Network Working Group
Internet Draft
<draft-ietf-calsch-ical-04.txt>
Expires Mary 1998

Frank Dawson, Lotus
Derik Stenerson, Microsoft

October 22, 1997

Internet Calendaring and Scheduling Core Object Specification (iCalendar)

Status of this Memo

Abstract

There is a clear need to provide and deploy interoperable calendaring and scheduling services for the Internet. Current group scheduling and Personal Information Management (PIM) products are being extended for use across the Internet, today, in proprietary ways. This memo has been defined to provide the a definition of a common format for openly exchanging calendaring and scheduling information across the Internet.

This memo is formatted as a registration for a MIME media type per [RFC 2048]. However, the format in this memo is equally applicable for use outside of a MIME message content type.

The proposed media type value is ''TEXT/CALENDAR''. This string would label a media type containing calendaring and scheduling information

encoded as text characters formatted in a manner outlined below.

This MIME media type provides a standard content type for capturing calendar event and to-do information. It also can be used to convey free/busy time information. The content type is suitable as a MIME message entity that can be transferred over MIME based email systems or using HTTP. In addition, the content type is useful as an object

for interactions between desktop applications using the operating system clipboard, drag/drop or file systems capabilities.

This memo is based on the earlier work of the vCalendar specification for the exchange of personal calendaring and scheduling information. In order to avoid confusion with this referenced work, this memo is to be known as the iCalendar specification. This memo is based on the calendaring and scheduling model defined in []. The document is also the basis for the calendaring and scheduling interoperability protocol defined in [ITIP].

This memo also includes the format for defining iCalendar object methods. An iCalendar object method is a set of usage constraints for the iCalendar object. For example, these methods might define scheduling messages that request an event be scheduled, reply to an event request, send a cancellation notice for an event, modify or replace the definition of an event, provide a counter proposal for an original event request, delegate an event request to another individual, request free or busy time, reply to a free or busy time request, or provide similar scheduling messages for a to-do or journal entry calendar component.

Table of Contents

1.

Introduction

The use of calendaring and scheduling has grown considerably in the last decade. Enterprise and inter-enterprise business has become dependent on rapid scheduling of events and actions using this information technology. However, the longer term growth of calendaring and scheduling, is currently limited by the lack of Internet standards for the message content types that are central to these groupware applications. This memo is intended to progress the level of interoperability possible between dissimilar calendaring and scheduling applications. This memo defines a MIME content type for exchanging electronic calendaring and scheduling information. The Internet Calendaring and Scheduling Core Object Specification, or iCalendar, allows for the capture and exchange of information normally stored within a calendaring and scheduling application; such as a Personal Information Manager or a Group Scheduling product.

The calendaring and scheduling model implemented by this memo is defined in the [ICMS].

The format is suitable as an exchange format between applications or systems. The format is defined in terms of a MIME content type. This will enable the object to be exchanged using several transports, including but not limited to SMTP, HTTP, a file system, desktop interactive protocols such as the use of a memory-based clipboard or drag/drop interactions, point-to-point asynchronous communication, wired-network transport, or some form of unwired transport such as infrared might also be used.

The definition of a calendaring and scheduling interoperability protocol is the subject of another memo [ITIP].

The memo also provides for the definition of iCalendar object methods that will map this content type to a set of messages for supporting calendaring and scheduling operations such as requesting, replying to, modifying, and canceling meetings or appointments, to-dos and journal entries. The iCalendar object methods can be used to define other calendaring and scheduling operations such a requesting for and replying with free/busy time data.

The memo also includes a formal grammar for the content type to aid in the implementation of parsers and to serve as the definitive reference when ambiguities or questions arise in interpreting the descriptive prose definition of the memo.

2. Basic Grammar and Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY" and "OPTIONAL" in this document are to be interopreted as described in [RFC 2119].

This memo makes use of both a descriptive prose and a more formal notation for defining the calendaring and scheduling format.

The notation used in this memo is the augmented BNF notation of [RFC 822]. Readers intending on implementing this format defined in this memo should be familiar with this notation in order to properly interpret the specifications of this memo.

All numeric and hexadecimal values used in this memo are given in decimal notation. All names of properties, property parameters, enumerated property values and property parameter values are case-insensitive. However, all other property values are case-sensitive, unless otherwise stated.

> Note: All indented editorial notes, such as this one, are intended to provide the reader with additional information that is not essential to the building of a conformant implementation of the specifications of this memo. The information is provided to highlight a particular feature or characteristic of the specifications.

The format for the iCalendar object is based on the syntax of the [MIME DIR] content type. While the iCalendar object is not a profile of the [MIME DIR] content type, it does reuse a number of the elements from the [MIME DIR] specification.

2.1 Formatting Conventions

The mechanisms defined in this memo are defined in propose. In order to refer to elements of the calendaring and scheduling model, core object or interoperability protocol defined in this memo, [ICMS] and [ITIP], some formatting conventions have been used. Calendaring and scheduling roles defined by [ICMS] are referred to in quoted-strings of text with the first character of each word in upper case. For example, "Organizer" refers to a role of a "Calendar User" within the scheduling protocol defined by [ITIP] Calendar components defined by this memo are referred to with capitalized, quoted-strings of text. All calendar components start with the letter "V". For example, "VEVENT" refers to the event calendar component, "VTODO" refers to the to-do calendar component and "VJOURNAL" refers to the daily journal calendar component. Scheduling methods defined by [ITIP] are referred to with capitalized, quoted-strings of text. For example, "REQUEST" refers to the method for requesting a scheduling calendar component be created or modified, "REPLY" refers to the method a recipient of a request uses to update their status with the "Organizer" of the calendar component.

The properties defined by this memo are referred to with capitalized, quoted-strings of text, followed by the word "property". For example, "ATTENDEE" property refers to the iCalendar property used to convey the calendar address of a calendar user. Property parameters defined by this memo are referred to with lower case, quoted-strings of text, followed by the word "parameter". For example, "value" parameter refers to the iCalendar property parameter used to override the default data type for a property value. Enumerated values defined by

this memo are referred to with capitalized text, either alone or followed by the word "value".

2.2 Related Memos

Implementers will need to be familiar with several other memos that, along with this memo, form a framework for Internet calendaring and scheduling standards. This memo, [ICAL], specifies a core specification of objects, data types, properties and property parameters.

[ICMS] - specifies a common terminology and abstract;

[ITIP] - specifies an interoperability protocol for scheduling between different implementations;

[IMIP] specifies an Internet email binding for [ITIP];

[IRIP] - specifies an Internet real time protocol binding for [ITIP].

This memo does not attempt to repeat the specification of concepts or definitions from these other memos. Where possible, references are made to the memo that provides for the specification of these concepts or definitions.

3. TEXT/CALENDAR Registration Information

The Calendaring and Scheduling Core Object Specification is intended for use as a MIME content type. However, the implementation of the memo is in no way limited solely as a MIME content type.

The following text is intended to register this memo as the MIME content type "text/calendar".

To: ietf-types@uninett.no

Subject: Registration of MIME content type text/calendar.

MIME media type name: text

MIME subtype name: calendar

Required parameters: method

The "method" parameter is used to convey the iCalendar object method to which the calendaring and scheduling information pertains. It also is an identifier for the set of properties that the iCalendar object will consist of. The parameter is to be used as a guide for applications interpreting the information contained within the body part. It should NOT be used to exclude or require particular pieces of information unless the identified method definition specifically calls for this behavior. Unless specifically forbidden by a particular method definition, a

text/calendar content type MAY contain any set of properties permitted by the Calendaring and Scheduling Core Object Specification.

The value for the "method" parameter is defined as follows:

```
method  = <Identifier for any IANA registered iCalendar
           object method>
```

Optional parameters: charset, component

The "charset" parameter is defined in [RFC 2046] for other body parts. It is used to identify the default character set used within the body part.

