MWSUG 2017 – Paper PH02

ISO 8601 and SAS®: A Practical Approach

Derek Morgan, PAREXEL International, Billerica, MA

ABSTRACT

The ISO 8601 standard for dates and times has long been adopted by regulatory agencies around the world for clinical data. While there are many homemade solutions for working in this standard, SAS has many built-in solutions, from formats and informats that even take care of time zone specification, to the IS8601_CONVERT routine, which painlessly handles durations and intervals. These built-in capabilities, available in SAS 9.2 and above, will streamline your code and improve efficiency and accuracy. This paper also assumes the use of SAS® version 9.2 and above.

WHAT IS ISO 8601?

It is an internationally accepted methodology to describe dates and times using numbers to facilitate the exchange of data, particularly between international parties. The standard removes the issue of translating month names (23 lip 2017), and defines the order of date and time elements to remove confusion due to local norms. For example, "03-08-16" can be translated as 3 possible dates: March 8, 2016 (assuming the date is in the 21st century); August 16, 2003, or even August 3, 2016, and it depends on the cultural norms of the date's location. The standard has two forms, basic and extended. Basic provides the numbers without delimiters, while extended uses delimiters such as dashes, colons, and periods.

Basic Notation		Extended Notation
2016		2016
201606		2016-06
20160614		2016-06-14
20160614T1422		2016-06-14T14:22
20160614T142237		2016-06-14T14:22:37
20160614T14223774		2016-06-14T14:22:37.74

Table 1: ISO 8601 Basic and Extended Notation

The complete standard (available from http://www.iso.org/iso/home.html) covers the following:

- Date
- Time of day
- Coordinated universal time (UTC)
- Local time with offset to UTC
- Date and time
- Time intervals
- Recurring time intervals

This standard has been adopted by the Clinical Data Interchange Standards Consortium (CDISC), which makes it relevant to the work many of us do every day. This paper will restrict discussion and examples to the extended form which contains delimiters. While both forms are acceptable according to CDISC, it is it much easier to understand the delimited form when viewing ISO dates and times.

There are several ways to represent ISO durations and intervals. Any of these is equally valid, but the one most of us have to calculate is the ISO duration form **P**n**Y**n**M**n**DT**n**H**n**M**n**S**. If the duration is less than one day, this will show as **PT**n**H**n**M**n**S**. If there is no time value, then it will be **P**n**Y**n**M**n**D**. The final duration form is a special case, and is used when the duration is only expressed in weeks: **P**n**W**. The letter **P** stands for "period", and **T** is the ISO 8601 delimiter indicating a time value follows. The

PTn**H**n**M**n**S** form can cause confusion in those unfamiliar with the standard; the "P" has been mistaken for the analysis period, or when followed by T, it has been interpreted as "partial time." Obviously, neither is correct. Some examples (the starting and ending datetimes are shown if you wish to try to replicate the results):

ISO Duration

AE Start Datetime (in ISO 8601 format)	AE End Datetime (in ISO 8601 format)	ISO8601 Duration
2016-08-05T17:19:00	2016-08-06T11:00:00	PT17H41M
2016-08-11T22:30:00	2016-08-14T09:00:00	P2DT10H30M
2016-08-05T09:25:00	2016-08-11T22:15:00	P6DT12H50M
2016-08-23T07:00:00	2016-08-26T07:58:00	P3DT58M
2016-07-29T09:34:00	2016-07-30T07:30:00	PT21H56M

Intervals can be represented in ISO 8601 in two ways: the starting and ending ISO datetimes of the interval, or the starting ISO datetime and an ISO duration, or an ISO duration followed by the ending ISO datetime.

Start/End Datetime Interval Form:

AE Start Datetime (in ISO 8601 format)	AE End Datetime (in ISO 8601 format)	ISO8601 Interval (datetime/datetime form)
2016-08-05T17:19:00	2016-08-06T11:00:00	2016-08-05T17:19/2016-08-06T11:00
2016-08-11T22:30:00	2016-08-14T09:00:00	2016-08-11T22:30/2016-08-14T09:00
2016-08-05T09:25:00	2016-08-11T22:15:00	2016-08-05T09:25/2016-08-11T22:15
2016-08-23T07:00:00	2016-08-26T07:58:00	2016-08-23T07:00/2016-08-26T07:58
2016-07-29T09:34:00	2016-07-30T07:30:00	2016-07-29T09:34/2016-07-30T07:30

