Network Working Group
Request for Comments: 2550
Category: Stinkards Track

S. Glassman
M. Manasse
J. Mogul
Compaq Computer Corporation

1 April 1999

Y10K and Beyond

Status of this Memo

This memo provides information for the Internet community. It does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

Copyright Notice

Abstract

As we approach the end of the millennium, much attention has been paid to the so-called "Y2K" problem. Nearly everyone now regrets the short-sightedness of the programmers of yore who wrote programs designed to fail in the year 2000. Unfortunately, the current fixes for Y2K lead inevitably to a crisis in the year 10,000 when the programs are again designed to fail.

This specification provides a solution to the "Y10K" problem which has also been called the "YAK" problem (hex) and the "YXK" problem (Roman numerals).

1. Introduction, Discussion, and Related Work

Many programs and standards contain, manipulate and maintain dates. Comparing and sorting dates is a common activity. Many different formats and standards for dates have been developed and all have been found wanting.

Early date formats reserved only two digits to represent the year portion of a date. Programs that use this format make mistakes when dealing with dates after the year 2000. This is the so-called Y2K problem.

The most common fix for the Y2K problem has been to switch to 4-digit years. This fix covers roughly the next 8,000 years (until the year 9999) by which time, everyone seems convinced that all current programs will have been retired. This is exactly the faulty logic and lazy programming practice that led to the current Y2K problem! Programmers and designers always assume that their code will eventually disappear, but history suggests that code and programs are often used well past their intended circumstances.

The 4-digit year leads directly to programs that will fail in the year 10,000. This proposal addresses the Y10K problem in a general way that covers the full range of date and time format issues.

1.1 Current approaches

A large number of approaches exist for formatting dates and times. All of them have limitations. The 2-digit year runs into trouble next year. The 4-digit year hits the wall in the year 10,000. A 16-bit year runs out in the year 65,536. A 32-bit counter for the number of seconds since 1970 [UNIX] wraps in 2038. A 32-bit counter for the number of milli-seconds since booting crashes a Windows (TM) PC in 49.7 days [Microsoft].

In this specification, we focus on the Y10K problems since they are most common and a large number of existing standards and protocols are susceptible to them (section 7). These standards, and new proposals on their way, will lead to a serious world-wide problem unless efforts are made now to correct the computing, government, and business communities.

Already, a small cottage industry is popping up to deal with the Y10K problem [YUCK]. We encourage these efforts and, in the coming years, this effort can only grow in size and importance.

1.2 A Fixed Format Y10K Fix

At the time of this writing, only one proposal [Wilborne] directly deals with the Y10K problem. In that proposal, dates are represented as decimal numbers with the dates compared numerically. The proposed format is simply YYYYYMMDD - i.e. 5-digit years.

To allow numerical comparison of dates, this representation requires a completely fixed representation for the date. There can be no optional fields, the date resolution is limited to the granularity of one day, and this solution fails in the year 100,000 (Y100K).

1.2.2 Limitations of Numerical Comparison

While sufficient for the specific Y10K problem, this solution is limited. Even if extended for 6-digit years, it fails on 32-bit systems (and future 32-bit system emulators) after the date represented by the number 2147481231 (December 31, 214748) leading to a Y214749 problem. Similarly, 64-bit and 128-bit systems also will fail, although somewhat later (after December 31, 922,337,203,685,477 and December 31, 17,014,118,346,046,923,173,168,730,371,588,410 respectively).

1.2.3 Granularity Issues

The granularity problems of a fixed format date can be improved by extending the date format to include greater precision in the date. However, since numerical comparison of dates requires a fixed representation date, an extended format can not provide sufficient resolution for all foreseeable needs.

For instance, if the precision were extended to the femto-second range the date format would become YYYYYMMDDHHMMSSmmmuuunnnpppfff (year, month, day, hour, minute, second, milli-second, micro-second, nano-second, pico-second, and femto-second). The additional 21 digits of this format limit the set of representable dates. Compared to 1.2.2, the 32-bit and 64-bit forms of the date are instantly exceeded, while the 128-bit version would be viable - expiring on December 31, 17,014,118,346,046.

