Previous Page | Next Page

CALL IS8601_CONVERT Routine

Converts an ISO 8601 interval to datetime and duration values, or converts datetime and duration values to an ISO 8601 interval.

Category: Date and Time

Syntax

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

Required Arguments

convert-from

specifies a keyword in single quotation marks that indicates the source for the conversion, such as a date, a datetime and duration, a duration, or an interval value. *Convert-from* can have one of the following values:

'd*n*'

specifies a date value, where *n* is a value from 1 to 6. The default value is 1. *N* is the number of components in the *from-variables* or *to-variables* arguments. The following components are valid:

- year
- month
- day
- hour
- minute
- second

'dt*n*'

specifies a datetime value, where *n* is a value from 1 to 6. The default value is 1. *N* is the number of components in the *from-variables* or *to-variables* arguments. The following components are valid:

- year
- month
- day
- hour
- minute
- second

'dt/dt'

specifies that the source value for the conversion is a datetime/datetime value.

'dt/du'

specifies that the source value for the conversion is a datetime/duration value.

'du*n*'

specifies that the source value for the conversion is a duration value, where *n* is a value from 1 to 6. The default value is 1. *N* is the number of components in the *from-variables* or *to-variables* arguments. The following components are valid:

- year
- month
- day
- hour
- minute
- second

'du/dt'

specifies that the source value for the conversion is a duration/datetime value.

'intvl'

specifies that the source value for the conversion is an interval value.

convert-to

specifies a keyword in single quotation marks that indicates the results of the conversion. *Convert-to* can have one of the following values:

'd*n*'

specifies a date value, where *n* is a value from 1 to 6. The default value is 1. *N* is the number of components in the *from-variables* or *to-variables* arguments. The following components are valid:

- year
- month
- day
- hour
- minute
- second

'dt*n*'

specifies a datetime value, where *n* is a value from 1 to 6. The default value is 1. *N* is the number of components in the *from-variables* or *to-variables* arguments. The following components are valid:

- year
- month
- day
- hour
- minute
- second

'dt/dt'

specifies to create a datetime/datetime interval.

'dt/du'

specifies to create a datetime/duration interval.

'du*n*'

specifies to create a duration, where *n* is a value from 1 to 6. The default value is 1. *N* is the number of components in the *from-variables* or *to-variables* arguments. The following components are valid:

- year
- month
- day
- hour
- minute
- second

'du/dt'

specifies to create a duration/datetime interval.

'end'

specifies to create a value that is the ending datetime or duration of an interval value.

'intvl'

specifies to create an interval value.

'start'

specifies to create a value that is the beginning datetime or duration of an interval value.

Optional Arguments

from-variable

specifies one or two variables that contain the source value. Specify one variable for an interval value, and specify two variables, one each, for datetime and duration values. The datetime and duration values are interval components where the first value is the beginning value of the interval and the second value is the ending value of the interval.

Requirements	An integer variable must be at least a 16-byte character variable whose value is read by the $N8601B*w.d* informat or the $N8601E*w.d* informat. The integer variable can also be an integer value that is returned by invoking the CALL IS8601_CONVERT routine.
	A datetime value must be a SAS datetime value or an 8-byte character value that is read by the $N8601B*w.d* informat or the $N8601E*w.d* informat, or by invoking the CALL IS8601_CONVERT routine.
	A duration value must be a numeric value that represents the number of seconds in the duration, or an 8-byte character value whose value is read by the $N8601B*w.d* informat or the $N8601E*w.d* informat, or by invoking the CALL IS8601_CONVERT routine.

to-variable

specifies one or two variables that contain converted values. Specify one variable for an interval value, and specify two variables, one each, for datetime and duration values.