Datetime/Duration Interval Form:

AE Start Datetime (in ISO 8601 format)	AE End Datetime (in ISO 8601 format)	ISO8601 Interval (datetime/duration form)
2016-08-05T17:19:00	2016-08-06T11:00:00	2016-08-05T17:19/P0Y0M0DT17H41M0S
2016-08-11T22:30:00	2016-08-14T09:00:00	2016-08-11T22:30/P0Y0M2DT10H30M0S
2016-08-05T09:25:00	2016-08-11T22:15:00	2016-08-05T09:25/P0Y0M6DT12H50M0S
2016-08-23T07:00:00	2016-08-26T07:58:00	2016-08-23T07:00/P0Y0M3DT0H58M0S
2016-07-29T09:34:00	2016-07-30T07:30:00	2016-07-29T09:34/P0Y0M0DT21H56M0S

Duration/Datetime Interval Form:

AE Start Datetime (in ISO 8601 format)	AE End Datetime (in ISO 8601 format)	ISO8601 Interval (duration/datetime form)
2016-08-05T17:19:00	2016-08-06T11:00:00	P0Y0M0DT17H41M0S/2016-08-06T11:00
2016-08-11T22:30:00	2016-08-14T09:00:00	P0Y0M2DT10H30M0S/2016-08-14T09:00
2016-08-05T09:25:00	2016-08-11T22:15:00	P0Y0M6DT12H50M0S/2016-08-11T22:15
2016-08-23T07:00:00	2016-08-26T07:58:00	P0Y0M3DT0H58M0S/2016-08-26T07:58
2016-07-29T09:34:00	2016-07-30T07:30:00	P0Y0M0DT21H56M0S/2016-07-30T07:30

SAS AND ISO 8601

SAS has built-in formats and informats to handle a variety of date and time displays, and the ISO 8601 standard is no exception. SAS will reliably display its date, time, and datetime values to the standard's specifications in both basic and extended forms. Conversely, SAS will translate ISO 8601 dates, times and datetimes into SAS date values. There is one restriction on SAS date, time and datetime values that does not apply to ISO 8601, and that is SAS must have an exact value to store as days since January 1, 1960, seconds since midnight, or seconds since midnight, January 1, 1960. SAS dates, times, and datetimes do not allow for missing components, but it is extremely important to the ISO standard. You might think this is a good reason to build homebrew solutions, but SAS provides its own internal solution, the IS8601_CONVERT routine, which can make some of the standard tasks we have to do much easier.

WHAT ABOUT IMPUTATION?

The SAS ISO 8601 facility allows you to start with a simple premise: read the ISO date or datetime string into a SAS variable with an informat. Just like you do now, if it's missing, then you should perform imputation as specified by the analysis plan.

READING ISO 8601 DATES AND DATETIMES

All you have to remember is E8601DT. for datetimes, and E8601DA. for dates. Most of my programs use some variation of the following code (shown in DATA step syntax, but the INPUT statement is the same in SQL):

```
① xxxdtm = INPUT(xxDTC,E8601DT.); /* Datetime */
② xxxdt = INPUT(xxDTC,E8601DA.); /* Date */
③ xxxtm = TIMEPART(xxxdtm); /* Time */
```

It is important to note that the informats begin with "E8601", which only reads extended notation (delimited) date or datetime strings. You will get an error if you try to read date or datetime values without delimiters ("basic" notation.) I would also advise against using the B8601 variety of informats, because missing month or day will be set to 1, and a missing time will be set to midnight. See what happens when you try to process basic and extended datetime strings with the two informats; this example corresponds to line ① of the above code:

```
DATA test_datetime;
INFILE datalines;
INPUT @1 extended E8601DT. @20 basic B8601DT.;
FORMAT extended basic DATETIME19.;
DATALINES;
2014-03-26T16:14   20140326T1614
2014-06-24         20140624
2414-09            201409
;
RUN;
```

And the result:

Obs	extended	basic
1	26MAR2014:16:14:00	26MAR2014:16:14:00
2		24JUN2014:00:00:00
3		01SEP2014:00:00:00

Here, the extended notation informat only reads the complete string with delimiters (1), while the basic notation informat reads it without delimiters. However, the basic notation also imputes a datetime for the partial basic datetime strings in (2) and (3). This imputation always uses the value 1 for a missing month or day, which may not be what you want.

