MWSUG - Paper PL-017

# Have a Date with ISO®? Using PROC FCMP to Convert Dates to ISO 8601

Richann Jean Watson, DataRich Consulting

## ABSTRACT

Programmers frequently have to deal with dates and date formats. At times, determining whether a date is in a day-month or month-day format can leave us confounded. Clinical Data Interchange Standards Consortium (CDISC) has implemented the use of the International Organization for Standardization (ISO) format, ISO® 8601, for datetimes in SDTM domains, to alleviate the confusion. However, converting "datetimes" from the raw data source to the ISO 8601 format is no picnic. While SAS® has many different functions and CALL subroutines, there is not a single magic function to take raw datetimes and convert them to ISO 8601. Fortunately, SAS allows us to create our own custom functions and subroutines. This paper illustrates the process of building a custom function with custom subroutines that takes raw datetimes in various states of completeness and converts them to the proper ISO 8601 format.

## INTRODUCTION

The goal is to convert the varying dates or datetimes in data sets which could originate from many different countries in the world into the proper ISO 8601 format. CDISC Study Data Tabulation Model (SDTM) standards ensure that all dates and datetimes are represented in a standard format, utilizing ISO 8601. This standard format is a way to represent the dates and times that is consistent regardless of where you are in the world. (ISO 8601 Date and Time Format, n.d.)

## WHAT IS ISO 8601 FORMAT?

It is easy to misread a date, especially if it is in the form of mm/dd/yyyy or dd/mm/yyyy. For example, the date field 07/09/2021 could have different meanings depending on the format This date field could represent either July 9, 2021, or September 7, 2021. The ISO 8601 format is a standard way of representing dates and times so that they are correctly interpreted across the globe. The internationally agreed way to represent dates is YYYY-MM-DD (ISO 8601 Date and Time Format, n.d.). Therefore, the date July 9, 2021, is represented as 2021-07-09 in ISO 8601 format.

However, there is more to the ISO 8601 format than just having a standard way to represent a date. The format specification also encompasses how the time component should be incorporated into datetimes. The standard format for a datetime is YYYY-MM-DDTHH:MM:SS, where:

- YYYY is the four-digit year
- MM (first one in the format) is the two-digit month, 01 - 12
- DD is the two-digit day, 01 - 28, 29, 30 or 31 depending on the month and year
- T indicates that a time value follows
- HH is the two-digit hour, represented in 24-hour clock, 00 - 23
- MM (second one in the format) is the two-digit minutes, 00 - 59
- SS is the two-digit seconds, 00 - 59

Additionally, the ISO 8601 format indicates how to handle missing components. Each missing component is represented by a single hyphen; but only components up to the last non-missing component are displayed. For example, a start date for a medication is required, but a patient does not recall the month they actually started taking the medication. A patient may only recall the year, day, and hour they started taking the medication because it has been taken at the same time every month. The estimated start datetime is 12 pm on the 9th in the year 2021. In this scenario, the month, minutes, and seconds are missing. This estimated date would be represented as 2021-**-**09T12 in the ISO 8601 format. Notice a hyphen is used to represent the missing month. Convention dictates that hyphens are used to represent the missing minutes and seconds. Since the hour a medication was taken is the last non-missing component, the datetime is represented up to the hour's component. Hyphens for the subsequent missing minutes and seconds are left off, implying that minutes and seconds are missing.

## SAMPLE DATA SET FOR PAPER

Now that you have an understanding what the ISO 8601 format is, the raw datetimes shown in Data Display 1 need to be converted. Notice that in the sample data, there is a smattering of anomalies, and no consistency in how the dates are recorded. However, they all need to be converted into the ISO 8601 format.

| DAT |
|---|
| 09JUL2021 |
| 09JUL2021T12:15 |
| 09JUL2021:12:15 |
| UNJUL2021 |
| UNUNK2021 |
| 2021-UN-UN |
| 2021-07-09 |
| 2021-07-09 12:15 |
| 2021-07-09 12:15:45 |
| 2021-07-09:12:15:45 |
| 2021-07-09 UN:15 |
| 2021-07-09T9:15 |
| 2021-07-09T  :15 |
| 09JUL2021 9:15 |
| 09JUL2021   :15 |
| 09JUL2021 T9:15 |
| UNJUL2021 TUN:UN |
| UNJUL2021TUN:UN |
| 2021-UN-09TUN:15 |
| 2021-UN-UNTUN:UN |
| 20210709:12:15 |
| 20210709 12:15 |
| 20210709T12:15 |

| DAT |
|---|
| 2021UNUN 12:15 |
| 202107UNTUN:UN |
| 2021UN09 UN:UN |
| 20210709 UN:UN |
| 2021UNUN UN:UN |
| 2021UNUNTUN:UN |
| 2021-07-09 12:15:45 |
| 2021-07-09 12:15:45 |
| 2021-07-09T12:15:45 |
| 2021-07-09:12:15:45 |
| 09JUL2021T12:15 AM |
| 09JUL2021:12:15 PM |
| 09JUL2021:2:15 PM |
| 19-JUL-2021:2:15 PM |
| UN-UNK-2021T3:25 PM |
| UN-MOR-2021 T5:25 PM |
| 0729 |
| 0715 T5 |
| 07/08 |
| 07/19 5:12 |
| T13:14 |
| JUL 9, 2021:12:15 |

*Data Display 1: Raw dates from various data sources captured in one field*

## ISO 8601 DATETIMES AND SAS

SAS has myriad formats and informats to help us to display or read dates and durations utilizing the ISO 8601 basic (without delimiters) and extended (with delimiters) formats. For example, E8601DT*w.d* is a SAS format (and informat) that will write the numeric datetime in the appropriate ISO 8601 extended format. While B8601DN*w*. writes (or reads) and the ISO 8601 datetime in the basic ISO 8601 notation. (SAS Institute Inc., 2022)

Consider the example shown in Data Display 2. The dates are stored as character data type and are in the ISO 8601 notation format. It is difficult to apply date calculations to these variables without some form of transformation.