1.2.3.1 Extrapolation of Future Granularity Issues

However, a simple extrapolation of Moore’s law shows that even femto-second resolution will soon be inadequate. Projecting current CPU clock speeds forward, a femto-second clock speed will be achieved in only 30 years. And, by the year 10,000 the projected clock speed of the Intel Pentium MMDCLXVI (TM) will be approximately 10 ** (-1609) seconds.

This discussion clearly shows that any fixed-format, integer representation of a date is likely to be insufficiently precise for future uses.

1.2.3.2 Floating Point Is No Solution

The temptation to use floating point numbers to represent dates should be avoided. Like the longer fixed-format, integer representations of the date, floating point representations merely delay the inevitable time when their range is exceeded. In addition,

the well known problems of real numbers - rounding, de-normalization, non-uniform distribution, etc. - just add to the problems of dealing with dates.

2 Structure of Y10K Solution

Any Y10K solution should have the following characteristics.

2.1 Compatibility

The format must be compatible with existing 4-digit date formats. Y2K compliant programs and standards must continue to work with Y10K dates before the year 10,000. Y10K compliant programs can gradually be developed over time and coexist with non-Y10K compliant programs.

2.2 Simplicity and Efficiency

Y10K dates must allow dates after 10,000 to be easily identified. Within a program, there must be a simple procedure for recognizing the Y10K dates and distinguishing them from legacy dates.

2.3 Lexical Sorting

Y10K dates must be sortable lexically based on their ASCII representation. The dates must not require specialized libraries or procedures.

2.4 Future Extensibility

Y10K dates must support arbitrary precision dates, and should support dates extending arbitrarily far into the future and past. Y10K dates from different eras and with different precisions must be directly comparable and sortable.

2.4.1 Environmental Considerations

The known universe has a finite past and future. The current age of the universe is estimated in [Zebu] as between 10 ** 10 and 2 * 10 ** 10 years. The death of the universe is estimated in [Nigel] to occur in 10 ** 11 - years and in [Drake] as occurring either in 10 ** 12 years for a closed universe (the big crunch) or 10 ** 14 years for an open universe (the heat death of the universe).

In any case, the prevailing belief is that the life of the universe (and thus the range of possible dates) is finite.

2.4.2 Transcending Environmental Considerations

However, we might get lucky. So, Y10K dates are able to represent any possible time without any limits to their range either in the past or future.

Y10K compliant programs MAY choose to limit the range of dates they support to those consistent with the expected life of the universe. Y10K compliant systems MUST accept Y10K dates from 10 ** 12 years in the past to 10 ** 20 years into the future. Y10K compliant systems SHOULD accept dates for at least 10 ** 29 years in the past and future.

3 Syntax Overview

The syntax of Y10K dates consists of simple, printable ASCII characters. The syntax and the characters are chosen to support a simple lexical sort order for dates represented in Y10K format. All Y10K dates MUST conform to these rules.

Every Y10K date MUST begin with a Y10K year. Following the year, there MAY be an arbitrary sequence of digits. The digits are interpreted as MMDDHHMMSSmmmuuunnnpppfff... (month, day, hour, minute, second, milli-second, micro-second, nano-second pico-second, femto-second, etc. - moving left to right in the date, digits always decrease in significance).

All dates and times MUST be relative to International Atomic Time (TAI) [NRAO].

When comparing dates, a date precedes every other date for which it is a prefix. So, the date "19990401000000" precedes the date "199904010000000000". In particular, dates with the format YYYYMMDD are interpreted to represent the exact instant that the day begins and precede any other date contained in that day.

3.1 Years 1 - 9999

The current 4-digit year syntax covers all years from 1000 - 9999. These years are represented as 4 decimal digits. Leading 0's MUST be added to the years before 1000 to bring the year to 4 digits. Files containing legacy pre-Y1K [Mike] dates will have to be converted.

3.2 Years 10,000 through 99,999

Four digits are not sufficient to represent dates beyond the year 9999. So, all years from 10,000 - 99,999 are represented by 5 digits preceded by the letter 'A'. So, 10,000 becomes "A10000" and 99,999

becomes "A99999". Since 'A' follows '9' in the ASCII ordering, all dates with 5-digit years will follow all dates with 4-digit years (for example, "A10000" will sort after "9999"). This gives us the sort and comparison behaviour we want.

