Paper 086-2009

Could You Remind Me? Creating Calendar Reminders from SAS

Clarke Thacher, SAS Institute Inc., Cary, NC

ABSTRACT

The iCalendar standard allows users to send meeting requests and tasks to other users through e-mail. The recipient of an iCalendar event can add this event to their personal or corporate calendar system.

This paper demonstrates how simple DATA step programming can create and send events in iCalendar format. The application of this technique can range from the trivial, such as birthday reminders, to critical business processes.

INTRODUCTION

Everyone knows that SAS software is great for analyzing data, but sometimes you need to take that data and use it to produce actions at a specific time and date. Many people use programs such as Microsoft Outlook, Lotus Notes, or Google Calendar to maintain a calendar of important events. These calendar systems can accept entries from users within or external to their organizations. The iCalendar standard is a standard for calendar exchange. It is officially referred to as RFC 2445.

This standard was the result of years of work by engineers from Microsoft, IBM, and other organizations. The standards document is nearly 150 pages long and defines every component of calendar objects that can be exchanged between calendar system users. While RFC 2445 is very precise and complete, it is not the easiest reading unless you are having trouble sleeping. This paper shows how to create RFC 2445 calendar objects with simple SAS DATA step programs and send those objects to the people who need to take action at a specific time. The examples below include references to specific parts of RFC 2445 in the comments.

THE CALENDAR OBJECT

The RFC 2445 standard describes the calendar *object*. A calendar object can have several properties and multiple components. The following table shows the properties and components that can be contained within a calendar object. The second column refers to the section of the RFC 2445 standard document that defines that property or component.

Property Name	RFC 2445 Section	Description
prodid	4.7.3	Product Identifier (required)
version	4.7.4	Version Identifier (required)
calscale	4.7.1	Calendar Scale
method	4.7.2	Method Associated With Object
x-prop	4.8.8.1	Non-standard properties
Calendar Components (can contain one or more)		
vevent	4.6.1	Event Component
vtodo	4.6.2	To-Do Component
vjournal	4.6.3	Journal Component
vfreebusy	4.6.4	Free/Busy Component
vtimezone	4.6.5	Time Zone Component

Table 1. Properties and Components for Calendar Objects

This paper discusses only the event component.

Event components have properties and can also contain components. The following table shows the properties and components for the event component.

Property Name	RFC 2445 Section	Description
Descriptive Properties		
attach	4.8.1.1	Attachment
categories	4.8.1.2	Categories
class	4.8.1.3	Classification
comment	4.8.1.4	Comment
description	4.8.1.5	Description
geo	4.8.1.6	Geographic Position
location	4.8.1.7	Location
priority	4.8.1.9	Priority
resources	4.8.1.10	Resources
status	4.8.1.11	Overall Status
summary	4.8.1.12	Summary
Date/Time Properties		
dtend	4.8.2.2	Date/Time End
dtstart	4.8.2.4	Date/Time Start (required)
duration	4.8.2.5	Duration
transp	4.8.2.7	Time Transparency
Relationship Properties		
attendee	4.8.4.1	Attendee
contact	4.8.4.2	Contact
organizer	4.8.4.3	Organizer
recurid	4.8.4.4	Recurrence ID
related	4.8.4.5	Related To
url	4.8.4.6	Uniform Resource Locator
uid	4.8.4.7	Unique Identifier
Recurrence Properties		
exdate	4.8.5.1	Exception Date/Time
exrule	4.8.5.2	Exception Rule
rdate	4.8.5.3	Recurrence Date/Times
rrule	4.8.5.4	Recurrence Rule
Change Management Properties		
created	4.8.7.1	Date/Time Created
dtstamp	4.8.7.2	Date/Time Stamp
last-mod	4.8.7.3	Last Modified
seq	4.8.7.4	Sequence Number
Miscellaneous Properties		
x-prop	4.8.8.1	Non-Standard Properties

rstatus	4.8.8.2	Request Status
Event Components		
valarm	4.8.6	Alarm Component

Table 2. Properties and Components for Event Components

A SIMPLE EXAMPLE

To understand how iCalendar works, let's examine a very simple example. RFC 2445 shows the following invitation to a party on July 14, 1997, starting at 5:00 p.m. and ending early on the following day (those standards folks know how to celebrate):

```
BEGIN:VCALENDAR
VERSION:2.0
PRODID:-//hacksw/handcal//NONSGML v1.0//EN
BEGIN:VEVENT
DTSTART:19970714T170000Z
DTEND:19970715T035959Z
SUMMARY:Bastille Day Party
END:VEVENT
END:VCALENDAR
```