Now let's run the same example strings with the 8601DA. informats; this example corresponds to line ② of the standard code, `xxxdt = INPUT(xxDTC,E8601DA.);`

```
DATA test_date;
INFILE datalines;
INPUT @1 extended E8601DA. @20 basic B8601DA.;
FORMAT extended basic DATE9.;
DATALINES;
2014-03-26T16:14   20140326T1614
2014-06-24         20140624
2414-09            201409
;
RUN;
```

And the result:

Obs	extended	basic
1	26MAR2014	26MAR2014
2	24JUN2014	24JUN2014
3		01SEP2014

This time, the extended notation informat extracted the date from both the datetime and date string, while leaving the partial date string missing (with an attendant note in the SAS log.) The basic notation informat once again read all three, using the value 1 for the missing day in the last string. This may not be what you want.

So you can extract datetime and date from an ISO 8601 datetime string without parsing first, improving the efficiency of this task. That brings us to line ③ of the standard code:

```
③  xxxtm = TIMEPART(xxxdtm);
```

Since the ISO 8601 datetime and date informats read the strings without parsing, why not use one of the ISO 8601 time informats? The ISO 8601 time informats are strictly designed to translate time strings with or without time zones or offsets into SAS values. To extract the time from a SAS datetime value, it's easier to use the TIMEPART() function on the SAS datetime value you created. No parsing is necessary; you do not have to search for the "T". If the datetime string is incomplete and you've used extended notation, then the SAS datetime value will be missing, and therefore, time will be missing.

From here, you can decide if date or time imputation is necessary by a simple test. If the SAS date or time is missing, then you need to run your imputation process. Three lines of code give you SAS datetime, date, and time values from an ISO 8601 datetime string, and let you know if imputation is necessary.

WRITING ISO 8601 DATES AND DATETIMES

If you're not using a format, you're doing this the hard way. Let's use the example datetime of March 26, 2014, 4:14 P.M. from the previous section.

```
DATA write_datetime;
dt_value = '26MAR2014:16:14'DT;
vsdtc = PUT(dt_value,E8601DT.);
vsdtm = PUT(dt_value,E8601DN.);
vstm = PUT(TIMEPART(dt_value),E8601TM.);
RUN;
```

The result:

dt_value	vsdtc	vsdtm	vstm
1711469640	2014-03-26T16:14:00	2014-03-26	16:14:00

It really is that simple. What if you don't want seconds in your datetime string? On the one hand, if you haven't collected them, they are technically "missing". On the other, it is generally assumed that filling this with zero seconds does not harm the integrity of the data, and it does allow you to create a complete SAS datetime value. The time value is less of an issue, because the TIME5. format is ISO 8601-compliant. What if you are really a stickler? Lop off the seconds with:

```
vsdtc = SUBSTR(PUT(dt_value,E8601DT.),1,16);
```

At the moment, the E8601DT. format does not allow a length less than 19. However, as with all formats and informats, check the most current version of the SAS documentation. New formats and informats are added from time to time, and specifications surrounding existing ones may change.

PARTIAL DATES AND TIMES

We all know SAS stores dates and times in numbers. However, it is assumed that you have an exact date or time; otherwise, SAS cannot accurately store a single value. Clinical data may not be so exact when it comes to dates and times and you may not be able to impute a day, month, or even year for a missing component. Without imputation, your SAS date or time value will be missing. What do you do?

First, let me reiterate that SAS date, time, and datetime values are stored as numeric values. There are NO exceptions. So how does SAS handle a standard for dates and times that explicitly accounts for missing date or time components?