3.3 Years 100,000 up to 10 ** 30

By a simple generalization of 3.2, 6-digit years are preceded by the letter 'B', 7-digit years by 'C', etc. Using just the 26 upper-case ASCII characters, we can cover all years up to 10**30 (the last year representable is "Z999999999999999999999999999999"). Again, since the ASCII characters are sorted alphabetically, all dates sort appropriately.

3.4 Years 10 ** 30 and beyond (Y10**30)

As discussed in 2.4.1, the end of the universe is predicted to occur well before the year 10 ** 30. However, if there is one single lesson to be learned from the current Y2K problems, it is that specifications and conventions have a way of out living their expected environment. Therefore we feel it is imperative to completely solve the date representation problem once and for all.

3.4.1 Naive Approach for Y10**30 Problem

The naive solution is to prepend a single '^' (caret) - caret sorts after all letters in the ASCII order) before all years from 10 ** 30 to 10 ** 56. Thus the year "Z999999999999999999999999999999" is followed by the year "^A100000000000000000000000000000". Similarly, all years from 10 ** 56 to 10 ** 82 get one more caret. So, the year "^Z99999999999999999999999999999999999999999999999999999999" is followed by the year "^^A10000000000000000000000000000000000000000000000000000000000000". This scheme can be extended indefinitely by prepending one addition caret for each additional factor of 10 ** 26 in the range of the year.

In this approach, the number of digits in a date that are used to represent the year is simply:

26 * <number of '^'> + ASCII(<prefix letter>) - ASCII('A') + 5

Note: this algorithm is provided for informational purposes only and to show the path leading to the true solution. Y10K dates MUST NOT use this format. They MUST use the format in the next section.

3.4.2 Space Efficient Approach for Y10**30 Problem

The solution in 3.4.1 is not a space efficient format for giving the number of digits in the year. The length of the prefix grows linearly in the length of the year (which, itself, grows logarithmically over time). Therefore, Y10K format dates use an improved, more compact encoding of the number of digits in the year.

3.4.2.1 Years 10 ** 30 to 10 ** 56

As in 3.4.1, a single '^' and letter precede the year.

3.4.2.2 Years 10 ** 56 to 10 ** 732

The year is preceded by two carets ("^^") and two letters. The letters create a two digit, base 26 number which is the number of digits in the year minus 57. So, the year "^Z99999999999999999999999999999999999999999999999999999" is followed by the year "^^AA100000000000000000000000000000000000000000000000000000". The last representable year with two carets is the year (10 ** 732) - 1 and is "^^ZZ999..999" (i.e. two carets and two Z's, followed by 732 consecutive 9's).

The formula for the number of digits in the year is, based on the two digit prefix is:

```
 26 * (ASCII(<prefix letter1>) - ASCII('A')) +
       ASCII(<prefix letter2>) - ASCII('A') + 57
```

3.4.2.3 Years 10 ** 732 to 10 ** 18308

The next block of years has the number of digits given by three carets ("^^^") followed by three letters forming a three-digit, base 26 number. The number of digits in the year is given by the formula:

```
 676 * (ASCII(<prefix letter1>) - ASCII('A')) +
  26 * (ASCII(<prefix letter2>) - ASCII('A')) +
        ASCII(<prefix letter3>) - ASCII('A') + 733
```

3.4.2.4 General Format for Y10K Dates

In general, if there is at least one letter in a Y10K year, the number of the digits in the year portion of the date is given by the formula:

```
    base26(fib(n) letters) + y10k(n)
```

Where "n" is the number of leading carets and the fig, base26 and y10k functions are defined with the following recurrence relations:

fib(n) is the standard Fibonacci sequence with:

fib(0) = 1

fib(1) = 1

fib(n+2) = fib(n) + fib(n+1)

base26(m letters) is the base 26 number represented by m letters A-Z:

base26(letter) = ASCII(<letter>) - ASCII('A')
base26(string letter) = 26 * base26(string) + base26(letter)

y10k(n) is the necessary fudge factor to align the sequences

properly:

y10k(0) = 5
y10k(n+1) = 26 ** fib(n) + y10k(n)

If the year does not have at least one letter in the year, then the number of digits in the year is:

4

This year format is space-efficient. The length of the prefix giving number of digits in the year only grows logarithmically with the number of digits in the year. And, the number of carets preceding the prefix only grows logarithmically with the number of digits in the prefix.