| DAT |
|---|
| 2021-07-09 |
| 2021-07-09T12:15 |

*Data Display 2: Character datetimes in ISO 8601 notation*

SAS Program 1 illustrates how the B8601DN. and E8601DT. informats can be used to read in the character ISO 8601 datetime notation. In addition, it shows how the same B8601DN. and E8601DT. informats can be used to format the numeric datetimes to display in the basic (B8601DN.) or extended (E8601DT.) notation but still be of numeric data type. The results from SAS Program 1 can be seen in Data Display 3, which shows both the formatted numeric variables (DT and DTTM) as well as the unformatted numeric variables (NF_DT and NF_DTTM). The unformatted variables show the number of seconds from January 1, 1960 (SAS Institute Inc., 2019).

```
Data isodt2;
   set isodt;
   format DT b8601dn. DTTM e8601dt.;
   DT = input(DAT, b8601dn.);
   DTTM = input(DAT, e8601dt.);

   NF_DT = input(DAT, b8601dn.);
   NF_DTTM = input(DAT, e8601dt.);
run;
```

*SAS Program 1: Read character datetime in ISO 8601 notation to numeric datetime*

| DAT | DT | DTTM | NF_DT | NF_DTTM |
|---|---|---|---|---|
| 2021-07-09 | 20210709 | . | 1941408000 | . |
| 2021-07-09T12:15 | 20210709 | 2021-07-09T12:15:00 | 1941408000 | 1941452100 |

*Data Display 3: Character datetimes in ISO 8601 notation converted to numeric datetime*

In addition, SAS has a CALL subroutine, CALL IS8601_CONVERT, which converts the ISO 8601 intervals to datetime and duration values or vice versa. But this requires the datetime to already be in the ISO 8601 format.

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

SAS Program 2 demonstrates how the IS8601_CONVERT subroutine can be used. The dates can be either numeric or character as shown with the use of DTTM (numeric) and DAT (character) in the two separate calls. With the use of a character date, it allows for simple substitution, not a true imputation, as seen with the population of DTDUR2 in Data Display 4 (Morgan D. , 2017). Note that for the example the default substitution values are used, but user-specified values can be set. The unformatted values for DTDUR and DTDUR2 (NF_DTDUR and NF_DTDUR2, respectively) are stored as character variables and are the SAS internal representation of the ISO date (Morgan D. , 2017). For more details on CALL IS8601_CONVERT, visit SAS Documentation on CALL IS8601_CONVERT. (SAS Institute Inc., 2022)

```
data isodt3;
   set isodt2 (drop = NF_:);
   format TRTSDT e8601da. TRTSDTM e8601dt. DTDUR DTDUR2 $N8601E.;
   TRTSDT = '08MAR2021'd;
   TRTSDTM = dhms(TRTSDT, 12, 13, 4);

   call IS8601_CONVERT('dt/dt', 'du', TRTSDTM, DTTM, DTDUR);
   call IS8601_CONVERT('dt/dt', 'du', TRTSDTM, DAT, DTDUR2);

   NF_DTDUR = DTDUR;
   NF_DTDUR2 = DTDUR2;
run;
```

*SAS Program 2: Illustration of CALL IS8601_CONVERT subroutine*

| DAT | DT | DTTM | TRTSDT | TRTSDTM |
|---|---|---|---|---|
| 2021-07-09 | 20210709 | . | 2021-03-08 | 2021-03-08T12:13:04 |
| 2021-07-09T12:15 | 20210709 | 2021-07-09T12:15:00 | 2021-03-08 | 2021-03-08T12:13:04 |

| DTDUR | DTDUR2 | NF_DTDUR | NF_DTDUR2 |
|---|---|---|---|
| | P4MT11H46M56S | FFFFFFFFFFFFFFFF | FFFF4FF114656FFC |
| P4M1DT1M56S | P4M1DT1M56S | FFFF401FF0156FFC | FFFF401FF0156FFC |

*Data Display 4: Illustration of CALL IS8601_CONVERT Subroutine*

However, the purpose of this paper is not to focus on the tools SAS has made available, but rather how to create and use user-defined subroutines and functions to overcome missing functionality.

## PROC FCMP BASICS

As noted, there is no SAS-provided function which will convert the values in Data Display 1 to the ISO 8601 format, so a user-defined function and subroutines must be created to handle this. The FCMP procedure (PROC FCMP) provides the means by which user-defined functions and subroutines can be created. There are a number of options and statements that can be used in PROC FCMP, but this paper only focuses on a subset of this powerful procedure's potential utility. The general syntax used to create the custom subroutines and functions in this paper is shown in SAS Program 3.

```
PROC FCMP outlib = libname.dataset.package  <options>;
   SUBROUTINE subroutine-name (argument-1 <, argument-2, …>);
      OUTARGS out- argument-1 <, out-argument-2, …>;
      <<< SAS STATEMENTS >>>
   ENDSUB;
   FUNCTION function-name (argument-1 <, argument-2, …>) <$>;
      OUTARGS out- argument-1 <, out-argument-2, …>;
      <<< SAS STATEMENTS >>>
      RETURN(expression)
   ENDFUNC;
QUIT;
```

*SAS Program 3: PROC FCMP general syntax*

On the PROC FCMP statement, you can specify several different options. A required option is OUTLIB =. OUTLIB specifies a three-level name of an output data set where the compiled functions and subroutines are written when PROC FCMP hits a step boundary (i.e., QUIT statement). An example of this option is `outlib = fcmp.funcs.ISO_date`.

Both functions and subroutines must be defined, and these definitions begin with a declaration statement and conclude with a termination statement. A user-defined function is declared with the FUNCTION statement and is terminated with the ENDFUNC statement, and a user-defined subroutine is declared with the SUBROUTINE statement and is terminated with the ENDSUB statement. In reality, however, the ENDFUNC and ENDSUB statements are interchangeable and either can be used to terminate a function or subroutine.