This calendar object defines two properties (VERSION and PRODID) and one event. The event has a starting and ending date and time, and a SUMMARY property describing the event.

WORKING WITH THE CALENDAR OBJECT

You can produce calendar objects by using simple DATA step programming with PUT statements.

The entire calendar object is enclosed by `BEGIN:VCALENDAR` and `END:VCALENDAR`. You can abstract this information into the following SAS macros:

```
%macro begin_vcalendar;
;* define Global variables used by other macros *;
length uuid_str $ 36 dt 8 outbuffer $ 200 outLine $ 75 Duration $ 32;
put "BEGIN:VCALENDAR";                        %* RFC 2445 4.4 ;
put "VERSION:2.0" ;                           %* RFC 2445 4.7.4;
put "PRODID:-//SAS Institute//Base SAS Example//EN"; %* RFC 2445 4.7.5 *;
%mend;

%macro end_vcalendar;
put "END:VCALENDAR";                         %* RFC 2445 4.4 *;
%mend;
```

The BEGIN_VCALENDAR macro also contains declarations for variables that are used by other macros. In addition to writing `BEGIN:VCALENDAR`, this macro also includes some properties that, while not required, can be added for documentation or debugging purposes. The VERSION property describes the version of the standard that should be used to interpret the calendar object; the version for RFC 2445 is 2.0. The PRODID property contains a string that identifies the application that wrote the calendar object.

The END_CALENDAR macro simply writes a single line, `END:VCALENDAR`. We might want to add error checking or other clean code to this. No additional calendar items can be written after `END:VCALENDAR`.

CREATING EVENTS

An event can be anything that has a scheduled date and time such as a meeting, birthday, vacation, or reminder to pay the bills. An event is created within a calendar object with the VEVENT component. Like the calendar object, the VEVENT component is contained within BEGIN and END lines:

```
%macro begin_vevent;
put "BEGIN:VEVENT";                    %* RFC 2445 4.6.1 *;
uuid_str = uuidgen(0);
```

```
put "UID:" uuid_str;                    %* RFC 2445 4.8.4.7 *;
%mend;

%macro end_vevent;
put "END:VEVENT";
%mend;
```

The BEGIN_VEVENT macro includes the UID property, which creates a unique identifying string that is used to match events with subsequent updates to the event. This example uses the UUIDGEN function to generate a unique 36-character string.

An event must have a starting date and time. It can also contain an ending date and time or duration. The datetime values must be in the ISO 8601 datetime format. Fortunately, formats that support ISO 8601 were added in SAS 9.2.

```
%macro dtstart(dt_val);
dt = &dt_val;
put "DTSTART:" dt b8601dt.;              /* RFC 2445 4.8.2.4 */
%mend;
%macro dtend(dt_val);
dt = &dt_val;
put "DTEND:" dt b8601dt.;                /* RFC 2445 4.8.2.2 */
%mend;
```

An alternative to DTEND is DURATION:

```
%macro put_duration(years=0,mon=0,day=0,hours=0,min=0,sec=0);
duration =  compress(
                 cat('P',PUT(&years,2.),'Y',PUT(&mon,2.),'M',PUT(&day,2.),'D',
                     'T',PUT(&Hours,2.),'H',PUT(&min,2.),'M',PUT(&sec,2.),'S'));

put "DURATION:" duration @;
%mend;
```