It stores them in character values. However, your data are not stored as the basic or extended notation character strings we are accustomed to. SAS stores them in its own internal representation. What does that look like? Let's run some sample code and see:

```
DATA iso_internal;
LENGTH INDX $ 1;
INFILE datalines PAD MISSOVER;
INPUT raw_iso $ 1-20 @1 extended E8601DT20. @1 iso8601_internal $N8601E20.;
extended_fmt = extended;
iso8601_fmt = iso8601_internal;
FORMAT extended_fmt DATETIME19. iso8601_fmt $N8601EA.;
DATALINES;
2015-03-20T15:05:30
2016-12-22T06:40
2014-01-15
2015-09
2017-02-15T02
2016
;
RUN;
```

	Original --DTC String	SAS Datetime Value	Formatted SAS Datetime Value	SAS ISO 8601 Internal Value	Formatted SAS ISO 8601 Internal Value
A	2015-03-20T15:05:30	1742483130	20MAR2015:15:05:30	2015320150530FFD	2015-03-20T15:05:30
B	2016-12-22T06:40	1798008000	22DEC2016:06:40:00	2016C220640FFFFD	2016-12-22T06:40
C	2014-01-15			2014115FFFFFFFFD	2014-01-15
D	2015-09			20159FFFFFFFFFFD	2015-09
E	2017-02-15T02			201721502FFFFFFD	2017-02-15T02
F	2016			2016FFFFFFFFFFFD	2016

This table demonstrates how SAS stores ISO datetime values internally. The "Original --DTC string" column is a sample of what we might see in a typical AEDTC file. As you can see, lines A and B are the only complete ISO 8601 datetime strings. Therefore, they are the only rows that can be stored as SAS datetime values. Lines C through F do not have enough information to pin down an exact time on an exact day, so the SAS datetime value is missing. The "SAS ISO 8601 Internal Value" column shows you what SAS has stored in the variable ISO8601_INTERNAL. Finally, the last column shows that you get the original –DTC string when you format the stored ISO 8601 value.

Since the first column and the last column are the same, why would you want to waste the processing to store a –DTC string in an internal representation that needs to be formatted to make any sense?

WELCOME TO THE IS8601_CONVERT ROUTINE

This routine does many things with ISO 8601 dates, including creating a SAS ISO 8601 value from individual date and time components (up to 6; year, month, day, hours, minutes, seconds.) The complete syntax for the IS8601_CONVERT routine is:

CALL IS8601_CONVERT(convert-from, convert-to, <from-variables>, <to-variables>, <date-time-replacements>);

One of the more convenient aspects of this routine is the ability to include simple imputations inside the routine itself. You can specify these replacements by individual component, that is, you must specify a value to use for each of a missing year, month, day, hour, minute, or second. The default value for year is missing; 1 for month and day, while hour, minute and second are imputed as 0.

Replacements are separated by commas in the order *year, month, day, hour, minute, seconds*, and you must provide a value or a comma as a placeholder if you are not going to change the default replacement. While this may be convenient, simple imputation may not be appropriate for your analysis purposes. Additionally, automatic imputation only works if one of the values you are working with is incomplete. When both values have missing components, the imputation is the same for both, so there is no effect on the result. Remember that calculations are always done with the default replacement values for missing date and/or time components.

Why should you make the effort? Let's take a look at ISO durations, most frequently seen in the --DUR variables from SDTM or added as an analysis variable in ADaM.

This little trick simplifies calculating and creating an ISO 8601 duration string beginning with the letter "P". Let's take the simple case where we have a complete datetime value. For this example, time is set to midnight of the given day. My recommendation would be to use this technique after you have performed any necessary imputation, so you do not have any partial dates. Note that the variables AESTDTM and AEENDTM are SAS datetime values, and the duration AEDUR is a character variable (line 3.)

```
DATA duration1;
SET full_datetimes;
LENGTH aedur aedur_unf $ 32; /* Important! If the result variable is
                                numeric, AEDUR will be in seconds, i.e.,
                                aeendtm – aestdtm */
CALL IS8601_CONVERT('dt/dt','du',aestdtm,aeendtm,aedur);
aedur_unf = aedur;
FORMAT aestdtm aeendtm datetime19. aedur $N8601E.;
RUN;
```

The result below shows the event durations calculated above.

indx	Start Date/Time of Adverse Event	End Date/Time of Adverse Event	Duration of Adverse Event	Unformatted Duration/of Adverse Event
A	20MAR2015:00:00:00	20MAR2015:00:00:00	P0W	000000000000000E
B	22DEC2016:00:00:00	23DEC2016:00:00:00	P1D	FFFFF01FFFFFFFFC
C	15JAN2014:00:00:00	04JUN2014:00:00:00	P4M20D	FFFF420FFFFFFFFC
D	25FEB2013:00:00:00	16MAY2017:00:00:00	P4Y2M19D	0004219FFFFFFFFC