3.5 B.C.E. (Before Common Era) Years

Now that have a format for all of the years in the future, we'll take on the "negative" years. A negative year is represented in "Y10K-complement" form. A Y10K-complement year is computed as follows:

1) Calculate the non-negative Y10K year string as in 3.4.2.4.
2) Replace all letters by their base 26 complement. I.E. A -> Z, B -> Y, ... Z -> A.
3) Replace all digits in the year portion of the date by their base 10 complement. I.E. 0 -> 9, 1 -> 8, ... 9 -> 0.
4) Replace carets by exclamation points ('!').
5) Four-digit years are pre-pended with a slash ('/')

6) Years that don't now begin with an exclamation point or slash are pre-pended with a star ('*'). (This rule covers the negative 5-31 digit years).

For example, the year 1 BCE is represented by "/9998". The conversion is accomplished by applying rules:

1) Calculate the non-negative Y10K year ("1" -> "0001")
2) Complement the digits ("0001" -> "9998")
3) Four-digit numbers get a leading slash.

The earliest four-digit BCE year (9999 BCE) becomes "/0000" and the year before that (10000 BCE) becomes "*Z89999". The earliest 5-digit BCE year (99999 BCE) is "*Z00000". And the year before that (100000 BCE) is "*Y899999". And so on.

These rules give the desired sort order for BCE dates. For example, the following dates get translated and sorted as:

Jun 6, 200 BCE	/97990606
199 BCE	/9800
Jan 1, 199 BCE	/98000101

3.6 Restrictions on Y10K Dates

There are no restrictions on legal values for Y10K dates. Y10K compliant programs MUST accept any syntactically legal Y10K date as a valid date. A '0' can be appended to the end of any Y10K date, yielding an equivalent date that sorts immediately after the original date and represents the instant after the original date.

The following are all valid representations (in sorted order) of the first instant of A10000:

A1
A10000
A1000001
A100000101000000
A10000010100000000000000000000000

Similarly, the following are all valid Y10K dates (in sorted order) for the time after the last instant of the A99999 and before the first instant of B100000:

A999991231250000
A999991232
A999992
A9999999999
A99999999990000000000000

4 ABNF

The following ABNF [Crocker] gives the formal syntax for Y10K years.

The initial characters definitions are given in their lexical collation (ASCII) order.

```
exclamation = '!'
star        = '*'
slash       = '/'
digit       =  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
letter      =  A | B | C | D | E | F | G | H | I | J | K | L | M |

               N | O | P | Q | R | S | T | U | V | W | X | Y | Z
caret       = '^'


year     = [*(caret | exclamation) | star | slash ] [ *letter ]
           *digit
month    = 2digit
day      = 2digit
hour     = 2digit
minute   = 2digit
second   = 2digit
fraction = *digit
date     = year [ month [ day [ hour [ minute [ second [ fraction
           ]]]]]]
```

5 Open Issues

There are a number date comparison problems that are beyond the scope of this specification.

1) Dates from different calendar systems can not be directly compared. For instance, dates from the Aztec, Bhuddist, Jewish, Muslim, and Hittite calendars must be converted to a common calendar before comparisons are possible.

2) Future re-numberings of years are not covered. If, and when, a new "Year 0" occurs and comes into general use, old dates will have to be adjusted.

3) Continued existence of Earth-centric time periods (year, day, etc.) are problematical past the up-coming destruction of the solar system (5-10 billion years or so). The use of atomic-time helps some since leap seconds are no longer an issue.

4) Future standards and methods of synchronization for inter-planetary and inter-galactic time have not been agreed to.

5) Survivability of dates past the end of the universe is uncertain.