The "component" parameter conveys the type of iCalendar calendar component within the body part. If the iCalendar object contains more than one calendar component, then the components are specified as a comma-separated list of values.

The value for the "component" parameter is defined as follows:

```
component       = "VEVENT" / "VTODO" / "VJOURNAL" / "VFREEBUSY"
                / x-token / iana-comp

x-token = <The two characters "X-" or "x-" followed, with
           no intervening white space, by any atom, where
           atom is from section 3.3 of [RFC 822]>

iana-comp = <A publicly defined extension component,
             registered with IANA, as specified by this
             document>
```

Optional content header fields: Any header fields defined by [RFC 2045].

Encoding considerations: This MIME content type can contain 8bit characters, so the use of quoted-printable or base64 MIME content-transfer-encodings MAY be necessary when iCalendar objects are transferrred across protocols restricted to the 7bit repertoire. Note that each property in the content entity MAY also have special characters encoded using a BACKSLASH character (ASCII decimal 92) escapement technique. This means that content values MAY end up encoded twice.

Security considerations: SPOOFING - - In this memo, the "Organizer"

is the only person authorized to make changes to an existing "VEVENT", "VTODO", "VJOURNAL" calendar component and redistribute the updates to the "Attendees". An iCalendar object that maliciously changes or cancels an existing "VEVENT", "VTODO" or "VJOURNAL" or "VFREEBUSY" calendar component MAY be constructed by someone other than the "Organizer" and sent to the "Attendees". In addition in this memo, an "Attendee" of a "VEVENT", "VTODO", "VJOURNAL" calendar component is the only person authorized to

update any parameter associated with their "ATTENDEE" property and send it to the "Organizer". An iCalendar object that maliciously changes the "ATTENDEE" parameters MAY be constructed by someone other than the real "Attendee" and sent to the "Organizer".

PROCEDURAL ALARMS - - An iCalendar object can be created that contains a "VEVENT" and "VTODO" calendar component with an "VALARM" calendar components. The "VALARM" calendar component MAY be of type PROCEDURE and MAY have an attachment containing some sort of executable program. Implementations that incorporate these types of alarms are subject to any virus or malicious attack that MAY occur as a result of executing the attachment.

ATTACHMENTS - - An iCalendar object MAY include references to Uniform Resource Locators that MAY be programmed resources.

Implementers and users of this memo should be aware of the network security implications of accepting and parsing such information. In addition, the security considerations observed by implementations of electronic mail systems should be followed for this memo.

Interoperability considerations: This MIME content type is intended to define a common format for conveying calendaring and scheduling information between different systems. It is heavily based on the earlier [VCAL] industry specification.

Intended Usage: COMMON

Published specification: This memo.

Author/Change controllers:

Frank Dawson
6544 Battleford Drive
Raleigh, NC 27613-3502
919-676-9515 (Telephone)

919-676-9564 (Data/Facsimile)
Frank_Dawson@Lotus.com (Internet Mail)

Derik Stenerson
One Microsoft Way
Redmond, WA 98052-6399
425-936-5522 (Telephone)
425-936-7329 (Facsimile)
deriks@microsoft.com (Internet Mail)

4. iCalendar Object Specification

The following sections define the details of a Calendaring and Scheduling Core Object Specification. This information is intended to be an integral part of the MIME content type registration. In addition, this information MAY be used independent of such content

registration. In particular, this memo has direct applicability for use as a calendaring and scheduling exchange format in file-, memory- or network-based transport mechanisms.

4.1 Content Considerations

The iCalendar object consists of lines of text. This section defines how the content lines MUST be formatted.

4.1.1 Content Lines

The iCalendar object consists of individual lines of text, delimited by a line break, which is a CRLF sequence (ASCII decimal 13, followed by ASCII decimal 10). Line of text should not be longer than 76 characters, excluding the line break.

Long lines of text can be split into a multiple line representations using a line "folding" technique. That is, a long line MAY be split at any point by inserting a CRLF immediately followed by a single LWSP character (i.e., SPACE, ASCII decimal 32 or HTAB, ASCII decimal 9). Any sequence of CRLF followed immediately by a single LWSP character is ignored (i.e., removed) when processing the content type.

For example the line:

```
DESCRIPTION:This is a long description that exists on a long line.
```

Can be represented as:

```
DESCRIPTION:This is a long description
  that exists on a long line.
```

The process of moving from this folded multiple line representation to its single line representation is called "unfolding". Unfolding is accomplished by removing the CRLF character and the LWSP character that immediately follows.

An intentional formatted text line break MAY only be included in a property value by representing the line break with the character sequence of BACKSLASH (ASCII decimal 92), followed by a LATIN SMALL LETTER N (ASCII decimal 110) or a LATIN CAPITAL LETTER N (ASCII decimal 78), that is "\n" or "\N".

For example a multiple line "DESCRIPTION" property value of:

```
Project XYZ Final Review
Conference Room - 3B
Come Prepared.
```

Could be represented as:


```
DESCRIPTION:Project_XYZ Final_Review\n
 Conference Room - 3B\nCome_Prepared.
```

The content information associated with an iCalendar object is formatted using a syntax similar to that defined by [MIME DIR]. That is, the content information consists of one or more CRLF-separated content lines as defined by the following notation:

```
contentline = name [";" [ws] paramlist] ":" [ws] value CRLF
;Folding permitted on content lines.
;Lines should be less than 76 characters, excluding CRLF.

LWSP       = SPACE / HTAB
```

```
SPACE      = <ASCII Decimal 32>

HTAB       = <ASCII Decimal 9>

ws         = SPACE / HTAB

name       = iana-name / x-name    ;An iCalendar property

iana-name  = <One of the properties defined by this memo or
              an IANA registered property, as defined by the
              registration process in this memo.>

x-name     = <The two characters "X-" or "x-" followed, with no
              intervening white space, by any atom. A non-standard
              property name.>

value      = boolean / cal-address / date / date-time / duration
           / float / integer / period / recur / text / time / url
           / utc-offset / x-token

iana-value = <One of the property values defined by this memo
              or an IANA registered property value as defined by
              the registration process in this memo.>
```

4.1.2 List and Field Separators

List of values MAY be specified for property values or property parameter values. Each value in a list of values MUST be separated by a COMMA character (ASCII decimal 44). A COMMA character in a property value or a property parameter value MUST be escaped with a BACKSLASH character (ASCII decimal 92).

Some property values are defined in terms of multiple components. These structured property values MUST have their components separated by a SEMICOLON character (ASCII decimal 59). A SEMICOLON character in a property value MUST be escaped with a BACKSLASH character (ASCII decimal 92).

Lists of property parameters MAY be specified for a property. Each property parameter in a list of property parameters MUST be separated by a SEMICOLON character (ASCII decimal 59). A SEMICOLON character in

a property parameter value MUST be escaped with a BACKSLASH character (ASCII decimal 92).

A COLON character (ASCII decimal 58) in a property parameter value MUST be escaped with a BACKSLASH character (ASCII decimal 92). A COLON character in a property value does not need to be escaped with a BACKSLASH character.

A BACKSLASH character (ASCII decimal 92) in a property value or a property parameter value MUST be escaped with another BACKSLASH character.

For example, in the following properties a SEMICOLON is used to separate property parameters and property value fields. A COMMA is used to separate values.

```
ATTENDEE;RSVP=TRUE;ROLE=ATTENDEE:J.Smith <jsmith@host.com>

RDATE;VALUE=DATE:19970304,19970504,19970704,19970904
```

4.1.3 Multiple Values

Each property defined in the iCalendar object MAY have multiple values, if allowed in the definition of the specific property. The general rule for encoding multi-valued items is to simply create a new content line for each value; including the property name. However, it should be noted that some properties support encoding multiple values in a single property by separating the values with a COMMA character (ASCII decimal 44).

4.1.4 Binary Content

There is no support for including binary content information, inline, within an iCalendar object. Binary content information MUST be referenced by a uniform resource locator (URL) type of property value.