Well, that was easy. A few things to note here: first, notice that the start and end date/time columns are right-justified. They are numeric values (seconds since midnight, January 1, 1960,) and by default, the

DATETIME. format is right-justified. Also, these are ISO 8601 durations, not analysis durations. We add 1 to the duration for analysis in many situations, so that events have a minimum duration of one day. If you want to represent that analysis duration as an ISO 8601 duration, adjust the end date and function call as appropriate. The code below only modifies the ending date if it is the same as the start date by adding a day's worth of seconds to the ending SAS datetime value. It's easy enough to remove the conditional nature of the adjustment.

```
IF aestdtm EQ aeendtm THEN
   tmp_aeendtm = aeendtm + 86400;
ELSE
   tmp_aeendtm = aeendtm;
CALL IS8601_CONVERT('dt/dt','du',aestdtm,tmp_aeendtm,aedur);
```

What if you have partial dates? I would prefer to perform the imputation as specified in the statistical analysis plan, create full datetime values, and use the above method. However, you can take advantage of the way SAS stores and handles ISO 8601 dates. If you choose this path, SAS will use a default substitution for missing datetime components; alternately, you can choose to define the values you wish to substitute for each component in the routine itself. Let's take our dates and create SAS ISO 8601 values by using the $N8601E. informat. Notice that AESTDT, AEENDT are character variables this time. This differs from the previous example because there are incomplete dates. If you try to read them as SAS datetime values, you will get missing values and missing durations as a result. You can only read partial dates as ISO 8601 dates, and SAS will do the imputation behind the scenes before calculating the duration result. Once again, note that AESTDT, AEENDT and AEDUR are all character variables.

```
DATA duration2;
INFILE datalines PAD MISSOVER;
LENGTH aestdt aestdt_unf aeendt aeendt_unf aedur aedur_unf $ 32;
INPUT @1 aestdt :$N8601E. @12 aeendt :$N8601E.;
indx = BYTE(64+_n_);
CALL IS8601_CONVERT('dt/dt','du',aestdt,aeendt,aedur);
aestdt_unf = aestdt;
aeendt_unf = aeendt;
aedur_unf = aedur;
FORMAT aestdt aeendt aedur $N8601E.;
DATALINES;
2015-03-20
2016-12-22 2016-12-23
2014-01    2014-06
2013       2017-01
2012-05    2016
;
RUN;
```

And the result:

	Adverse Event Start Date/Time	**Unformatted Start Date/Time of Adverse Event**	**Adverse Event End Date/Time**	**Unformatted Adverse Event End Date/Time**	**Adverse Event Duration**	**Unformatted Duration of Adverse Event**
A	2015-03-20	2015320FFFFFFFFD	****************			FFFFFFFFFFFFFFFF
B	2016-12-22	2016C22FFFFFFFFD	2016-12-23	2016C23FFFFFFFFD	P1D	FFFFF01FFFFFFFFC
C	2014-01	20141FFFFFFFFFFD	2014-06	20146FFFFFFFFFFD	P5M	FFFF5FFFFFFFFFFC
D	2013	2013FFFFFFFFFFFD	2017-01	20171FFFFFFFFFFD	P4Y1M	00041FFFFFFFFFFC
E	2012-05	20125FFFFFFFFFFD	2016	2016FFFFFFFFFFFD	P3Y7M	00037FFFFFFFFFFC

The columns are shown with and without formatting. As you can see, the values in AESTDT and AEENDT are stored in SAS ISO 8601 format, and they are character values. I cannot stress this enough, because we're used to dealing with dates and times as numbers in SAS. Another difference from regular SAS dates and times is that all three variables use the $N8601E. format, but they display differently. When you use this format, SAS determines the context from the values and chooses the appropriate display.