In the function and subroutine statements, a name (function-name or subroutine-name) should be provided, and for most functions and subroutines at least one argument (argument-1, argument-2) should also be provided. If the arguments are character, then they should have a dollar sign after the argument name, for example, `subroutine zfill(_comp $).`

Both subroutine and function statements allow for use of general SAS language statements. A difference between a subroutine and function is that a function will return one value. The RETURN statement is used to do this. A function's definition can contain multiple RETURN statements; however, only one RETURN statement will be executed during a function call, so only one value will be returned. If the return value is a character value, then the function statement needs a dollar sign at the end, as seen in the

example `function ISO_dttm(dattim $) $`. The subroutine does not return a value and therefore the RETURN statement is not included. However, both the function and subroutine statements allow for multiple arguments to be modified within the function or subroutine, and they use OUTARGS to indicate which arguments are to be changed in the calling program. (Hughes, 2022)

In order to use the custom subroutines and functions in a DATA step, stored subroutines and functions need to be accessed via the use of the CMPLIB = option. In the CMPLIB = option, a two-level name is specified indicating where the subroutines and functions are stored. The two-level name must match the first two levels of the compiled subroutines and functions specified in the OUTLIB = option on the PROC FCMP statement.

## CUSTOM ISO 8601 CONVERSION FUNCTION

It is common to receive unclean data that is rarely properly formatted. With the basics of PROC FCMP in your programming toolkit, custom function and subroutines can be constructed to help convert raw date/times into the appropriate ISO 8601 format required by CDISC.

### BUILDING ISO 8601 CUSTOM SUBROUTINES AND FUNCTION

Recall that the proper ISO 8601 notation has a four-digit year and two digits for months, days, hours, minutes and seconds, and any missing component is represented by a single hyphen, '-'. Data can come in a variety of formats as illustrated in Data Display 1, and requires cleaning to conform to standards. Two subroutines can be constructed to do this cleaning. The first subroutine, ZFILL, checks to ensure that all non-missing date and time components are two digits. If a date or time component is missing, then it is represented with a single hyphen (SAS Program 4).

```
subroutine zfill(_comp $);
   outargs _comp;
   if not missing(_comp) and not( notdigit(cats(_comp)) )
       then _comp  = put(input(_comp, best.), Z2.);
   else _comp = '-';
endsub;
```

*SAS Program 4: Subroutine to zero fill non-missing two-digit date and time components*

The second subroutine (DTTMFMT, SAS Program 5) handles the year to make sure it is a four-digit value if it is non-missing. If the year is a two-digit year, it is converted to four digits using 40 as the cutoff. If the year is missing, it is represented by a single hyphen. In addition, this subroutine has a call to the first subroutine created to ensure the other date and time components are formatted correctly.

```
subroutine dttmfmt(_year $, _month $, _day $,
                   _hour $, _minute $, _second $);
   outargs _year, _month, _day, _hour, _minute, _second;

   /* SAS statements to handle two-digit years or missing years */

   /* need to 0 fill each non-missing month, day, hour, minute, second */
   call zfill(_month);
   call zfill(_day);
   call zfill(_hour);
   call zfill(_minute);
   call zfill(_second);
endsub;
```

*SAS Program 5: Subroutine to format non-missing date and time components to correct number of digits*

Now that you have two subroutines to help with the data cleaning (or as much of the data cleaning as possible), you can now work to build a user-defined function (ISO_DTTM, SAS Program 6). ISO_DTTM takes one argument (DATTIM) which is the datetime in its raw format. The function first extracts the

portion that is assumed to be the date by looking for a delimiter (i.e., a space, a 'T' or a colon). In addition, it compresses any hyphens that may be in the data. Further processing of the date component takes place in order to break the date into individual components so that each component can be checked for a non-missing value and formatted accordingly. In addition, processing of the portion that is assumed to be the time component takes place to break the time into its individual components. Once the DATTIM is parsed to the individual pieces, the subroutine DTTMFMT is called to format each datetime part. Recall DTTMFMT subroutine has calls to the ZFILL subroutine in order to zero fill any non-missing portion that is to be a two-digit value. Now that all the individual components have been formatted per the requirements, they can all be combined. The last bit of code in the function is to check for the last non-missing component since any missing components after the last non-missing component will be left null and not have the single hyphen. The final datetime in the proper ISO 8601 notation is then returned using the RETURN statement.

For the full code refer to the Appendix A.

```
function ISO_DTTM(dattim $) $;
   length __dtc __dttm $20 iso_dtc $10 iso_tmc $8 year $4 month $2 day $2;

   /* extract the date portion; compress hyphens to determine format */
   __dtpart = compress(scan(dattim, 1, ' T:', 'm'), '-');
   if length(strip(__dtpart)) in (7 9) then do;
      /* SAS code to determine date format and extract date components */
   end;
   /* More SAS code to determine date formats and extract date components */
   else do;
      /* PUT statements to write to log if invalid/insufficient info */
      __dtc = '_ERROR_';
      return(__dtc);
   end;

   /* SAS code to extract time components */

   call dttmfmt(year, month, day, hour, minute, second);

   /* use new variables to build ISO 8601 dates in the proper format */
   iso_dtc = catx('-', year, month, day);
   iso_tmc = catx(':', hour, minute, second);

   /* SAS code to see if there is a time component */
   /* SAS code to create __dtc */

   return(__dtc);
endfunc;
```

*SAS Program 6: User-defined function to convert raw date/times to proper ISO 8601 notation*

## COMPILED SUBROUTINES AND FUNCTIONS

After you have created the subroutines and functions, the information is stored in the indicated data set as a package (i.e., the three-level output name specified on OUTLIB) when the PROC FCMP ends (i.e., reaches a QUIT statement). Note that if OUTLIB is not specified then the compiled subroutines and function will not be saved.

Data Display 5 shows a snippet of the data set that is created once the subroutines and function are compiled. The data set name is FUNCS which is saved in the FCMP library. The package name is

ISO_DATE. The ISO_DATE package is just a collection of the subroutines and function defined.