6 Affected Standards

A number of standards currently and RFCs use 4-digit years and are affected by this proposal:

rfc2459: Internet X.509 Public Key Infrastructure Certificate and CRL Profile
rfc2326: Real Time Streaming Protocol (RTSP)
rfc2311: ODETTE File Transfer Protocol
rfc2280: Routing Policy Specification Language (RPSL)
rfc2259: Simple Nomenclator Query Protocol (SNQP)
rfc2244: ACAP -- Application Configuration Access Protocol
rfc2167: Referral Whois (RWhois) Protocol V1.5
rfc2065: Domain Name System Security Extensions
rfc2060: Internet Message Access Protocol - Version 4rev1
rfc1922: Chinese Character Encoding for Internet Messages
rfc1912: Common DNS Operational and Configuration Errors
rfc1903: Textual Conventions for Version 2 of the Simple Network Management Protocol (SNMPv2)
rfc1521: MIME (Multipurpose Internet Mail Extensions) Part One:

rfc1123: Requirements for Internet hosts - application and support

The following standards internally represent years as 16-bit numbers (0..65536) and are affected by this proposal:

rfc2021: Remote Network Monitoring Management Information Base Version 2 using SMIv2
rfc1514: Host Resources MIB

The following ISO standard is affected:
ISO8601: International Date Format

8 Security Considerations

Y10K dates will improve the security of all programs where they are used. Many errors in programs have been tracked to overflow while parsing illegal input. Programs allocating fixed size storage for dates will exhibit errors when presented with larger dates. These errors can be exploited by wily hackers to compromise the security of systems running these programs. Since Y10K dates are arbitrary length strings, there is no way to make them overflow.

In addition, positive Y10K dates are easy to compare and less error-prone for humans. It is easier to compare the three projected end of the universe dates - "H100000000000", "I1000000000000" and "K100000000000000" - by looking at the leading letter than by counting the 0's. This will reduce inadvertent errors by people. This advantage will become more noticeable when large dates are more common.

Unfortunately, negative Y10K dates are a bit more difficult to decipher. However, by comparing the current age of the universe to its projected end, it is obvious that there will be many more positive dates than negative dates. And, while the number of negative dates for human history is currently greater than the number of positive dates, the number of negative dates is fixed and the number of positive dates is unbounded.

9 Conclusion

It is not too early to aggressively pursue solutions for the Y10K problem. This specification presents a simple, elegant, and efficient solution to this problem.

10 References

[Crocker] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, November 1997.

[Drake] Review for the Drake Equation http://www.umsl.edu/~bwilking/assign/drake.html

[Microsoft] SNMP SysUpTime Counter Resets After 49.7 Days http://support.microsoft.com/support/kb/articles/Q169/8/47.asp

[Mike] Y1K http://lonestar.texas.net/~mdlvas/y1k.htm

[Nigel] Nigel's (en)lighening tour of Thermodynamics for Economists ;-) http://www.santafe.edu/~nigel/thermo-primer.html

[NRAO] Astronomical Times http://sadira.gb.nrao.edu/~rfisher/Ephemerides/times.html

[RFC] Here are all the online RFCs. Note: this is a LONG menu. http://info.internet.isi.edu/1s/in-notes/rfc/files

[UNIX] Year 2000 Issues http://www.rdrop.com/users/caf/y2k.html

[Wilborne] PktCDateLig
http://www3.gamewood.net/mew3/pilot/pocketc/pktcdate/index.html

[YUCK] Y10K Unlimited Consulting Knowledgebase
http://www.loyd.net/y10k/index.html

[Zebu] The Search for H0
http://zebu.uoregon.edu/1997/ph410/l6.html

11 Authors' Addresses

Steve Glassman
Compaq Systems Research Center
130 Lytton Avenue
Palo Alto, CA 94301 USA

Phone: +1 650-853-2166
EMail: steveg@pa.dec.com

Mark Manasse
Compaq Systems Research Center
130 Lytton Avenue
Palo Alto, CA 94301 USA

Phone: +1 650-853-2221
EMail: msm@pa.dec.com

Jeff Mogul
Compaq Western Resarch Lab
250 University Avenue
Palo Alto, CA 94301 USA

Phone: +1 650-617-3300
EMail: mogul@pa.dec.com

12. Full Copyright Statement