The following example specifies an "ATTACH" property with a reference to an attachment consisting of a binary object:

```
ATTACH:ftp://xyz.com/public/quarterly-report.doc
```

4.1.5 Property Parameters

A property MAY have additional attributes associated with it. These "property parameters" contain meta information about the property or the property value. Property parameters MAY be used to specify the location of an alternate text representation for a property value, the content encoding used on a property value, the language of a text

property value or thedata type of the property value.

Property parameter values that contain the COLON, SEMICOLON, COMMA or BACKSLASH character separators MUST be specified as quoted-string text values. For example:

```
DESCRIPTION;ALTREP="http://www.wiz.org":The Fall'98 Wild Wizards
  Conference - - Las Vegas, NV, USA
```

Property parameters are defined by the following notation:

```
(  = parameter *(";" [ws] parameter)

parameter  = altrepparm            ;Alterate text representation
           / languageparm          ;Text language
           / valuetypeparm         ;Property value data type
           / [parmtype "="] parmvalues ;Parameter extensions

parmtype   = x-token / iana-ptype

iana-ptype = <A publicly defined extension parameter type,
              registered with IANA, as specified in this memo>

parmvalues = parmvalue / parmvalue *("," [ws] parmvalue)

parmvalue  = x-name / iana-pvalue

iana-pvalue = <A publicly defined extension parameter value,
               registered with IANA, as specified in this memo>
```

4.1.6 Alternate Text Representation

The "ALTREP" property parameter is an OPTIONAL property parameter. It specifies the URL that points to an alternate representation for a textual property value. The property MUST include a value that reflects the default representation. This property parameter MAY include multiple values, separated by the COMMA character (ASCII decimal 44). The property parameter MAY only be specified in the "COMMENT", "CONTACT", "DESCRIPTION", "LOCATION" and "SUMMARY" properties.

For example:

```
DESCRIPTION;ALTREP="CID:<part3.msg.970415T083000@host.com>":Project
```

XYZ Review Meeting will include the following agenda items: (a) Market Overview, (b) Finances, (c) Project Management

The "ALTREP" property parameter value might point to a "text/html" content portion.

Content-Type:text/html
Content-Id:<part3.msg.970415T083000@host.com>

Project XYZ Review Meeting will include the following agenda items:
Market Overview
Finances
Project Management

The "ALTREP" property parameter is defined by the following notation:

altrepparm = "altrep" "=" urltype

urltype = <quoted-string text URL value>

4.1.7 Character Set

There is not a property parameter to declare the character set used in a property value. The default character set for an iCalendar object is [UTF-8].

The "charset" Content-Type parameter MAY be used in MIME transports to specify any other IANA registered character set.

4.1.8 Language

The "LANGUAGE" property parameter MAY be used to identify the language used in text values. The value of the "language" property parameter is that defined in [RFC 1766].

Note: For transport in a MIME entity, the Content-Language header field MAY be used to set the default language for the entire body part.

The "LANGUAGE" property parameter is defined by the following notation:

```
  languageparm =     "language" "=" language

  language = <Text identifying a language, as defined in [RFC 1766]>
```

4.1.9 Value Data Types

The "VALUE" property parameter is an OPTIONAL property parameter. It is used to identify the data type and format of the property value. The values of a property MUST only be of a single data type. For example, a "RDATE" property can not have a combination of DATE-TIME and TIME values.

The "VALUE" property parameter is defined by the following notation:

```
  valuetypeparm = "value" "=" valuetype

  valuetype  = "boolean"
             / "cal-address"
             / "date"
             / "date-time"
             / "duration"
```



```
             / "float"
             / "integer"
             / "period"
             / "recur"
             / "text"
             / "time"
             / "url"
             / "utc-offset"
             / x-token
             / iana-value

  iana-value = <A publicly defined extension value type, registered
               with IANA, as specified by this memo>
```

The following data types are defined by this memo.

4.1.9.1 Boolean

The "BOOLEAN" data type is used to identify properties that contain

either a "true" or a "false" boolean value. These values are case insensitive. The data type is defined by the following notation:

```
boolean    = "TRUE" / "FALSE"
```

For example, any of the following are equivalent:

```
TRUE
true
TrUe
```

4.1.9.2 Calendar User Address

The "CAL-ADDRESS" data type is used to identify properties that contain an address of a calendar user. The phrase component of the address MAY be used to match an unknown address with an otherwise known individual, group, or resource. The data type is as defined by the following notation:

```
cal-address        = addr-spec / phrase "<" addr-spec ">"

addr-spec  = local-part "@" domain         ;RFC 822 style address
local-part = WORD *("." WORD)
domain     = domain-ref *("." domain-ref)
domain-ref = ATOM

phrase     = 1*WORD
WORD       = ATOM / quoted-string
quoted-string = <"> *(qtext/quoted-pair) <"> ; Regular qtext or
                                             ;   quoted chars.
qtext      = <any CHAR excepting <">,        ; => MAY be folded
             "\" & CR, and including linear-
             white-space>
quoted-pair ="\" CHAR                        ; MAY quote any char
```



```
CHAR       = <any a character from the
             selected character set>
ATOM       = 1*<any CHAR except specials,
             SPACE and CTLs>
```

4.1.9.3 Date

The "DATE" data type is used to identify values that contain a calendar date. The format is expressed as the [ISO 8601] complete representation, basic format for a calendar date. The text format specifies a four-digit year, two-digit month, and two-digit day of the month. There are no separator characters between the year, month and day component text. The data type is defined by the following notation:

```
DIGIT      =<any ASCII decimal digit>      ;0-9

date-fullyear      = 4DIGIT
date-month         = 2DIGIT        ;01-12
date-mday          = 2DIGIT        ;01-28, 01-29, 01-30, 01-31
                                   ;based on month/year

date               = date-fullyear date-month date-mday
```

For example, the following represents July 14, 1997:

```
19970714
```

4.1.9.4 Date-Time

The "DATE-TIME" data type is used to identify values that contain a precise calendar date and time of day. The format is expressed as the [ISO 8601] complete representation, basic format for a calendar date and time of day. The text format is a concatenation of the "date", followed by the LATIN CAPITAL LETTER T character (ASCII decimal 84) time designator, followed by the "time" format defined above. The data type is defined by the following notation:

```
date-time  = date "T" time ;As specified above in date and time
```

The following represents July 14, 1997, at 1:30 PM in UTC and the equivalent time in New York (four hours behind UTC in DST), expressed as a local time and local time with UTC offset:

```
19970714T133000Z
19970714T083000
19970714T083000-0400
```

4.1.9.5 Duration

The "DURATION" data type is used to identify properties that contain a duration of time. The format is expressed as the [ISO 8601] basic format for the duration of time. The format can represent durations

in terms of years, months, days, hours, minutes, and seconds. The data type is defined by the following notation:

```
DIGIT      =<any ASCII decimal digit>      ;0-9

dur-second = 1*DIGIT "S"
dur-minute = 1*DIGIT "M" [dur-second]
dur-hour   = 1*DIGIT "H" [dur-minute]
dur-time   = "T" (dur-hour / dur-minute / dur-second)

dur-week   = 1*DIGIT "W"
dur-day    = 1*DIGIT "D"
dur-month  = 1*DIGIT "M" [dur-day]
dur-year   = 1*DIGIT "Y" [dur-month]
dur-date   = (dur-day / dur-month / dur-year) [dur-time]

duration   = "P" (dur-date / dur-time / dur-week)
```

For example, a duration of 10 years, 3 months, 15 days, 5 hours, 30 minutes and 20 seconds would be:

```
P10Y3M15DT5H30M20S
```

4.1.9.6 Float

The "FLOAT" data type is used to identify properties that contain a real value number value. If the property permits, multiple "float" values MAY be specified using a COMMA character (ASCII decimal 44) separator character. The data type is defined by the following notation:

```
DIGIT      =<any ASCII decimal digit>      ;0-9

float      = ["+" / "-"] *DIGIT ["." *DIGIT]
```

For example:

```
1000000.0000001
1.333
-3.14
```

4.1.9.7 Integer

The "INTEGER" data type is used to identify properties that contain a signed integer value. The valid range for "integer" is -2147483648 to

2147483647. If the sign is not specified, then the value is assumed to be positive. If the property permits, multiple "integer" values MAY be specified using a COMMA character (ASCII decimal 44) separator character. The data type is defined by the following notation:

```
DIGIT      =<any ASCII decimal digit>      ;0-9

integer    = ["+" / "-"] *DIGIT
```


For example:

```
1234567890
-1234567890
+1234567890
432109876
```

4.1.9.8 Period of Time

The "PERIOD" data type is used to identify values that contain a precise period of time. There are two forms of a period of time.