Requirement	The interval variable must be a character variable that is at least 16 bytes.
Tip	The datetime and duration variables can be numeric or character. To avoid losing precision of a numeric value, the length of a numeric variable must be at least eight characters. Datetime and duration character variables must be at least 16 bytes; they are padded with blank characters for values that are less than the length of the variable.

date-time-replacements

specifies date or time component values to use when a month, day, or time component is omitted from an interval, datetime, or duration value. The *date-time-replacements* argument is specified as a series of numbers that are separated by commas and represent components in this order: year, month, day, hour, minute, and second. Components of *date-time-replacements* can be omitted only in the reverse order: second, minute, hour, day, month, and year. If no substitute values are specified, the conversion is performed using default values.

Default	The following default values for date and time components are omitted: 1 month 1 day 0 hour 0 minute 0 second
Requirement	A year component must be part of the datetime or duration value and therefore is not valid in *date-time-replacements*. A comma is required as a placeholder for the year in *date-time-replacements*. For example, in the replacement value string, ,9,4,,2,' , the first comma is a placeholder for a year value.

Details

The Basics

How Arguments in the CALL IS8601_CONVERT Routine Are Used

The Basics

ISO 8601 representations of date, time, and datetime values within the ISO 8601 standards consist of basic and extended notations. A value is basic when delimiters that separate the various components within a value are omitted. A value is extended when delimiters are used to separate those components. A set of formats and informats are used with ISO 8601 date, time, and datetime values. For more information, see Working with Dates and Times By Using the ISO 8601 Basic and Extended Notations in *SAS Formats and Informats: Reference* and Reading Dates and Times By Using the ISO 860 Basic and Extended Notations in *SAS Formats and Informats: Reference*.

After your values are stored as SAS variables, you can calculate intervals, durations, and datetime values with the CALL IS8601_CONVERT routine. This routine enables you to convert an ISO 8601 interval to datetime and duration values, or to convert datetime and duration values to an ISO 8601 interval. The CALL IS8601_CONVERT routine accepts missing values in the d*n*, dt*n*, and du*n* arguments. You can also use the CALL IS8601_CONVERT routine to perform calculations with dates and times.

You can use the year, month, day, hour, minute, and second components in any order.

How Arguments in the CALL IS8601_CONVERT Routine Are Used

The first argument to the CALL IS8601_CONVERT routine, *convert-from*, can be one or two values. The number of values is based on how many variables you provide to the routine for an expected result. For example, datetime and duration values can be specified with an expected output of an interval. In this example, the supplied first argument would be `'dt/du'`. If two datetime values are supplied with an expected duration as output, the first argument would be `'dt/dt'`.

The second argument to the CALL IS8601_CONVERT routine, *convert-to*, can also be one or two values. The values depend on the type of result that you expect SAS to compute using the input that you supply in the first argument.

Note: The minimal length for durations is 16, and the minimum length for intervals is 32.

Examples

Example 1: Using the CALL IS8601_CONVERT Routine

This DATA step uses the CALL IS8601_CONVERT routine to perform these tasks:

- create an interval by using datetime and duration values
- create datetime and duration values from an interval that was created using the CALL IS8601_CONVERT routine
- create an interval from datetime and duration values by using replacement values for omitted date and time components in the datetime value