An event should also contain a SUMMARY property that describes the event. The SUMMARY property takes a text value. Each line of text must be terminated by a CRLF (Carriage Return Line Feed) character sequence and cannot be longer than 75 bytes. Long lines can be split by continuing on the next line, starting with a single space or tab character. Text can contain any printable ASCII characters except for these special characters: colon (**:**), semicolon (**;**), backslash (****), comma (**,**), and the quotation mark (**“**). These special characters can be inserted into text by preceding them with a backslash character. New lines can be inserted into text with the `\n` character sequence. The standard allows other encoding schemes to be used by specifying another ENCODING property such as BASE64. The PUT_TEXT macro writes text that conforms to these rules. The PUT_NL macro forces a new line in the text.

```
%macro put_text(pre,text);
outbuffer = prxchange('s/([,:\\;])/\\$1/',-1,&text);
outline = cat(&pre,substr(outbuffer,1,75-length(&pre)));
put outline;
outbuffer = substr(outbuffer,length(outline)+1);
do while(length(outbuffer) > 74);
   outline = cat(' ',substr(outbuffer,1,74));
   put outline;
   outbuffer = substr(outbuffer,75);
   end;
%mend;

%macro put_nl;
put ' \n';
%mend;
```

You can use PUT_TEXT and PUT_NL with the SUMMARY, LOCATION, and DESCRIPTION properties.

```
%macro SUMMARY(SUM_STR);
/* Text format defined in 4.3.11 */
%put_text("SUMMARY:",&SUM_STR);          /* RFC 2445 4.8.1.12 */
%mend;

%macro location(str);
%put_text("LOCATION:",&str);             /* RFC 2445 4.8.1.7 */
%mend;
```

```
%macro description(str);
%put_text("DESCRIPTION:",&str);           /* RFC 2445 4.8.1.5 */
%mend;
```

You can add a reminder to an event with the VALARM component. This component has many properties available. The following macro creates a reminder at a specified time before the event is scheduled:

```
%macro reminder(years=0,mons=0,days=0,hours=0,mins=0,secs=0);
put "BEGIN:VALARM";
PUT "ACTION:DISPLAY";
put "DESCRIPTION:REMINDER";
duration = cats('-P',PUT(&years,2.),'Y',PUT(&mons,2.),'M',PUT(&days,2.),'D',
               'T',PUT(&Hours,2.),'H',PUT(&mins,2.),'M',PUT(&secs,2.),'S');
put "TRIGGER:" duration;
PUT "END:VALARM";
%mend;
```

The property used by TRIGGER uses the ISO 8609 duration format.

Any other property defined by RFC 2445 can be sent to the output file using PUT statements or the PUT_TEXT and PUT_NL macros. If you use additional properties, you need to read the standard very carefully.

USING THE MACROS

Now that we have a set of macros that create the fundamental calendar components, we can begin to write programs that use them. Let's go back to the Bastille Day party invitation. We could send the invitation to our friends using the SAS SMTP e-mail facility:

```
filename icalfile "bastille.ics" termstr=crlf lrecl=73 recfm=v;
filename mailer email;
data _null_;
  file icalfile;
  %begin_VCALENDAR;
    %begin_vevent;
     %dtstart('14Jul2009:06:00PM'dt);
     %dtend('14Jul2009:11:00PM'dt);
     %summary("Bastille Day Party");
     %location("Paris, France");
     %description("Wear red, white and blue.”);
     %put_nl;
     %put_text(“ Drink red wine.”);
     %put_nl;
     %put_text(“ Watch BluRay.");
   %end_vevent;
 %end_vcalendar;
 run;
 data _null_;
file mailer subject="Party" to=("Thacher@sas.com") attach="bastille.ics";
put "We're having a party.  Please add this reminder to your calendar." /
    "Vive la France!";
run;
```

MORE EXAMPLES

Here's an example that creates a calendar of birthdays from a data source:

```
filename icalfile "birthdays.ics" termstr=crlf lrecl=73 recfm=v;
filename mailer email;
data _NULL_; set bdays end=done;
file bdays;
if _n_ = 1 then do;
   %begin_vcalendar;
   end;
%begin_vevent;
  %dtstart(dhms(bday,0,0,0));
  %dtend(dhms(bday,23,59,59));
```

```
    %summary(catt(name,"'s birthday."));
    %put_text("CATEGORIES:","BIRTHDAYS");
    put "PRIORITY:1";
    put "RRULE:FREQ=YEARLY";
    %reminder(days=1);
  %end_vevent;
  if done then do;
     %end_vcalendar;
     end;
  run;

  data _null_;
  file mailer subject="Birthdays" to=("Thacher@sas.com") attach="birthdays.ics";
  put "Here's the calendar with everyone's birthday." /
      "Now you have no excuse for neglecting birthdays.";
  run;
```