A period of time MAY be identified by it start and its end. This format is expressed as the [ISO 8601] complete representation, basic format for "DATE-TIME" start of the period, followed by a SOLIDUS character (ASCII decimal 47), followed by the "DATE-TIME" of the end of the period. For example, the period starting at 10 AM in Seattle (eight hours behind UTC) on January 1, 1997 and ending at 11 PM in Seattle on January 1, 1997 would be:

```
19970101T100000-0800/19970101T230000-0800
```

A period of time MAY also be defined by a start and a duration of time. The format is expressed as the [ISO 8601] complete representation, basic format for the "DATE-TIME" start of the period, followed by a SOLIDUS character (ASCII decimal 47), followed by the [ISO 8601] basic format for "DURATION" of the period. For example, the period start at 10 AM in Seattle (eight hours behind UTC) on January 1, 1997 and lasting 5 hours and 30 minutes would be:

```
19970101T100000-0800/PT5H30M
```

The data type is defined by the following notation:

```
period-explicit = date-time "/" date-time
;ISO 8601 complete representation basic format for a period of time
;consisting of a start and end. The start MUST be before the end.

period-start = date-time "/" duration
;ISO 8601 complete representation basic format for a period of time
;consisting of a start and duration of time.

period     = period-explicit / period-start
```

4.1.9.9 Recurrence Rule

The "RECUR" data type is used to identify properties that contain a recurrence rule specification. The data type is a structured value consisting of a list of one or more recurrence grammar components. Each component is defined by a NAME=VALUE pair. The components are separated from each other by the SEMICOLON character (ASCII decimal 59). The components are not ordered in any particular sequence. Individual components MAY only be specified once.

The FREQ component identifies the type of recurrence rule. This component MUST be specified in the recurrence rule. Valid values include MINUTELY, to specify repeating events based on an interval of a minute or more; HOURLY, to specify repeating events based on an interval of an hour or more; DAILY, to specify repeating events based on an interval of a day or more; WEEKLY, to specify repeating events based on an interval of a week or more; MONTHLY, to specify repeating events based on an interval of a month or more; and YEARLY, to specify repeating events based on an interval of a year or more.

The INTERVAL component contains a positive integer representing how often the recurrence rule repeats. The default value is "1" or every minute for a MINUTELY rule, every hour for a HOURLY rule, every day for a DAILY rule, every week for a WEEKLY rule, every month for a MONTHLY rule and every year for a YEARLY rule.

The UNTIL component defines a date-time value which bounds the recurrence rule in an inclusive manner. If the value specified by UNTIL is synchronized with the specified recurrence, this date-time becomes the last instance of the recurrence. If not present, and the COUNT component is also not present, the RRULE is considered to repeat forever.

The COUNT component defines the number of occurrences at which to range-bound the recurrence. This component is ignored if the "UNTIL" component is also present.

The BYMINUTE component specifies a COMMA character (ASCII decimal 44) separated list of minutes within an hour. Valid values are 0 to 60. Zero is the beginning of the hour and 60 is the beginning of the next hour.

The BYHOUR component specifies a COMMA character (ASCII decimal 44) separated list of hours of the day. Valid values are 0 to 24. Zero is the start of the day and 24 is the start of the next day.

The BYDAY component specifies a COMMA character (ASCII decimal 44) separated list of days of the week; MO, indicates Monday; TU, indicates Tuesday; WE, indicates Wednesday; TH, indicates Thursday; FR, indicates Friday; SA, indicates Saturday; SU, indicates Sunday.

Each BYDAY values MAY also be preceded by a positive (+n) or negative (-n) integer. If present, this indicates the nth occurrence of the specific day within the MONTHLY or YEARLY RRULE. For example, within a MONTHLY rule, +1MO (or simply 1MO) represents the first Monday within the month, whereas -1MO represents the last Monday of the month. If an integer modifier is not present, it means all days of this type within the specified frequency. For example, within a MONTHLY rule, MO represents all Mondays within the month.

The BYMONTHDAY component specifies a COMMA character (ASCII decimal 44) separated list of days of the month. Valid values are 1 to 31 or -31 to -1.

Each BYMONTHDAY value MAY be preceeded by a postive (+n) or negative (-n) integer. If present, this indicates the nth occurrence of the specific day of the month within the MONTHLY rule. If an integer modifier is not present, it means all days of this type within the specified frequency. For example, within a MONTHLY rule, -10 represents the tenth to the last day of the month.

The BYYEARDAY component specifies a COMMA character (ASCII decimal 44) separated list of days of the year. Valid values are 1 to 366 or -366 to -1. For example, -1 represents the last day of the year (December 31st).

The BYWEEKNO component specifies a comma-separated list of weeks of

the year. Valid values are 1 to 53. This corresponds to weeks according to week numbering as defined in [ISO 8601]. That is, a week as "A seven day period within a calendar year, starting on a Monday and identified by its ordinal number within the year; the first calendar week of the year is the one that includes the first Thursday of that year." This component is only valid for YEARLY rules.

Each BYWEEKNO value MAY be preceded by a positive (+n) or negative (-n) integer. If present, this indicates the nth occurrence of the specific week of the year within the YEARLY rule. If an integer modifier is not present, it means all days of this type within the specified frequency. For example, within a YEARLY rule, 3 represents the third week of the year.

The BYMONTH component specifies a comma separated list of months of the year. Valid values are 1 to 12.

The WKST component specifies the day on which the workweek starts. Valid values are MO, TU, WE, TH, FR, SA and SU. This is significant when a WEEKLY RRULE has an interval greater than 1, and a BYDAY component is specified. The default value is MO.

The BYSETPOS component specifies a COMMA character (ASCII decimal 44) separated list of values which corresponds to the nth occurrence within the set of events specified by the rule. Valid values are 1 to 366 or -366 to -1. It MUST only be used in conjunction with another Byxxx component. For example "the last work day of the month" could be represented as:

```
RRULE:FREQ=MONTHLY;BYDAY=MO,TU,WE,TH,FR;BYSETPOS=-1
```

If BYxxx component values are found which are beyond the available scope (ie, BYMONTHDAY=-30 in February), they are simply ignored

Information, not contained in the rule, necessary to determine the various recurrence instance start time and dates are derived from the Start Time (DTSTART) entry attribute. For example, "FREQ=YEARLY;BYMONTH=1" doesn t specify a specific day within the month or a time. This information would be the same as what is specified for DTSTART.

BYxxx components modify the recurrence in some manner. BYxxx components for a period of time which is the same or greater than the frequency generally reduce or limit the number of occurrences of the

recurrence generated. For example, "FREQ=DAILY;BYMONTH=1" reduces the number of recurrence instances from all days (if BYMONTH tag is not present) to all days in January. BYxxx components for a period of time less than the frequency generally increase or expand the number of occurrences of the recurrence. For example, "FREQ=YEARLY;BYMONTH=1,2" increases the number of days within the yearly recurrence set from 1 (if BYMONTH tag is not present) to 2.

