AdiIRC - $base - # 9

Added in 1.9.0

$base(N,inbase,outbase,zeropad,precision)

Converts number N from inbase to outbase. The last two parameters are optional.

In version 3.3+, it should work reliably to at least base64, otherwise to base32.

Parameters

N - The number to convert. Values above 2^53 are (AdiIRC only)
inbase - The base to convert from. (2-64) 37-64 are (AdiIRC only)
outbase - The base to convert to. (2-64) 37-64 are (AdiIRC only)
zeropad - If length is higher than zeropad, pad numbers with zeroes to the specified zeropad length. Values above 100 are (AdiIRC only)
precision - Truncate fractions/decimal places to the specified precision. (Can't use if Zeropad is blank) Values above 6 are (AdiIRC only)

base: From 2-36, base N uses the first N characters from the alphanumeric alphabet:
0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ
From 37-64, base N uses the first N characters from the 64-char mime alphabet:
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/

Notes: base 2-36 are case-insensitive, but base 37-64 are case-sensitive. This base 32 uses the first 32 characters of the base36 alphabet, which is different than the 'a' switch in $encode which uses A-Z and 2-7. Even though this base 64 uses the same alphabet as $encode's mime 'm' switch, this is NOT the same as mime-encoding. It does support the '=' padding used by mime, and returns results compatible with mime ONLY if the base64 string's length is a multiple of 4, or the base64 string used by $base must be left-padded with "A" into becoming a length that's a multiple of 4 compatible with the mime encoding of a string whose length is a multiple of 3. zero-pad means inserting the lowest character of that base's 'alphabet', which for base 37-64 is the "A" character.

Examples

```
; Returns F
//echo -ag $base(15,10,16)

; Returns 1.8
//echo -ag base(1.5,10,16)

; Returns 002
//echo -ag $base(2,10,16,3)

; Returns fraction chopped to 4 digits
//echo -a $base($pi,10,10,0,4)

; Zero-Pad for base 37-64 uses "A", while base 2-36 uses '0'
//echo -a $base(deadbeef,16,64,20)
//echo -a $base(deadbeef,16,10,20)

; Using same in/out base to ensure number contains at least 12 digits:
//echo -a $base($ticks,10,10,12)

; Accurate results above the 2^53 limit:
//var %a $sha256(abc) | echo -a $lower($base($base(%a,16,10),10,16)) same as | echo -a %a

; Base 37-64 use case-sensitive alphabets
//echo -a $base(deadbeef,64,10) vs $base(DEADBEEF,64,10)

; Only if inbase is 16, optional '0x' prefix does not affect output
//echo -a $base(0xdeadbeef,16,10) same as $base(deadbeef,16,10)

; Allows 2-35 using invalid base36 characters, or 37-63 using invalid base64 chars
//echo -a $base(MZ,10,10) is $base(M,36,10) * 10 + $base(Z,10,10) = 255

; Demonstrates how 64 in $base is compatible with 'm' mime in $encode only if mime string's length
```

```
 is a multiple of 4. If edit the ++adiirc to be a mime string that's not a multiple of 4, the outp
ut no longer matches.
//var -s %mime ++adiirc , %base64 %mime | ;while ($calc($len(%base64) % 4)) var -s %base64 A $+ %b
ase64 | bset -t &v 1 %mime | noop $decode(&v,bm) $encode(&v,bx) | echo 4 -a $bvar(&v,1-).text | ec
ho 3 -a $base(%base64,64,16)
```