For easier reading, numeric variables end with an N, and character variables end with a C.

```
data _null_;
      /* Declare variable length and type. Character datetime and duration   */
      /* values must be at least 16 characters. To avoid losing precision,   */
      /* the numeric datetime value has a length of 8.                       */
   length dtN duN 8 dtC duC $16 intervalC $32;
      /* Assign a numeric datetime value and a character duration value.     */
   dtN='15Sep2011:09:00:00'dt;
   duC=input('P2y3m4dT5h6m7s', $n8601b.);
   put dtN=;
   put duC=;
      /* Create an interval from a datetime and duration value and format the */
      /* interval using the ISO 8601 extended notation for character values.  */
   call is8601_convert('dt/du', 'intvl', dtN, duC, intervalC);
   put '**  Character interval created from datetime and duration values **/';
   put intervalC $n8601e.;
   put ' ';
      /* Create numeric datetime and duration values from an interval and     */
      /* format the values using the ISO 8601 extended notation for numeric   */
      /* values.                                                              */
   call is8601_convert('intvl', 'dt/du', intervalC, dtN, duN);
   put '**  Character datetime and duration created from an interval  **/';
   put dtN=;
   put duN=;
   put ' ';
      /* Assign a new datetime value with omitted components.                  */
   dtC=input('2012---15T10:-:-', $n8601b.);
   put '** This datetime is a character value. **';
   put dtC $n8601h.;
   put ' ';
      /* Create an interval by reading a datetime value that has omitted date  */
      /* and time components.  Use replacement values for the month, minutes,  */
      /* and seconds.                                                          */
   call is8601_convert('du/dt', 'intvl', duC, dtC, intervalC, , 7, , , 35, 45);
   put '**  Interval created using a datetime with omitted values,      **';
   put '**  inserting replacement values for month (7), minute (35)     **';
```

```
      put '**  seconds (45).                                        **';
      put intervalC $n8601e.;
      put ' ';
   run;
```

SAS writes the following results to the log:

```
   dtN=1631696400
   duC=0002304050607FFC
   **  Character interval created from datetime and duration values **/
   2011-09-15T09:00:00.000/P2Y3M4DT5H6M7S

   **  Character datetime and duration created from an interval  **/
   dtN=1631696400
   duN=71211967

   ** This datetime is a character value. **
   2012---15T10:-:-

   **  Interval created using a datetime with omitted values,      **
   **  inserting replacement values for month (7), minute (35)     **
   **  seconds (45).                                               **
   P2Y3M4DT5H6M7S/2012-07-15T10:35:45
```

Example 2: Finding an Interval Value

This example writes duration and datetime values to the log.

```
   data _null_;
      length mynew $32;
      x='P8w';
      y='11feb2012:12:35:22'dt;
      call is8601_convert('du/dt', 'intvl', x, y, mynew);
      put mynew=$n8601e.;
   run;
```

SAS writes the following results to the log:

```
   mynew=P8W/2012-02-11T12:35:22.000
```

Example 3: Finding the Start Date of an Interval

This example returns the start date of an interval.

```
   data temp;
      length dt $32;
      x='P6w';
      y='11feb2012:11:13:22'dt;
      call is8601_convert('du/dt', 'start', x, y, dt);
      put dt=$n8601e.;
   run;
```

SAS writes the following results to the log:

```
   dt=2011-12-31T00:00:00.000
```

In this example, P6w specifies a six-week duration, and *y* is the date from which *start* is calculated. *start* is the beginning duration of an interval. This datetime value is six weeks before *y*.

Example 4: Finding an Interval Using Duration and Datetime Values

This example uses duration and datetime values to find an interval.

```
   data _null_;
      infile datalines;
      input mo d yr hour min sec;
      length mynew $32;
      x='P8w';
      call is8601_convert('du/dt6', 'intvl', x, mo, d, yr, hour, min, sec, mynew);
      put mynew=$n8601e.;
      datalines;
   02 22 2011 10 30 45
   12 13 2010 12 35 25
   03 26 2010 10 10 10
   ;
   run;
```

SAS writes the following results to the log:

```
         P8W/2002-14-21T10:30:45.000
```

```
mynew=P8W/0002-14-91T10:30:45.000
mynew=P8W/0012-13-90T12:35:25.000
mynew=P8W/0003-14-90T10:10:10.000
```

Example 5: Finding the Start Date for Multiple Intervals

This example uses data lines as input to find the start date for an interval.

```
data temp;
   input y mo d h min s;
   length mynew $16;
      /* This value is the duration. */
   x='P2w';
      /* This CALL routine uses the date and time values that are */
      /* listed in the data lines to create a DATETIME value.     */
   call is8601_convert('dt6', 'dt', y, mo, d, h, min, s, dt);
   put dt=datetime.;
      /* This CALL routine uses the DATETIME value that was previously */
      /* created.                                                      */
   call is8601_convert('du/dt', 'start', x, dt, mynew);
   put mynew=$n8601e.;
datalines;
2011 6 7 10 15 20
2011 12 5 10 25 45
2011 6 30 10 32 20
;
```