If only one BYxxx component is specified in the recurrence rule, the list of "n" unique values would cause "n" occurrences of the recurrence within each specified frequency interval, where each unique list value is substituted in the appropriate date position within DTSTART for each such occurrence.

If multiple BYxxx components are specified, then the list of "n" unique values for each lower frequency BYxxx components is applied to the list of "n" unique values for higher frequency BYxxx components. This process will not always increase the set of occurrences. If a higher component is inconsistent with what was generated for lower components, it would reduce the set. The ordering of BYxxx components from lower frequency to higher frequency is as follows: BYMINUTE, BYHOUR, BYDAY, BYMONTHDAY, BYYEARDAY, BYWEEKNO, BYMONTH, BYSETPOS.

Here is an example of evaluating multiple BYxxx components.

```
  "FREQ=YEARLY;INTERVAL=2;BYMONTH=1;BYDAY=SU;BYHOUR=8,9;BYMINUTE=30"
```

would first apply the "BYMINUTE=30" To "BYHOUR=8,9" to arrive at "every 8:30AM and 9:30AM". This in turn would be applied to "BYDAY=SU" to arrive at "every Sunday at 8:30AM and 9:30AM". This would be applied to "BYMONTH=1" to arrive at "every Sunday in January at 8:30AM and 9:30AM". Considering the FREQUENCY and INTERVAL, this would become "Every Sunday in January at 8:30AM and 9:30AM, every other year". If the BYMINUTE, BYDAY, BYMONTHDAY, BYYEARDAY, BYHOUR or BYMONTH component was missing, the appropriate mintues, hour, day or month would have been retrieved from DTSTART.

The data type is defined by the following notation:

```
  recur      = "FREQ"=freq ";"
             [("UNTIL" "=" enddate ";") / ("COUNT" "=" digits ";")]
             ["INTERVAL" "=" digits ";"]
             ["BYMINUTE" "=" byminlist ";"]
             ["BYHOUR" "=" byhrlist ";"]
             ["BYDAY" "=" bywdaylist ";"]
             ["BYMONTHDAY" "=" bymodaylist ";"]
             ["BYYEARDAY" "=" byyrdaylist ";"]
             ["BYWEEKNO" "=" bywknolist ";"]
             ["BYMONTH" "=" bymolist ";"]
             ["BYSETPOS" "=" bysplist ";"]
             ["WKST" "=" weekday ";")]
```

```
               *("X-" word "=" word) ";"
        ;Individual components MAY only be specified once.
        ;Rule components need not be specified in particular any order.

     freq       = "MINUTELY" / "HOURLY" / "DAILY" / "WEEKLY" / "YEARLY"

     enddate    = date          ;A UTC value

     digits     = 1*DIGIT

     DIGIT      =<any ASCII decimal digit>      ;0-9

     byminlist  = minutes / ( minutes *("," minutes) )

     minutes    = 1*2digits     ;0 to 60

     byhrlist   = hour / ( hour *("," hour) )

     hour       = 1*2 digits    ;0 to 24

     bywdaylist = weekdaynum / ( weekdaynum *("," weekdaynum) )

     weekdaynum = [([plus] ordwk / minus ordwk)] weekday

     plus       = "+"

     minus      = "-"

     ordwk      = 1*2digits     ;1 to 53

     weekday    = "SU" / "MO" / "TU" / "WE" / "TH" / "FR" / "SA"
     ;Corresponding to SUNDAY, MONDAY, TUESDAY, WEDNESDAY, THURSDAY,
     ;FRIDAY, SATURDAY and SUNDAY days of the week.

     bymodaylist = monthdaynum / ( monthdaynum *("," monthdaynum) )

     monthdaynum = ([plus] ordmoday) / (minus ordmoday)

     ordmoday   = 1*2digits     ;1 to 31

     byyrdaylist = yeardaynum / ( yeardaynum *("," yeardaynum) )

     yeardaynum = ([plus] ordyrday) / (minus ordyrday)

     ordyrday   = 1*3digits     ;1 to 366
```

```
bywknolist = weeknum / ( weeknum *("," weeknum) )

weeknum    = ([plus] ordwk) / (minus ordwk)

bymolist   = monthnum / ( monthnum *("," monthnum) )

monthnum   = 1*2digits     ;1 to 12
```



```
bysplist   = setposday / ( setposday *("," setposday) )

setposday  = yeardaynum
```

For example, the following is a rule which specifies 10 meetings which occur every other day:

```
FREQ=DAILY;COUNT=10;INTERVAL=2
```

There are other examples specified in the "RRULE" specification.

4.1.9.10 Text

The "TEXT" data type is used to identify values that contain human readable text. The character set and language in which the text is represented is controlled by the "LANGUAGE" property parameters. The data type is defined by the following notation:

```
text       = <Any character in the selected character set, but
             not including CRLF>
```

4.1.9.11 Time

The "TIME" data type is used to identify values that contain a time of day. The format is expressed as the [ISO 8601] complete representation, basic format for a time of day. The text format consists of a two-digit 24-hour of the day (i.e., values 0-23), two-digit minute in the hour (i.e., values 0-59), and two-digit seconds in the minute (i.e., values 0-59). If seconds of the minute are not supported by an implementation, then a value of "00" should be specified for the seconds component. Fractions of an hour, minute or second are not supported by this format. This format is used to represent local time, local time with UTC offset and UTC time. UTC time is identified by a LATIN CAPITAL LETTER Z suffix character

(ASCII decimal 90), the UTC designator, appended to the time. The local time with UTC offset is expressed as a local time, suffixed with the signed offset from UTC. The UTC offset is express as the 2-digit hours and 2-digit minutes difference from UTC. It is expressed as positive, with an OPTIONAL leading PLUS SIGN character (ASCII decimal 43), if the local time is ahead of UTC. It is expressed as a negative, with a leading HYPEN-MINUS character (ASCII decimal 45), if the local time is behind UTC. Local time has neither the UTC designator nor the UTC offset suffix text. The data type is defined by the following notation:

```
DIGIT      =<any ASCII decimal digit>      ;0-9

time-hour          = 2DIGIT        ;00-23
time-minute        = 2DIGIT        ;00-59
time-second        = 2DIGIT        ;00-59
time-numzone       = ("+" / "-") time-hour time-minute
time-zone          = "Z" / time-numzone
```



```
time               = time-hour time-minute time-second [time-zone]
```

For example, the following represents 8:30 AM in New York, five hours behind UTC, in local time and local time with UTC offset. In addition, 1:30 PM in UTC is illustrated:

```
083000
083000-0500
133000Z
```

There are cases when a floating time is intended within a property value. For example, an event MAY be defined that indicates that an individual will be busy from 11:00 AM to 1:00 PM every day. In these cases, a local time MAY be specified. The recipient of an iCalendar object with a property value consisting of a local time, without any relative time zone information, should interpret the value as being fixed to the recipient's locale and time zone. In most cases, a fixed time is desired. To properly communicate a fixed time in a property value, either UTC, local time with UTC offset, or local time with a "VTIMEZONE" calendar component MUST be specified.

4.1.9.12 URL

The "URL" data type is used to identify values that contain a uniform resource locator (URL) type of reference to the property value. This data type might be used to reference binary information, for values that are large, or otherwise undesirable to include directly in the iCalendar object.

The URL value formats in RFC 1738, RFC 2111 and any other IETF registered value format MAY be specified.

The data type is defined by the following notation:

```
url        = <As defined by any IETF RFC>
```

Any IANA registered URL type MAY be used. These include, but are not limited to, those for FTP and HTTP protocols, file access, content identifier and message identifier.