The example above uses a few new properties. CATEGORIES enables you to specify a set of categories that would be assigned to the event. PRIORITY takes a numeric value between 1 and 10, with 1 being the highest priority and 10 the lowest. RRULE is used to specify the repeat rule. In the birthday example, each birthday event repeats annually.

Many calendar programs take multiple VEVENT components and create a new calendar instead of adding the events to existing calendars.

Our last example uses calendar events to solve the problem of having your SAS SETINIT expire at an inconvenient time:

```
filename logfile temp;
options nonotes nodate nostimer ls=100;
title '';
proc printto log=logfile;run;
proc setinit;run;
proc printto;run;
options notes nodate stimer;

data setinit;
length sitename $ 40 sitenum $ 20 prodname $ 60;
retain sitename  sitenum instexpd daysgrace dayswarn birthday oscode ;
format    instexpd  date9. prodexpd date9.  birthday date9. ;
informat  instexpd prodexpd birthday date9. ;
infile logfile flowover;
input @'Site name:' sitename & $quote. /
      @'Site number:' sitenum $ /
      @'Expiration:' instexpd : date.
      @'Grace Period:' daysgrace /
      @'Warning Period:' dayswarn /
      @'System birthday:' birthday & date. /
      @'Operating System:' oscode $ ;
output;
put _all_;
stop;
run;

filename ical 'setinit.ics' recfm=v lrecl=75;
%macro Text_continue(text);
outbuffer = prxchange('s/([,:\\;])/\\$1/',-1,&text);
outline = substr(outbuffer,1,74);
put @2 outline;
outbuffer = substr(outbuffer,75);
do while(length(outbuffer) > 74);
   outline = substr(outbuffer,1,74);
   put @2 outline;
   outbuffer = substr(outbuffer,75);
   end;
%mend;
```

```
data _null_;
set setinit;
length Install_dir $ 100;
file ical;
install_dir =  sysget("SASROOT");
%begin_vcalendar;
  %begin_vevent;
   %dtstart(dhms(instexpd,0,0,0));
   %dtend(dhms(instexpd,23,59,59));
   put "PRIORITY:1";
   %summary("Setinit expires on &SYSHOSTNAME");
   %description("Contact your SAS site representative to get a new one.");
     %put_nl;
     %text_continue("&SYSVER is installed on &SYSHOSTNAME in ");
     %put_nl;
     %text_continue(install_dir);
  %reminder(days=30);
  %end_vevent;
%end_vcalendar;
run;

data _null_;
file mailer subject="SAS Setinit reminder"
           to=("siterep@mycompany.com") attach="setinit.ics";
put "Don't forget to get me the setinit this year" /
    "You remember how unhappy Jim was when he couldn't get his dashboard
reports";
run;
```

The program first sends the log from PROC SETINIT to a file. The second step extracts the system expiration date and other information, and writes this information to WORK.SETINIT. The next step creates a high-priority calendar event for the expiration date from the log file with a reminder 30 days in advance. The last step sends the event to the person who needs to act to keep everything running happily.

CONCLUSION

We have shown a small subset of the features of the iCalendar specification. More features are illustrated in the source code that is available for download. Other features in the standard that are not implemented can be added by careful reading of the documentation. Be warned that we have not tested this code on all calendar systems and there might be discrepancies between the different implementations. Test the code on your system before putting it into production.

REFERENCES

The entire text for RFC 2445 can be found at http://tools.ietf.org/html/rfc2445.

This paper and all source code are available on the SAS Presents Web site (http://support.sas.com/saspresents/).

CONTACT INFORMATION

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

Clarke Thacher
SAS Institute Inc.
SAS Campus Drive
Cary, NC 27513
E-mail: thacher@sas.com