to be defined

The Changes in Detail

In the previous number system, which defaults to `SETUP.RADIX Hex`, numerical constants without radix are, by default, interpreted as hexadecimal values. PowerView also supports setting the radix to `Decimal`. Numbers are interpreted according to below table:

Radix Identifier	Example(s)	SETUP.RADIX Hex (default)	SETUP.RADIX Decimal
0x<digits>	0x1200	Hexadecimal	Hexadecimal
<digits>.	123456.	Decimal	Decimal
<digits>.<digits> <digits>.<digits>E<digits>	37.7 37.7E7	Floating Point	Floating Point
no identifier, only numbers	42, 678	**Hexadecimal**	**Decimal**
no identifier, numbers + 'a'…'f'	4B 0a5	**Hexadecimal**	**ERROR**

With the introduction of the new wide number system, we will switch the `SETUP.RADIX` default from `Hex` to `WideStrictWarn` or `WideStrict`. This requires all hexadecimal numbers to start with "`0x`" and all decimal numbers to end with a "`.`". This intermediate step ensures that all numbers used in scripts have radix identifiers and can be interpreted correctly, independent of the `SETUP.RADIX` setting.

In the new wide number system, numerical constants are interpreted according to below table:

Radix Identifier	Example(s)	SETUP.RADIX WideStrict / WideStrictWarn	SETUP.RADIX WideDecimal	SETUP.RADIX WideHex
0x<digits>	0x1200	Hexadecimal	Hexadecimal	Hexadecimal
<digits>.	123456.	Decimal	Decimal	Decimal
<digits>.<digits> <digits>.<digits>E<digits>	37.7 37.7E7	Floating Point	Floating Point	Floating Point
no identifier, only numbers	42 678	ERROR / Warning	Decimal	Hexadecimal
no identifier, numbers + 'a'…'f'	4B 0a5	ERROR / Warning	ERROR	Hexadecimal

Action Required

We strongly recommend checking your scripts regarding compatibility with the new radix default. As the command `SETUP.RADIX WideStrict` was already introduced in PowerView build 159006 (April 26, 2023), you have the chance to test and update your scripts before the default setting changes.

Currently used Radix setting	What you can do
None / Default setting SETUP.RADIX Hex	Test your scripts by temporarily setting `SETUP.RADIX WideStrict`. As numeric constants without radix identifier will then cause an error, it is easy to spot those constants. For scripts, you may also use the `SETUP.RADIX WideStrictWarn` setting - which just causes warnings during script execution. Make the numerical constants compatible by adding the prefix `0x`
SETUP.RADIX Decimal	Change `SETUP.RADIX Decimal` to `SETUP.RADIX WideDecimal` to benefit from the wide number support.