For example, the following is an URL for a local file:

```
file:///my-report.txt
```

4.1.9.13 UTC Offset

The "UTC-OFFSET" data type is used to identify properties that contain an offset from UTC to local time. The data type is defined by the following notation:

```
utc-offset = time-numzone  ;As defined above in time data type
```


For example, the following are UTC offsets are given for standard time for New York (five hours behind UTC) and Geneva (one hour ahead of UTC):

```
-0500

+0100
```

4.2 iCalendar object

The Calendaring and Scheduling Core Object is a collection of calendaring and scheduling information. Typically, this information

will consist of a single iCalendar object. However, multiple iCalendar objects MAY be sequentially, grouped together. The first line and last line of the iCalendar object MUST contain a pair of iCalendar object delimiter strings. The syntax for an iCalendar object is as follows:

```
icalobject = "BEGIN" ":" [ws] "VCALENDAR" CRLF
             icalbody
             "END" ":" [ws] "VCALENDAR" CRLF [icalobject]
```

The following is a simple example of an iCalendar object:

```
BEGIN:VCALENDAR
VERSION:2.0
PRODID:-//hacksw/handcal//NONSGML v1.0//EN
BEGIN:VEVENT
DTSTART:19970714T120000-0500
DTEND:19970714T235959-0500
SUMMARY:Bastille Day Party
END:VEVENT
END:VCALENDAR
```

4.3 Property

A property is the definition of an individual attribute describing a calendar property or a calendar component. A property takes the following form:

```
property   = propname [";" [ws] parmlist] ":" [ws] value CRLF

propname   = <any properties defined in this memo>
           / iana-prop / x-token

x-token    = <The two characters "X-" or "x-" followed, with no
              intervening white space, by any atom>

iana-prop  = <A publicly defined extension property, registered
              with IANA, as specified by this memo>
```

The following is an example of a property:


```
DTSTART:19960415T083000-05:00
```

This memo places no imposed ordering of properties within an iCalendar object.

Property names, parameter names and parameter values (i.e., everything to the left of the ":" on a line) are case insensitive. For example, the property name "DUE" is the same as "due" and "Due".

4.4 Calendar Components

The body of the iCalendar object consists of a sequence of calendar properties and one or more calendar components. The calendar properties are attributes that apply to the calendar as a whole. The calendar components are collections of properties that with a particular calendar semantic. For example, the calendar component MAY specify a an event, a to-do, journal entry, time zone information, or free/busy time information, or alarm.

The body of the iCalenar Object is defined by the following notation:

```
icalbody   = calprops 1*component

calprops   = [calscale] prodid method [source] [name] version

component  = 1*(eventc / todoc / journalc / freebusyc /
           / timezonec)
```

4.4.1 Event Component

A "VEVENT" calendar component is a grouping of component properties and an OPTIONAL "VALARM" calendar component that represent a scheduled amount of time on a calendar. For example, it MAY be an activity; such as a one-hour, department meeting from 8:00 AM to 9:00 AM, tomorrow. Generally, these events will take up time on an individual calendar. Hence, the event will appear as an opaque interval in a search for busy time. Alternately, the event MAY have its Time Transparency set to "TRANSPARENT" in order to prevent blocking of the event in searches for busy time.

The "VEVENT" is also the calendar component used to specify an anniversary or daily reminder within a calendar. These events have a start time but no end time. The start time MAY also be specified as a DATE value data type, instead of the default DATE-TIME.

A "VEVENT" calendar component is defined by the following notation:

```
eventc     = "BEGIN" ":" [ws] "VEVENT" CRLF
             eventprop *alarmc
             "END" ":" [ws] "VEVENT" CRLF
```

```
     eventprop  = *attach *attendee *categories [class] *comment
                  *contact [created] [description] [dtend / duration]
```



```
                  dtstart *exdate *exrule [geo] [last-mod] [location]
                  [priority] [rstatus] *related *resources *rdate
                  *rrule dtstamp [seq] [status] summary [transp] uid
                  *url [recurid]
```

The "VEVENT" calendar component can not be nested within another calendar component. The "VEVENT" calendar components MAY be related to each other or to a "VTODO" or "VJOURNAL" calendar component with the "RELATED-TO" property.

The following is an example of the "VEVENT" calendar component used to represent a meeting that will also be opaque to searches for busy time:

```
  BEGIN:VEVENT
  UID:19970901T130000Z-123401@host.com
  DTSTAMP:19970901T1300Z
  DTSTART:19970903T083000-0800
  DTEND:19970903T110000-0800
  SUMMARY:Annual Employee Review
  CLASS:PRIVATE
  CATEGORIES:BUSINESS,HUMAN RESOURCES
  END:VEVENT
```

The following is an example of the "VEVENT" calendar component used to represent a reminder that will not be opaque, but rather transparent, to searches for busy time:

```
  BEGIN:VEVENT
  UID:19970901T130000Z-123402@host.com
  DTSTAMP:19970901T1300Z DTSTART:19970401T083000-0800
  DTEND:19970401T170000-0800
  SUMMARY:Laurel is in sensitivity awareness class.
  CLASS:PUBLIC
  CATEGORIES:BUSINESS,HUMAN RESOURCES
  TRANSP:TRANSPARENT
  END:VEVENT
```

The following is an example of the "VEVENT" calendar component used to represent an anniversary that will occur annually. Since it takes up no time, it will not appear as opaque in a search for busy time;

no matter what the value of the "TRANSP" property indicates:

```
BEGIN:VEVENT
UID:19970901T130000Z-123403@host.com
DTSTAMP:19970901T1300Z
DTSTART:19971102
SUMMARY:Our Blissful Anniversary
CLASS:CONFIDENTIAL
CATEGORIES:ANNIVERSARY,PERSONAL,SPECIAL OCCASION
RRULE:FREQ=YEARLY
END:VEVENT
```


4.4.2 To-do Component

A "VTODO" calendar component is a grouping of component properties and an OPTIONAL "VALARM" calendar component that represent an action-item or assignment. For example, it MAY be an item of work assigned to an individual; such as "turn in travel expense today".

A "VTODO" calendar component is defined by the following notation:

```
todoc      = "BEGIN" ":" [ws] "VTODO" CRLF
             todoprop *alarmc
             "END" ":" [ws] "VTODO" CRLF

todoprop   = *attach *attendee *categories [class] *comment
             [completed] *contact [created] [description] dtstamp
             dtstart [due / duration] *exdate *exrule [geo]
             [last-mod] [location] [percent] priority [rstatus]
             *related *resources *rdate *rrule [recurid] [seq]
             [status] summary uid *url
```

The "VTODO" calendar component can not be nested within another calendar component. If "VTODO" calendar components need to be related to each other or to a "VTODO" or "VJOURNAL" calendar component, they can specify a relationship with the "RELATED-TO" property.

The following is an example of a "VTODO" calendar component:

```
BEGIN:VTODO
UID:19970901T130000Z-123404@host.com
DTSTAMP:19970901T1300Z
```

```
DTSTART:19970415T083000-0500
DUE:19970415T235959-0500
SUMMARY:1996 Income Tax Preparation
CLASS:CONFIDENTIAL
CATEGORIES:FAMILY,FINANCE
PRIORITY:1
STATUS:NEEDS-ACTION
END:VEVENT
```

4.4.3 Journal Component

A "VJOURNAL" calendar component is a grouping of component properties that represent one or more descriptive text notes on a particular calendar date. The "DTSTART" property is used to specify the calendar date that the journal entry is associated with. Generally, it will have a DATE value data type, but it MAY also be used to specify a DATE-TIME value data type. Examples of a journal entry include a daily record of a legislative body or a journal entry of individual telephone contacts for the day or an ordered list of accomplishments for the day. The calendar component can also be used to associate a document with a calendar date.

The "VJOURNAL" calendar component does not take up time on a calendar. Hence, it does not play a role in free or busy time searches - - it is as though it has a time transparency value of TRANSPARENT. It is transparent to any such searches.