Another possibility is to convert ISO durations into their SAS time/datetime values:

```
DATA duration_convert;
INFILE datalines PAD MISSOVER;
LENGTH aedur $ 32 hours 8;
INPUT @1 aedur :$N8601E32.;
indx = BYTE(64+_n_);
CALL IS8601_CONVERT('du','du',aedur,hours);
aedur_unf = aedur;
hours_unf = hours;
FORMAT aedur $N8601E16. hours time8.;
DATALINES;
P1D
P1M2D
PT14H23M
P3M12DT6H30M
;
RUN;
```

	Duration of Adverse Event	Unformatted Duration of Adverse Event	Duration of Adverse Event (h)	Duration of Adverse Event (sec)
A	P1D	FFFFF01FFFFFFFFC	24:00:00	86400
B	P1M2D	FFFF102FFFFFFFFC	768:00	2764800
C	PT14H23M	FFFFFFF1423FFFFC	14:23:00	51780
D	P3M12DT6H30M	FFFF3120630FFFFC	2454:30	8836200

As you can see, the ISO duration is now a SAS time value, without having to derive a start or end date. If you have an ISO start date and an ISO duration, you can also calculate the end datetime using this routine:

```
DATA calc_enddate;
INFILE datalines PAD MISSOVER;
LENGTH aestdt $ 32 aedur $ 32 enddate 8;
INPUT @1 aestdt :$N8601E. @12 aedur :$N8601E.;
indx = BYTE(64+_n_);
CALL IS8601_CONVERT('dt/du','end',aestdt,aedur,enddate);
aestdt_unf = aestdt;
aedur_unf = aedur;
enddate_unf = enddate;
FORMAT aestdt aedur $N8601E16. enddate dtdate9.;
DATALINES;
2016-04-05 P6D
2017-02-15 P1M11D
2016-11-17 P3W
2014-08-22 P3M12D
;
RUN;
```

Start Date of Adverse Event	Unformatted Start Date of Adverse Event	Duration of Adverse Event	Unformatted Duration of Adverse Event	Calculated End Date of Adverse Event	Unformatted Calculated End Date of Adverse Event
2016-04-05	2016405FFFFFFFFD	P6D	FFFFF06FFFFFFFFC	11APR2016	1775952000
2017-02-15	2017215FFFFFFFFD	P1M11D	FFFF111FFFFFFFFC	26MAR2017	1806105600
2016-11-17	2016B17FFFFFFFFD	P3W	000000000000030E	08DEC2016	1796774400
2014-08-22	2014822FFFFFFFFD	P3M12D	FFFF312FFFFFFFFC	04DEC2014	1733270400

When you look at the last column, you will see a large value; this is a SAS datetime value, so it's seconds since midnight, January 1, 1960. The DTDATE. format is used to display it as a date, but it is a complete SAS datetime value.

These are some of the tasks that can easily be automated with the IS8601_CONVERT routine. For more details and possibilities, consult the SAS documentation.

SUMMARY

ISO 8601 has been adopted as the standard for dates and times in clinical studies around the world, which makes it of interest to clinical SAS programmers. It provides for missing components in dates, times, and datetimes. This is incompatible with the way we think about standard SAS dates, times and datetimes, which need to point to an exact moment so an exact value can be assigned. Without this exact value, SAS will leave it as missing, possibly leading to a loss of information.

Converting ISO 8601 dates and datetimes to SAS requires using the E8601 series of informats. Similarly, to display a SAS date or datetime value, you can use the E8601 series of formats. If you need to read/write time on its own, the TIME5. format and informat will suffice. The ISO 8601 standard for times includes time zone information, which is often unnecessary or unavailable in our clinical programming environment.

Even though normal date, time and datetime handling in SAS requires an exact value, there is a fully-compliant ISO 8601 facility in SAS, and it revolves around the IS8601_CONVERT routine. This is a call with no returned value. Values read are stored in character variables with an internal format, much like SAS does with standard dates, times, and datetimes. As with standard SAS dates, times and datetime, these ISO values need formats to understand them, and informats to convert ISO strings to SAS values.

The IS8601_CONVERT routine is capable of much more than creating ISO values in SAS. Calculating durations, and displaying them in ISO8601-compliant form is a prime example, replacing the need to break down a SAS datetime value in seconds into years, months, days, hours, minutes and seconds, and then assemble the ISO duration with concatenation.

This information and these examples should help improve your efficiency in dealing with the ISO 8601 standard in your day-to-day programming.

REFERENCES

Morgan, Derek P. 2014. *The Essential Guide to SAS® Dates and Times, Second Edition*. Cary, NC: SAS Institute Inc.

CONTACT INFORMATION:

Further inquiries are welcome to:

Derek Morgan
E-mail: mrdatesandtimes@gmail.com