| Obs | _Key_ | Owner | Sequence | Type | Subtype | Name | Continue | NValue | Encoded | Value |
|---|---|---|---|---|---|---|---|---|---|---|
| 5 | F.ISO_DATE.DTTMFMT | CMP | 112 | Statement Source | Executable | SUBROUTI | 0 | 65 | . | subroutine dttmfmt(_year $, _month $, _day $, _hour $, _minute $, _second $); |
| 6 | F.ISO_DATE.DTTMFMT | CMP | 113 | Statement Source | Executable | OUTARGS | 0 | 14 | . | outargs _year, _month, _day, _hour, _minute, _second; |
| 7 | F.ISO_DATE.DTTMFMT | CMP | 114 | Statement Source | Comment | CMT | 0 | 101 | . | /* make sure year is a four-digit number */ |
| 8 | F.ISO_DATE.DTTMFMT | CMP | 115 | Statement Source | Executable | IF | 0 | 2 | . | if not( notdigit(cats(_year)) ) then |
| 9 | F.ISO_DATE.DTTMFMT | CMP | 116 | Statement Source | Executable | DO | 0 | 3 | . | do; |
| 10 | F.ISO_DATE.DTTMFMT | CMP | 117 | Statement Source | Executable | IF | 0 | 2 | . | if length(strip(_year)) = 4 then |
| 11 | F.ISO_DATE.DTTMFMT | CMP | 118 | Statement Source | Executable | ASSIGN | 0 | 1 | . | _year = strip(_year); |
| 12 | F.ISO_DATE.DTTMFMT | CMP | 119 | Statement Source | Executable | ELSE | 0 | 9 | . | else |
| 13 | F.ISO_DATE.DTTMFMT | CMP | 120 | Statement Source | Executable | IF | 0 | 2 | . | if length(strip(_year)) = 2 then |
| 14 | F.ISO_DATE.DTTMFMT | CMP | 121 | Statement Source | Executable | DO | 0 | 3 | . | do; |
| 15 | F.ISO_DATE.DTTMFMT | CMP | 122 | Statement Source | Executable | IF | 0 | 2 | . | if input(_year, best.) <= 40 then |
| 16 | F.ISO_DATE.DTTMFMT | CMP | 123 | Statement Source | Executable | ASSIGN | 0 | 1 | . | _year = cats('20', _year); |
| 17 | F.ISO_DATE.DTTMFMT | CMP | 124 | Statement Source | Executable | ELSE | 0 | 9 | . | else |
| 18 | F.ISO_DATE.DTTMFMT | CMP | 125 | Statement Source | Executable | ASSIGN | 0 | 1 | . | _year = cats('19', _year); |
| 19 | F.ISO_DATE.DTTMFMT | CMP | 126 | Statement Source | Executable | END | 0 | 10 | . | end; |
| 20 | F.ISO_DATE.DTTMFMT | CMP | 127 | Statement Source | Executable | END | 0 | 10 | . | end; |
| 21 | F.ISO_DATE.DTTMFMT | CMP | 128 | Statement Source | Executable | ELSE | 0 | 9 | . | else |
| 22 | F.ISO_DATE.DTTMFMT | CMP | 129 | Statement Source | Executable | ASSIGN | 0 | 1 | . | _year = '-'; |
| 23 | F.ISO_DATE.DTTMFMT | CMP | 130 | Statement Source | Comment | CMT | 0 | 101 | . | /* need to zero fill each non-missing month, day, hour, minute, second */ |
| 24 | F.ISO_DATE.DTTMFMT | CMP | 131 | Statement Source | Executable | CALL | 0 | 4 | . | call zfill(_month); |
| 25 | F.ISO_DATE.DTTMFMT | CMP | 132 | Statement Source | Executable | CALL | 0 | 4 | . | call zfill(_day); |
| 26 | F.ISO_DATE.DTTMFMT | CMP | 133 | Statement Source | Executable | CALL | 0 | 4 | . | call zfill(_hour); |
| 27 | F.ISO_DATE.DTTMFMT | CMP | 134 | Statement Source | Executable | CALL | 0 | 4 | . | call zfill(_minute); |
| 28 | F.ISO_DATE.DTTMFMT | CMP | 135 | Statement Source | Executable | CALL | 0 | 4 | . | call zfill(_second); |
| 29 | F.ISO_DATE.DTTMFMT | CMP | 136 | Statement Source | Executable | ENDSUB | 0 | 14 | . | endsub; |
| 149 | ISO_DATE | CMP | 0 | Header | Package | | 0 | . | . | <L n="Header"><S n="Version"><![CDATA[1.1]]></S><N n="Datetime"> 1969968854.286</N><S n="DatetimeStr"><![CDATA[04JUN22:13:34:14]]></S><S n="SubType"><![CDATA[Package]]></S><N n="Obfuscate">0</N></L> |

*Data Display 5: Snippet of data set of compiled user-defined subroutines and function*

## USING CUSTOM SUBROUTINES AND FUNCTIONS

The user-defined subroutines and functions have been defined, compiled, and saved for future use. In order to use them, you need to specify the CMPLIB option to point to the data set where the compiled subroutines and function are saved. Once you have specified this option, then the subroutines and functions could be used like any other SAS subroutine or function as shown in SAS Program 7. The results from SAS Program 7 are shown in Data Display 6.

```
options cmplib = fcmp.funcs;

data outdsname;
   set indsname;
   DTC = ISO_DTTM(DAT);
run;
```

*SAS Program 7: Using ISO_DTTM with a datetime captured in one field*

| DAT | DTC |
|---|---|
| 09JUL2021 | 2021-07-09 |
| 09JUL2021T12:15 | 2021-07-09T12:15 |
| 09JUL2021:12:15 | 2021-07-09T12:15 |
| UNJUL2021 | 2021-07 |
| UNUNK2021 | 2021 |
| 2021-UN-UN | 2021 |
| 2021-07-09 | 2021-07-09 |
| 2021-07-09 12:15 | 2021-07-09T12:15 |
| 2021-07-09 12:15:45 | 2021-07-09T12:15:45 |
| 2021-07-09:12:15:45 | 2021-07-09T12:15:45 |
| 2021-07-09 UN:15 | 2021-07-09T-:15 |
| 2021-07-09T9:15 | 2021-07-09T09:15 |

| DAT | DTC |
|---|---|
| 2021UNUN 12:15 | 2021----T12:15 |
| 202107UNTUN:UN | 2021-07 |
| 2021UN09 UN:UN | 2021---09 |
| 20210709 UN:UN | 2021-07-09 |
| 2021UNUN UN:UN | 2021 |
| 2021UNUNTUN:UN | 2021 |
| 2021-07-09 12:15:45 | 2021-07-09T12:15:45 |
| 2021-07-09 12:15:45 | 2021-07-09T12:15:45 |
| 2021-07-09T12:15:45 | 2021-07-09T12:15:45 |
| 2021-07-09:12:15:45 | 2021-07-09T12:15:45 |
| 09JUL2021T12:15 AM | 2021-07-09T00:15 |
| 09JUL2021:12:15 PM | 2021-07-09T12:15 |