SAS writes the following results to the log:

```
dt=07JUN11:10:15:20
mynew=2011-05-24T00:00:00.000
dt=05DEC11:10:25:45
mynew=2011-11-21T00:00:00.000
dt=30JUN11:10:32:20
mynew=2011-06-16T00:00:00.000
```

Example 6: Converting a Duration to a SAS Time

In this example, you supply the duration value P8w, which specifies eight weeks. This type of value is often coupled with a datetime value to produce a duration, but the value can be used alone and converted to a SAS time value. There is no SAS informat to convert the P8w value to a SAS time value, but you can use the CALL IS8601_CONVERT routine to convert the value.

```
data a;
   x='P8w';
   call is8601_convert('du', 'du', x, mynew);
   put mynew=time8.;
run;
```

The value of Mynew is 1344:00, which indicates 1344 hours.

In this example, the following statements apply:

- The *X* variable is a duration. Therefore, 'du' is the first argument.
- A time value is expected as output, but there is no *convert-to* value for time. As a result, the value for 'du' is used for duration. Both arguments require single quotation marks.
- *X* is the variable name that is supplied, and Mynew is the variable that is being created.
- Because the result is a SAS time value, you can use a SAS time format (TIME8.).

Example 7: Converting a Duration Value and a Datetime Value to an Interval

This example takes the previous example one step further, by using the same duration value with a datetime value to create an interval as output. The following DATA step is based on an event that lasts for eight weeks, ending on February 11, 2012, at 12:22 p.m.

```
data a;
   length mynew $32;
   x='P8w';
   y='11feb2012:12:22'dt;
   call is8601_convert('du/dt', 'intvl', x, y, mynew);
   put mynew=$n8601e.;
run;
```

The value of Mynew is P8W/2012-02-11T12:22:00.000.

In this example, the following statements apply:

- The first argument, 'du/dt', to the CALL IS8601_CONVERT routine indicates the types of variables that are being passed in for conversion. The value 'du/dt' specifies that two variables are being passed: one is a duration value, *X*, and the other is a datetime value, *Y*. Because the duration value 'du/dt' specifies that two variables are being passed: one is a duration value, *X*, and the other is a datetime value, *Y*. Because the duration value comes before the datetime value, the datetime value is assumed to be the end of the interval.

- The result that you want from this DATA step is an interval. Therefore, the second argument is 'intvl'.

- The remaining arguments name the incoming variables, *X* and *Y*, and the new variable, Mynew, that is being created. The order of these variables must match the types of variables that are specified in the first argument. For example, if *X* and *Y* are reversed, the following note appears in the log:

```
NOTE: Invalid argument to function IS8601_CONVERT('du/dt','intvl',1644582120,
      'P8w',''[12 of 32 characters shown]) at line 770 column 9.
mynew=************************************************************
mynew=  x=P8w y=1644582120 _ERROR_=1 _N_=1
```

Example 8: Computing the Start Datetime of an Interval When You Have a Duration and a Datetime

Converting a Duration Value and a Datetime Value to an Interval computes an interval when datetime and duration values are supplied. This example computes the starting datetime for an interval when a duration and datetime value are supplied.

```
data a;
   length mynew $16;
      /* If you omit the LENGTH statement, replace the PUT statement */
      /* with 'put mynew datetime22.;'.                              */
   x='P8w';
   y='11feb2012:12:22'dt;
   call is8601_convert('du/dt', 'start', x, y, mynew);
   put mynew=$n8601e.;
run;
```

The value of Mynew is 2011-12-17T00:00:00.000.