A "VJOURNAL" calendar component is defined by the following notation:

```
journalc   = "BEGIN" ":" [ws] "VJOURNAL" CRLF
             jourprop
             "END" ":" [ws] "VJOURNAL" CRLF

jourprop   = *attach *attendee *categories [class] *comment
             *contact [created] [description] dtstart dtstamp
             *exdate *exrule [last-mod] *related *rdate *rrule
             [rstatus] [seq] summary uid *url [recurid]
```

The "VJOURNAL" calendar component can not be nested within another calendar component. If "VJOURNAL" calendar components need to be related to each other or to a "VEVENT" or "VTODO" calendar component,

they can specify a relationship with the "RELATED-TO" property.

The following is an example of the "VJOURNAL" calendar component:

```
BEGIN:VJOURNAL
UID:19970901T130000Z-123405@host.com
DTSTAMP:19970901T1300Z
DTSTART;VALUE=DATE:19970317
SUMMARY:Staff meeting minutes
DESCRIPTION:1. Staff meeting: Participants include Joe\, Lisa
  and Bob. Aurora project plans were reviewed. There is currently
  no budget reserves for this project. Lisa will escalate to
  management. Next meeting on Tuesday.
  2. Telephone Conference: ABC Corp. sales representative called
  to discuss new printer. Promised to get us a demo by Friday.
  3. Henry Miller (Handsoff Insurance): Car was totaled by tree.
  Is looking into a loaner car. 654-2323 (tel).
END:VJOURNAL
```

4.4.4 Free/Busy Component

A "VFREEBUSY" calendar component is a grouping of component properties that represents either a request for, or a reply with, free or busy time information. Typically, this component exists in an iCalendar object that is being used to either request or return free or busy time information.

A "VFREEBUSY" calendar component is defined by the following notation:

```
freebusyc  = "BEGIN" ":" [ws] "VFREEBUSY" CRLF
             fbprop
             "END" ":" [ws] "VFREEBUSY" CRLF
```



```
fbprop     = fbrequest / fbreply

fbrequest  = *attendee dtstart dtend [duration] *comment dtstamp
             [last-mod] *related [seq] uid
;This set of properties is used to for free/busy time request.

fbreply    = *attendee [created] *comment [dtstart dtend] dtstamp
             *freebusy [last-mod] *related [rstatus] [seq] uid *url
```

```
  ;This set of properties is used for free/busy time reply.
```

The "VFREEBUSY" calendar component can not be nested within another calendar component. The "VFREEBUSY" calendar components MAY be related to each other with the "RELATED-TO" property. Multiple "VFREEBUSY" calendar components MAY be specified within a iCalendar object. This permits the grouping of Free/Busy information into logical collections, such as monthly groups of busy time information.

The "VFREEBUSY" calendar component is intended for use in iCalendar object methods involving requests for free time, requests for busy time, requests for both free and busy, and the associated replies.

Free/Busy information can be expressed using the "FREEBBUSY" property. This property provides a terse representation of time periods. One or more "FREEBUSY" properties MAY be specified in the "VFREEBUSY" calendar component to describe the Free/Busy information.

Optionally, the "DTSTART" and "DTEND" properties MAY be specified to express the start and end date and time for all of the Free/Busy information in the "VFREEBUSY"calendar component. When present in a "VFREEBUSY" calendar component, they should be specified prior to any "FREEBUSY" properties. In a free time request, these properties MAY be used in combination with the "DURATION" property to express a request for a duration of free time within a given window of time.

The recurrence properties ("RRULE", "EXRULE", "RDATE", "EXDATE") are not permitted within a "VFREEBUSY" calendar component. Any recurring events are resolved into their individual busy time periods using the "FREEBUSY" property.

The following is an example of a "VFREEBUSY" calendar component:

```
  BEGIN:VFREEBUSY
  DTSTART:19971015T050000Z
  DTEND:19971016T050000Z
  FREEBUSY;VALUE=PERIOD:19971015T050000Z/PT8H30M,
   19971015T160000Z/PT5H30M,19971015T223000Z/PT6H30M
  END:VFREEBUSY
```

4.4.5 Alarm Component

A "VALARM" calendar component is a grouping of component properties that is a reminder or alarm for an event or a to-do. The "VALARM" calendar component MAY only be specified in a "VEVENT" or "VTODO"

calendar component. For example, it MAY define a reminder for a pending event or an overdue to-do.

The "DTSTART" property specifies the calendar date and time of day that the alarm will be triggered. The value MAY alternately be set to a period of time, before or after the event or to-do, that the alarm will be triggered.

A "VALARM" calendar component is defined by the following notation:

```
alarmc     = "BEGIN" ":" [ws] "VALARM" CRLF
             alarmprop
             "END" ":" [ws] "VALARM" CRLF

alarmprop  = *attach 1*categories *comment [description]
             dtstart duration *related repeat [summary]
```

The "VALARM" calendar component can only appear within either a "VEVENT" or "VTODO" calendar component. The "VALARM" calendar components can not be nested.

The following is an example of the "VALARM" calendar component:

```
BEGIN:VALARM
DTSTART:19970317T133000Z
REPEAT:4
DURATION:PT15M
CATEGORIES:DISPLAY,AUDIO
ATTACH:file:///mmedia/sounds/bell1.wav
DESCRIPTION:Breakfast meeting with executive team at 8:30 AM
END:VALARM
```

4.4.6 Timezone Component

The "VTIMEZONE" calendar component is used to define a time zone.

A time zone is unambiguously defined by the set of time measurement rules determined by the governing body for a given geographic area. These rules describe at a minimum the base offset from UTC for the time zone, often referred to as the Standard Time offset. Many locations adjust their Standard Time forward or backward by one hour, in order to accommodate seasonal changes in number of daylight hours, often referred to as Daylight Saving Time. Some locations adjust their time by a fraction of an hour. Standard Time is also known as Winter Time. Daylight Saving Time is also known as Advanced Time, Summer Time, or Legal Time in certain countries. The following table shows the changes in time zone rules for the eastern United States.

```
     Effective  Transition Rule
     Date       (Date/Time)                Offset   Abbreviation

     1920-1920 last Sun in Mar, 02:00  -0400    EDT
```



```
     1920-1920 last Sun in Oct, 02:00  -0500    EST

     1921-1966 last Sun in Apr, 02:00  -0400    EDT

     1921-1954 last Sun in Sep, 02:00  -0500    EST

     1955-1966 last Sun in Oct, 02:00  -0500    EST

     1967-*     last Sun in Oct, 02:00  -0500    EST

     1967-1973  last Sun in Apr, 02:00  -0400    EDT

     1974-1974  Jan 6, 02:00            -0400    EDT

     1975-1975  Feb 23, 02:00           -0400    EDT

     1976-1986  last Sun in Apr, 02:00  -0400    EDT

     1987-*     first Sun in Apr, 02:00 -0400    EDT
```

Interoperability between two calendaring and scheduling applications, especially for recurring events, to-dos or journal entries, is dependent on the ability to capture and convey date and time information in an unambiguous format. The specification of current time zone information is integral to this behavior.

The "VTIMEZONE" calendar component is a grouping of component properties that define a time zone description. The time zone description specifies the effective Standard Time or Daylight Savings Time rules for a particular time zone. The "VTIMEZONE" calendar component can not be nested within other calendar components. The "VTIMEZONE" calendar component MAY be specified multiple times. If the "VTIMEZONE" calendar component is missing, the recipient should assume all local times are relative to the recipient's time zone. The "VTIMEZONE" calendar component should be specified in the iCalendar object before any other calendar components.