| DAT | DTC |
|---|---|
| 2021-07-09T :15 | 2021-07-09T-:15 |
| 09JUL2021 9:15 | 2021-07-09T09:15 |
| 09JUL2021 :15 | 2021-07-09T-:15 |
| 09JUL2021 T9:15 | 2021-07-09T-:09:15 |
| UNJUL2021 TUN:UN | 2021-07 |
| UNJUL2021TUN:UN | 2021-07 |
| 2021-UN-09TUN:15 | 2021---09T-:15 |
| 2021-UN-UNTUN:UN | 2021 |
| 20210709:12:15 | 2021-07-09T12:15 |
| 20210709 12:15 | 2021-07-09T12:15 |
| 20210709T12:15 | 2021-07-09T12:15 |

| DAT | DTC |
|---|---|
| 09JUL2021:2:15 PM | 2021-07-09T14:15 |
| 19-JUL-2021:2:15 PM | 2021-07-19T14:15 |
| UN-UNK-2021T3:25 PM | 2021----T15:25 |
| UN-MOR-2021 T5:25 PM | 2021----T-:05:25 |
| 0729 | _ERROR_ |
| 0715 T5 | _ERROR_ |
| 07/08 | _ERROR_ |
| 07/19 5:12 | _ERROR_ |
| T13:14 | ---T13:14 |
| JUL 9, 2021:12:15 | _ERROR_ |

*Data Display 6: Datetime conversion using ISO_DTTM function with datetime in one field*

Earlier it was mentioned that messages would be output to the log when the format was invalid, or there was insufficient information to determine how to convert to the proper ISO 8601 notation. In the data the value will display "_ERROR_" and one of the log messages shown below in SAS Log 1 will appear depending on the circumstances. The 'XXXXX' will display the input value to the function as shown in SAS Log 2.

```
WARNING: XXXXX insufficient to determine if date part represents YYYY or DDMM or MMDD.
WARNING: XXXXX insufficient to determine if date part represents DDMMM or YYYYM or YYMMDD
or YYYYMM or MMDDYY.
WARNING: XXXXX date part not in a usable format.
```

*SAS Log 1: Sample log messages*

```
WARNING: 0729 insufficient to determine if date part represents YYYY or DDMM or MMDD.
WARNING: 0715 T5 insufficient to determine if date part represents YYYY or DDMM or MMDD.
WARNING: 07/08 insufficient to determine if date part represents DDMMM or YYYYM or YYMMDD
or YYYYMM or MMDDYY.
WARNING: 07/19 5:12 insufficient to determine if date part represents DDMMM or YYYYM or
YYMMDD or YYYYMM or MMDDYY.
WARNING: JUL 9, 2021:12:15 date part not in a usable format.
```

*SAS Log 2: Sample log messages from sample run*

But what if the date and time are in separate fields in the data, such as shown in Data Display 7. The date is captured in one field and the time is captured in a separate field.

| DT | TM |
|---|---|
| 09JUL2021 | |
| 09JUL2021 | 12:15 |
| 09JUL2021 | 12:15 |
| UNJUL2021 | |
| UNUNK2021 | |
| 2021-UN-UN | |
| 2021-07-09 | |
| 2021-07-09 | 12:15 |
| 2021-07-09 | 12:15:45 |
| 2021-07-09 | 12:15:45 |
| 2021-07-09 | UN:15 |
| 2021-07-09 | 9:15 |

| DT | TM |
|---|---|
| 2021UNUN | 12:15 |
| 202107UN | UN:UN |
| 2021UN09 | UN:UN |
| 20210709 | UN:UN |
| 2021UNUN | UN:UN |
| 2021UNUN | UN:UN |
| 2021-07-09 | 12:15:45 |
| 2021-07-09 | 12:15:45 |
| 2021-07-09 | 12:15:45 |
| 2021-07-09 | 12:15:45 |
| 09JUL2021 | 12:15 AM |
| 09JUL2021 | 12:15 PM |

| DT | TM |
|---|---|
| 2021-07-09 | :15 |
| 09JUL2021 | 9:15 |
| 09JUL2021 | :15 |
| 09JUL2021 | T9:15 |
| UNJUL2021 | TUN:UN |
| UNJUL2021 | UN:UN |
| 2021-UN-09 | UN:15 |
| 2021-UN-UN | UN:UN |
| 20210709 | 12:15 |
| 20210709 | 12:15 |
| 20210709 | 12:15 |

| DT | TM |
|---|---|
| 09JUL2021 | 2:15 PM |
| 19-JUL-2021 | 2:15 PM |
| UN-UNK-2021 | 3:25 PM |
| UN-MOR-2021 | T5:25 PM |
| 0729 | |
| 0715 | T5 |
| 07/08 | |
| 07/19 | 5:12 |
| | T13:14 |
| JUL 9, 2021 | 12:15 |

*Data Display 7: Raw dates from various data sources date captured in one field and time captured in separate field*

Even though the date and time are captured in two different fields, the user-defined function ISO_DTTM can still be used. You still need to make sure you indicate where the compiled subroutines and functions are stored if this has not already been done in the program. The difference with having the date and time stored in two fields is that you will need to concatenate them together when you call the user-defined function. The nice thing about SAS functions is that you can nest functions; therefore, you can use the CATX function to concatenate the date and time. When concatenating the date and time you need to use one of the following delimiters: space, T, or colon. The user-defined function is set up to look for one of those specific delimiters. Refer to SAS Program 8 for an illustration of this technique. The results are shown in Data Display 8 and are the same as Data Display 7 and yield the same warnings in the log.

```
options cmplib = fcmp.funcs;

data dateparts2;
   set dateparts;
   DTC = ISO_DTTM(catx('T', DT, TM));
run;
```