In this example, the following statements apply:

- The CALL IS8601_CONVERT routine uses the argument `start` as the second argument in order to compute the starting datetime value for the interval.

- Because the SAS datetime value is computed, you can remove the LENGTH statement so that Mynew is created as a numeric variable. If you remove the LENGTH statement, add a PUT statement that specifies the DATETIME22. format. It is also valid to leave the LENGTH statement and create variable output by using the $N8601E*w*. format, as shown in the second PUT statement.

Example 9: Converting a Duration of One Type to a Duration of a Different Type

If you receive data that is represented in hours, you can perform a conversion similar to the previous example to obtain output as a duration.

```
data _null_;
   x='1271:59:00';
   time=input(x, time10.);
   length dur $16;
   call is8601_convert('du', 'du', time, dur);
   put dur=$n8601e.;
run;
```

The value of dur is P0Y1M22DT23H59M0.0S.

In this example, the following statements apply:

- The TIME*w.d* format reads the time value into SAS.

- The CALL IS8601_CONVERT routine converts the time value to a duration value.

Example 10: Converting Two Datetime Values to a Duration

This example determines the amount of time between two datetime values, and creates a duration as output.

```
data a;
   length dur $16;
   start='02apr2012:12:30:22'dt;
   end='08apr2012:14:32:22'dt;
   call is8601_convert('dt/dt', 'du', start, end, dur);
   put dur=$n8601e.;
run;
```

The value of dur is P6DT2H2M.

In this example, the following statements apply:

- The CALL IS8601_CONVERT routine uses the following arguments:
  - 'dt/dt' indicates that two datetime values are being passed into the function.

  - 'du' indicates that a duration value is the expected outcome.
  - 'start' is the name of the first datetime variable.
  - 'end' is the name of the second datetime variable.
  - 'dur' is the name of the desired output variable.
- A LENGTH statement is required in order to specify that Dur is a character variable and that the length should be 16. If you create an interval, a length of 32 is required.
- The routine creates an indecipherable hexadecimal value. Therefore, it is necessary to format the value with one of the $N8601 formats; in this case, $N8601E*w.d*.

Example 11: Converting an Interval to a Datetime Value and a Duration

You can specify an interval value as two datetime values that are separated by a forward slash (/). The slash separates the beginning and ending values for an event, as shown in this example.

```
data _null_;
   length Final2 $16;
   int=input('2012-03-15T14:32:00/2012-03-29T09:45:00', $n8601e40.);
   call is8601_convert('intvl', 'dt/du', int, Final1, Final2);
   put Final1= / Final2=$n8601e.;
run;
```

The resulting values for Final1 and Final2 are listed here:

- Final1=1647441120
- Final2=P13DT19H13M

In this example, the following statements apply:

- The datetime values are specified inside single quotation marks so that the INPUT function and the $N8601E*w*. informat can convert the values into an interval value in the variable *Int*.
- The CALL IS8601_CONVERT routine converts that interval value to two variables: Final1 is a datetime variable, and Final2 is a duration variable.
- Because a SAS datetime variable (Final1) is a numeric variable that can be stored in 8 bytes, a length value is not required for this variable in the LENGTH statement. However, a duration requires a length of 16. Therefore, a length is specified for Final2 in the LENGTH statement.
- Because Final2 is a duration, the $N8601E*w*. format is used to write its value to the log so that the output is understandable.

If the first datetime value in the example was missing the month value (2012- - -15T14:32:00), you could supply that value by specifying the value in the last arguments of the CALL routine, as shown here:

```
call is8601_convert('intvl', 'dt/du', int, Final1, Final2, , 2);
```

The month is specified here as 2, and the two consecutive commas preceding that value indicate that a year value was not supplied. Therefore, a datetime and a duration are computed based on the date 2012–02–15T14:32:00 and the other date that is specified in the code. In this case, the output from the PUT statement for each variable is as follows:

- Final1=1644935520
- Final2=P1M13DT19H13M

Previous Page | Next Page | Top of Page