*SAS Program 8: Using ISO_DTTM function with date captured in one field and time captured in separate field*

| DT | TM | DTC |
|---|---|---|
| 09JUL2021 | | 2021-07-09 |
| 09JUL2021 | 12:15 | 2021-07-09T12:15 |
| 09JUL2021 | 12:15 | 2021-07-09T12:15 |
| UNJUL2021 | | 2021-07 |
| UNUNK2021 | | 2021 |
| 2021-UN-UN | | 2021 |
| 2021-07-09 | | 2021-07-09 |
| 2021-07-09 | 12:15 | 2021-07-09T12:15 |
| 2021-07-09 | 12:15:45 | 2021-07-09T12:15:45 |
| 2021-07-09 | 12:15:45 | 2021-07-09T12:15:45 |
| 2021-07-09 | UN:15 | 2021-07-09T-:15 |
| 2021-07-09 | 9:15 | 2021-07-09T09:15 |
| 2021-07-09 | :15 | 2021-07-09T-:15 |
| 09JUL2021 | 9:15 | 2021-07-09T09:15 |
| 09JUL2021 | :15 | 2021-07-09T-:15 |
| 09JUL2021 | T9:15 | 2021-07-09T-:09:15 |
| UNJUL2021 | TUN:UN | 2021-07 |
| UNJUL2021 | UN:UN | 2021-07 |
| 2021-UN-09 | UN:15 | 2021---09T-:15 |
| 2021-UN-UN | UN:UN | 2021 |
| 20210709 | 12:15 | 2021-07-09T12:15 |
| 20210709 | 12:15 | 2021-07-09T12:15 |
| 20210709 | 12:15 | 2021-07-09T12:15 |

| DT | TM | DTC |
|---|---|---|
| 2021UNUN | 12:15 | 2021----T12:15 |
| 202107UN | UN:UN | 2021-07 |
| 2021UN09 | UN:UN | 2021---09 |
| 20210709 | UN:UN | 2021-07-09 |
| 2021UNUN | UN:UN | 2021 |
| 2021UNUN | UN:UN | 2021 |
| 2021-07-09 | 12:15:45 | 2021-07-09T12:15:45 |
| 2021-07-09 | 12:15:45 | 2021-07-09T12:15:45 |
| 2021-07-09 | 12:15:45 | 2021-07-09T12:15:45 |
| 2021-07-09 | 12:15:45 | 2021-07-09T12:15:45 |
| 09JUL2021 | 12:15 AM | 2021-07-09T00:15 |
| 09JUL2021 | 12:15 PM | 2021-07-09T12:15 |
| 09JUL2021 | 2:15 PM | 2021-07-09T14:15 |
| 19-JUL-2021 | 2:15 PM | 2021-07-19T14:15 |
| UN-UNK-2021 | 3:25 PM | 2021----T15:25 |
| UN-MOR-2021 | T5:25 PM | 2021----T-:05:25 |
| 0729 | | _ERROR_ |
| 0715 | T5 | _ERROR_ |
| 07/08 | | _ERROR_ |
| 07/19 | 5:12 | _ERROR_ |
| | T13:14 | ---T13:14 |
| JUL 9, 2021 | 12:15 | _ERROR_ |

*Data Display 8: Datetime conversion using ISO_DTTM function with date captured in one field and time captured in separate field*

In case you are wondering, the subroutines can be used outside of the user-defined ISO_DTTM function. They can be used directly in a DATA step. If the data comes in with each date and time broken out into individual components (Data Display 9), you can use the DTTMFMT format without using the ISO_DTTM function. This will allow you to convert the components to the proper format and then you can either add additional SAS statements to combine and ensure that only the last non-missing component is displayed, or you can build a separate function (see Appendix B) to handle this logic.

| MO | DY | YR | HR | MN | SC |
|---|---|---|---|---|---|
| JUL | 9 | 2021 | | | |
| JUL | 9 | 2021 | 12 | 15 | |
| JUL | UN | 2021 | | | |
| UNK | UN | 2021 | | | |
| 07 | 09 | 2021 | 9 | 15 | 1 |
| 07 | 9 | 21 | 12 | 5 | |
| JUL | UN | 21 | UN | Un | |

*Data Display 9: Date and time components captured as individual fields with unknowns indicated with UN or UNK*

SAS Program 9 illustrates the use of the DTTMFMT subroutine directly in a DATA step without the ISO_DTTM function. The results are displayed in Data Display 10. Recall that a subroutine overwrites any argument that is indicated in the OUTARGS statement. Therefore, all the variables are replaced with the appropriately formatted value and no new variables are created.

```
options cmplib = fcmp.funcs;

data ind_comp_dt2;
   set ind_comp_dt;
   call dttmfmt(YR, MO, DY, HR, MN, SC);
run;
```

*SAS Program 9: Using DTTMFMT subroutine without ISO_DTTM function*

Recall that a subroutine does not return a value; instead it modifies the values that are indicated in the OUTARGS statement.

| MO | DY | YR | HR | MN | SC |
|---|---|---|---|---|---|
| - | 09 | 2021 | - | - | - |
| - | 09 | 2021 | 12 | 15 | - |
| - | - | 2021 | - | - | - |
| - | - | 2021 | - | - | - |
| 07 | 09 | 2021 | 09 | 15 | 01 |
| 07 | 09 | 2021 | 12 | 05 | - |
| - | - | 2021 | - | - | - |

*Data Display 10: Date and time components modified using DTTMFMT subroutine*

Furthermore, you can even use the ZFILL subroutine as a stand-alone subroutine. It does not need to be used with the DTTMFMT subroutine or the ISO_DTTM function. For example, if date and time components are captured as individual numeric fields stored as character values but the unknown values are left null then the ZFILL subroutine can be used to ensure that the values have the appropriate format. Note that for the ZFILL subroutine to work it is expecting a character value. If the individual components are not character fields, they would need to be converted to character values since a subroutine will update the existing value, and if it is missing it will attempt to update with a '-' (i.e., a non-numeric character). Data Display 11 illustrates a data set where the values are numeric but stored as character and unknowns are left null.

| MO | DY | YR | HR | MN | SC |
|---|---|---|---|---|---|
| 7 | 9 | 2021 | 1 | 1 | 1 |
| 7 | 09 | 2021 | 1 | 10 | 07 |
| | | 2021 | 7 | | 20 |

*Data Display 11: Date and time components captured as individual fields unknown values are left null*

SAS Program 10: Using ZFILL subroutine without DTTMFMT subroutine and ISO_DTTM function takes the data in Data Display 11 and converts all the two-digit fields to the proper format as seen in Data Display 12. For the sake of this example, it is assumed that YR is in the proper format and does not need to be updated. Notice that ZFILL is called for each of the individual components and each component is replaced with the proper format. As with the example illustrated in SAS Program 9: Using DTTMFMT subroutine without ISO_DTTM function. you can add additional programming logic to combine the fields and ensure the only up the last non-missing component is displayed, or you can use a second user-defined function to perform these steps (Appendix B).

```
options cmplib = fcmp.funcs;

data dt_zero2;
  set dt_zero;
  call zfill(MO);
  call zfill(DY);
  call zfill(HR);
  call zfill(MN);
  call zfill(SC);
run;
```

*SAS Program 10: Using ZFILL subroutine without DTTMFMT subroutine and ISO_DTTM function*

| MO | DY | YR | HR | MN | SC |
|---|---|---|---|---|---|
| 07 | 09 | 2021 | 01 | 01 | 01 |
| 07 | 09 | 2021 | 01 | 10 | 07 |
| - | - | 2021 | 07 | - | 20 |

*Data Display 12: Date and time components modified using ZFILL subroutine*

## CONCLUSION

While SAS provides a plethora of functions and subroutines, there are times you need to have something that is user specific to meet your needs. With PROC FCMP, programmers can build their own functions and subroutines. Users can build a subroutine and/or function that can stand-alone (i.e., called directly from a DATA step) or that can be utilized within another subroutine/function. With the ability to create custom subroutines and functions, developers can control what values are returned or modified. Adding PROC FCMP to your programming toolkit can expand your repertoire of useful SAS functions.

## REFERENCES AND RECOMMENDED READINGS


Carpenter, A. (2013). Using PROC FCMP to the Fullest: Getting Started and Doing More. *SAS Global Forum.* San Francisco. Retrieved Apr 2022, from https://support.sas.com/resources/papers/proceedings13/139-2013.pdf

Eberhardt, P. (2011). A Cup of Coffee and Proc FCMP: I Cannot Function Without Them. *PharmaSUG.* Nashville: PharmaSUG. Retrieved Apr 2022, from https://www.lexjansen.com/pharmasug/2011/TU/PharmaSUG-2011-TU07.pdf

Hughes, T. M. (2022). *SAS® Data-Driven Development: from abstract design to dynamic functionality, second edition.* Troy Martin Hughes.

*ISO 8601 Date and Time Format*. (n.d.). Retrieved Apr 2022, from International Organization for Standardization: https://www.iso.org/iso-8601-date-and-time-format.html

Morgan, D. (2017). ISO 8601 and SAS®: A Practical Approach. *MWSUG.* St. Louis: MWSUG. Retrieved Jun 2022, from https://www.mwsug.org/proceedings/2017/PH/MWSUG-2017-PH02.pdf

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

SAS Institute Inc. (2019, 11 Apr). *About SAS Date, Time, and Datetime Values*. Retrieved Jun 2022, from SAS® 9.4 and SAS® Viya® 3.5 Programming Documentation: https://documentation.sas.com/doc/en/pgmsascdc/9.4_3.5/lrcon/p1wj0wt2ebe2a0n1lv4lem9hdc0v.htm

SAS Institute Inc. (2022, Apr 14). *CALL IS8601_CONVERT Routine*. Retrieved Jun 2022, from SAS® 9.4 and SAS® Viya® 3.2 Programming Documentation: https://documentation.sas.com/doc/en/pgmsascdc/9.4_3.2/lefunctionsref/p0bhy7ndmdivmmn10b2okmbgiqmj.htm#p0h41lprw0az65n1tafhcnsqndru

SAS Institute Inc. (2022, May 24). *FCMP Procedure*. Retrieved Jun 2022, from SAS® 9.4 and SAS® Viya® 3.2 Programming Documentation: https://documentation.sas.com/doc/en/pgmsascdc/9.4_3.5/proc/p10b4qouzgi6sqn154ipglazix2q.htm

SAS Institute Inc. (2022, May 13). *Working with Dates and Times by Using the ISO 8601 Basic and Extended Notations*. Retrieved Jun 2022, from SAS® 9.4 and SAS® Viya® 3.5 Programming Documentation: https://documentation.sas.com/doc/en/pgmsascdc/9.4_3.5/leforinforref/p1a0qt18rxydrkn1b0rtdfh2t8zs.htm

Watson, R., & Hadden, L. (2022). Functions (and More!) on CALL! *PharmaSUG.* PharmaSUG. Retrieved Jun, from https://www.pharmasug.org/proceedings/2022/AP/PharmaSUG-2022-AP-111.pdf

Wilson, K. (2012). Harnessing the Power of SAS ISO 8601 Informats, Formats, and the CALL IS8601_CONVERT Routine. *PhramaSUG.* San Francisco: PharmaSUG. Retrieved Jun 2022, from https://www.lexjansen.com/pharmasug/2012/DS/PharmaSUG-2012-DS22-SAS.pdf


## ACKNOWLEDGMENTS


Thanks to Louise Hadden and Josh Horstman helping me come up with ‘perfect’ way to phrase my thoughts so that I was able to adequately convey a concept.

Thanks to Troy Martin Hughes for being so patient with me as I sent him copious text messages asking for his advice and input and for reviewing this paper to ensure that I was not misstating anything.


## CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors at:


Richann Watson
DataRich Consulting
richann.watson@datarichconsulting.com

## APPENDIX A: USER-DEFINED SUBROUTINES AND FUNCTIONS FOR CREATING ISO 8601 DATETIMES

```
libname fcmp 'C:\Users\gonza\Desktop\Conferences\FCMP';

proc fcmp outlib = fcmp.funcs.ISO_date;
   /* need to zero fill each non-missing month, day, hour, minute, second */
   subroutine zfill(_comp $);
      outargs _comp;
      if not missing(_comp) and not( notdigit(cats(_comp)) )
            then _comp  = put(input(_comp, best.), Z2.);
      else _comp = '-';
   endsub;

   /* input values are character so need $ for each */
   subroutine dttmfmt(_year $, _month $, _day $,
                      _hour $, _minute $, _second $);
      outargs _year, _month, _day, _hour, _minute, _second;
      /* make sure year is a four-digit number */
      if not( notdigit(cats(_year)) ) then do;
         if length(strip(_year)) = 4 then _year = strip(_year);
         else if length(strip(_year)) = 2 then do;
           if input(_year, best.) <= 40 then _year = cats('20', _year);
           else _year = cats('19', _year);
         end;
      end;
      else _year = '-';

      /* need to 0 fill each non-missing month, day, hour, minute, second */
      call zfill(_month);
      call zfill(_day);
      call zfill(_hour);
      call zfill(_minute);
      call zfill(_second);
   endsub;

   function ISO_DTTM(dattim $) $;
      length __dtc __dttm $20 iso_dtc $10 iso_tmc $8
             year $4 month $2 day $2;

      /* extract the date portion; compress hyphens to determine format */
      __dtpart = compress(scan(dattim, 1, ' T:', 'm'), '-');

      /* need to convert all dates to the yymmdd10 format so that further */
      /* processing can proceed - if not then need to get out             */
      if length(strip(__dtpart)) in (7 9) then do;
         year = strip(substr(__dtpart, 6));
         if strip(substr(__dtpart, 3, 3)) in ('JAN', 'FEB', 'MAR', 'APR',
'MAY', 'JUN', 'JUL', 'AUG', 'SEP', 'OCT', 'NOV', 'DEC')
             then month = put(whichc(strip(substr(__dtpart, 3, 3)), 'JAN',
'FEB', 'MAR', 'APR', 'MAY', 'JUN', 'JUL', 'AUG', 'SEP', 'OCT', 'NOV',
'DEC'), Z2.);
         else month = substr(__dtpart, 3, 2);
         day = strip(substr(__dtpart, 1, 2));
      end;
      else if length(strip(__dtpart)) = 8 then do;
         year = strip(substr(__dtpart, 1, 4));
```

```
         month = strip(substr(__dtpart, 5, 2));
         day = strip(substr(__dtpart, 7, 2));
      end;
      else if lengthn(strip(__dtpart)) = 0
          then call missing(year, month, day);
      else do;
         if length(strip(__dtpart)) = 4 then
           put %sysfunc(compress("WARN ING:")) dattim "insufficient to
determine if date part represents YYYY or DDMM or MMDD.";
         else if length(strip(__dtpart)) in (5 6) then
           put %sysfunc(compress("WARN ING:")) dattim "insufficient to
determine if date part represents DDMMM or YYYYM or YYMMDD or YYYYMM or
MMDDYY.";
         else
           put %sysfunc(compress("WARN ING:")) dattim "date part not in a
usable format.";
         __dtc = '_ERROR_';
         return(__dtc);
      end;

      /* extract time portion to see if hh:mm:ss (assume 24-hr clock) */
      /* or if HH:MM:SS AM/PM (12-hr clock)                           */
      __tmpart = substr(dattim, prxmatch('/T|:| /i', dattim) );
      if first(__tmpart) in ('T' ':') then __tmpart = substr(__tmpart, 2);

      if prxmatch('/AM|PM/i', __tmpart) then __tmpart2 =
          transtrn(transtrn(__tmpart, 'AM', trimn('')), 'PM', trimn(''));
      else __tmpart2 = __tmpart;

      /* split the time components */
      array tm_c (3) $ hour minute second;
      Io i = 1 to 3;
         tm_c[i] = scan(strip(__tmpart2), i, 'T:', 'm');
         if strip(tm_c[i]) in ('' '.') then tm_c[i] = '-';
      end;

      if find(__tmpart, 'PM') and strip(hour) ne '12' and
         not(notdigit(strip(hour))) then
          hour = put(input(hour, best.) + 12, Z2.);
      else if find(__tmpart, 'AM') and strip(hour) = '12' then hour = '00';

      call dttmfmt(year, month, day, hour, minute, second);

      /* use new variables to build ISO 8601 dates in the proper format */
      iso_dtc = catx('-', year, month, day);
      iso_tmc = catx(':', hour, minute, second);

      /* if time is nothing but '-' and ':' then default to blank */
      /* if there is at least one number portion then need to keep*/
      /* up through the last time element that has a numeric part */
      if notpunct(strip(iso_tmc)) > 0 then _iso_tmc =
          substr(iso_tmc, 1, notpunct(strip(iso_tmc), -length(iso_tmc)));
      else call missing(_iso_tmc);

      /* combine time with date to build ISO datetime */
      __dttm = catx('T', iso_dtc, _iso_tmc);
```

```
      /* if no time portion keep only up to last numeric portion of date */
      if anyalpha(strip(__dttm)) > 0 then __dtc = __dttm;
      else if notpunct(strip(__dttm)) > 0 then __dtc =
          substr(__dttm, 1, notpunct(strip(__dttm), -length(__dttm)));
      else call missing(__dtc);

      return(__dtc);
   endfunc;
quit;
```

## APPENDIX B: USER-DEFINED FUNCTION USING INDIVIDUAL COMPONENTS FOR CREATING ISO 8601 DATETIMES

User-defined function that combines the individual components into the proper ISO 8601 notation.

```
proc fcmp outlib = fcmp.funcs.ISO_DATES_PT;
   function dtisopt(mn $, dy $, yr $, hr $, mi $, sc $) $;
      length NEWDTC newdttm $20 isodt $10 isotm $8;

      /* use new variables to be build ISO 8601 date time */
      isodt = catx("-", yr, mn, dy);
      isotm = catx(":", hr, mi, sc);

      /* if time is nothing but '-' and ':' then default to blank */
      /* if it there is so numeric portion of the time then keep  */
      /* up through the last time element that had a numeric part */
      if notpunct(strip(isotm)) > 0 then
          _isotm = substr(isotm, 1, notpunct(strip(isotm), -length(isotm)));
      else _isotm = ' ';

      /* combine time with date to build ISO datetime */
      newdttm = catx("T", isodt, _isotm);

      /* if there is no time portion then keep only up */
      /* through last numeric portion of date          */
      if anyalpha(strip(newdttm)) > 0 then NEWDTC = newdttm;
      else if notpunct(strip(newdttm)) > 0 then NEWDTC =
          substr(newdttm, 1, notpunct(strip(newdttm), -length(newdttm)));
      else NEWDTC = ' ';

      return(NEWDTC);
   endfunc;
quit;
```