A "VTIMEZONE" calendar component is defined by the following notation:

```
timezonec  = "BEGIN" ":" [ws] "VTIMEZONE" CRLF
             tzprop
             "END" ":" [ws] "VTIMEZONE" CRLF

tzprop     = *comment [daylight] dtstart [rdate / rrule]
             [tzname] tzoffset
```

The "VTIMEZONE" calendar component is important for correct interpretation of individual as well as recurring calendar components that span a time zone transition. For example, from EST to EDT. The exception to this are calendar components that are considered floating (i.e., occurs at a particular local time no matter what time zone you are in). If the iCalendar object contains a non-floating calendar component that has a recurring date pattern (i.e., includes

the "RRULE" property) or a list of date and local time values (i.e., includes the "RDATE" property), one or more "VTIMEZONE" calendar components MUST be specified, such that for the given range of the recurrence (i.e., the earliest instance to latest instance), there is valid time zone information for all instances. In other words, if all of the instances of the pattern is entirely within one offset observance, (e.g., all are in Standard Time), only one "VTIMEZONE" calendar component need be present. If a time zone transition is crossed, then other "VTIMEZONE" calendar components are needed. Further, if there are known changes to the rules for the time zone, even more "VTIMEZONE" calendar components are needed.

Each "VTIMEZONE" calendar component consists of several properties:

The "DAYLIGHT" property is a BOOLEAN value indicating Standard Time (FALSE) or Daylight Savings Time (TRUE). The default for DAYLIGHT is FALSE or Standard Time.

The "DTSTART" property in this usage is a fully specified DATE-TIME value, including the UTC offset, indicating the effective start date and time for the time zone information. For example, 19671029T020000-0400 represents the time at which the transition to Standard Time took effect in 1967 for the eastern United States.

The "TZOFFSET" property is a UTC-OFFSET value indicating the UTC offset for the time zone (Standard Time or Daylight Savings Time).

The "TZNAME" property is the customary name for the time zone.

The "RRULE" property indicates the recurrence rule for the transition to this time zone. For example, in the United States, the transition from Standard Time to Daylight Saving Time occurs on the first Sunday in April at 02:00. If a recurrence rule describing the transition is known to have an effective end date, the UNTIL recurrence rule parameter is used to specify that end date and time. If the recurrence rule for a particular observance (Daylight Saving Time) is changing, then the UNTIL of the first rule will be equal to the last valid instance (the last date-time) of this particular rule. See example below.

Alternatively, the "RDATE" property can be used. The "RDATE" property is a property that indicates the individual dates and/or times that the transition takes effect. The values supplied for "RDATE" property for each "VTIMEZONE" calendar component MUST provide valid time zone information of all instances of the recurrence specified for the calendar component to which this time zone information is to be applied.

The following are examples of the "VTIMEZONE" calendar component:

This is a simple example showing time zone information for the Eastern United States using "RDATE" property. Note that this is only suitable for a recurring event that starts on or later than 1997, April 6, at 02:00:00 EST (i.e., the earliest effective transition

date and time) and ends no later than 1998, April 7, 02:00:00 EST (i.e., latest valid date and time for EST in this scenario). For example, this can be used for a recurring event that ocurrs every Friday, 8am-9am, starting June 1, 1997, ending Dec 31, 1997.

```
BEGIN:VTIMEZONE
DAYLIGHT:FALSE
RDATE:19971026T020000-0400
TZOFFSET:-0500
TZNAME:EST
END:VTIMEZONE

BEGIN:VTIMEZONE
DAYLIGHT:TRUE
RDATE:19970406T020000-0500
TZOFFSET:-0400
TZNAME:EDT
END:VTIMEZONE
```

This is a simple example showing the current time zone rules for the Eastern United States using a RRULE recurrence pattern. Note that there is no effective end date to either of the Standard Time or Daylight Time rules. This information would be valid for a recurrening event starting today and continuing on into the known future.

```
BEGIN:VTIMEZONE
DAYLIGHT:FALSE
DTSTART:19671029T020000-0400
RRULE:FREQ=YEARLY;BYDAY=-1SU;BYMONTH=10
TZOFFSET:-0500
TZNAME:EST
END:VTIMEZONE

BEGIN:VTIMEZONE
DAYLIGHT:TRUE
DTSTART:19870405T020000-0500
RRULE: FREQ=YEARLY;BYDAY=1SU;BYMONTH=4
TZOFFSET:-0400
TZNAME:EDT
END:VTIMEZONE
```

This is an example showing a ficticious set of rules for the Eastern United States, where the Daylight Time rule has an effective end date (i.e., after that date, Daylight Time is no longer observed).

```
BEGIN:VTIMEZONE
DAYLIGHT:FALSE
DTSTART:19671029T020000-0400
RRULE:FREQ=YEARLY;BYDAY=-1SU;BYMONTH=10
TZOFFSET:-0500
TZNAME:EST
END:VTIMEZONE
```



```
BEGIN:VTIMEZONE
DAYLIGHT:TRUE
DTSTART:19870405T020000-0500
RRULE:FREQ=YEARLY;BYDAY=1SU;BYMONTH=4;UNTIL=19980404T020000-0500
TZOFFSET:-0400
TZNAME:EDT
END:VTIMEZONE
```

This is an example showing a fictitious set of rules for the Eastern

United States, where the first Daylight Time rule has an effective end date. There is a second Daylight Time rule that picks up where the other left off.

```
BEGIN:VTIMEZONE
DAYLIGHT:FALSE
DTSTART:19671029T020000-0400
RRULE:FREQ=YEARLY;BYDAY=-1SU;BYMONTH=10
TZOFFSET:-0500
TZNAME:EST
END:VTIMEZONE

BEGIN:VTIMEZONE
DAYLIGHT:TRUE
DTSTART:19870405T020000-0500
RRULE:FREQ=YEARLY;BYDAY=1SU;BYMONTH=4;UNTIL=19980404T020000-0500
TZOFFSET:-0400
TZNAME:EDT
END:VTIMEZONE

BEGIN:VTIMEZONE
DAYLIGHT:TRUE
DTSTART:19990327T020000-0500
RRULE:FREQ=YEARLY;BYDAY=-1SU;BYMONTH=3
TZOFFSET:-0400
TZNAME:EDT
END:VTIMEZONE
```

4.5 Calendar Properties

The Calendar Properties are attributes that apply to the iCalendar object, as a whole. These properties do not appear within a calendar component. They should be specified after the "BEGIN:VCALENDAR" property and prior to any calendar component.

4.5.1 Calendar Scale

This property is identified by the property name CALSCALE. This property defines the calendar scale used for the calendar information specified in the iCalendar object. This memo is based on the Gregorian calendar scale. The Gregorian calendar scale is assumed if this property is not specified in the iCalendar object. It is expected that other calendar scales will be defined in other specifications or by future versions of this memo.

The property is defined by the following notation:

```
calscale   = "CALSCALE" ":" [ws] calvalue CRLF

calvalue   = "GREGORIAN" / iana-scale

iana-scale = <Any other designator for a calendar scale
              registered with IANA>
```

The following is an example of this property:

```
CALSCALE:GREGORIAN
```

The data type for this property is TEXT.

4.5.2 Method

This property is identified by the property name METHOD. This property defines the iCalendar object method associated with the calendar object. When used in a MIME message entity, the value of this property MUST be the same as the Content-Type "method" parameter value. This property can only appear once within the iCalendar object.

The property is defined by the following notation:

```
method     = "METHOD" ": [ws] profvalue CRLF

profvalue  = <Any IANA registered iCalendar object method.>
```

The following is an example of this property when the iCalendar object is used to request a meeting:

```
METHOD: REQUEST
```

In the event that this property is not specified, the iCalendar object method is undefined. The data type for this property is TEXT.

4.5.3 Product Identifier

This property is identified by the property name PRODID. This property specifies the identifier for the product that created the iCalendar object. The vendor of the implementation should assure that this is a globally unique identifier; using some technique such as an ISO 9070 FPI value. This calendar property MUST be specified in the iCalendar object but can only appear once.

This property should not be used to alter the interpretation of an

iCalendar object beyond the semantics specified in this memo. For example, it is not to be used to further the understanding of non-standard properties.

The property is defined by the following notation:


```
prodid     = "PRODID" ":" [ws] pidvalue CRLF

pidvalue   = text
;Any text that describes the product and version
;and that is generally assured of being unique.
```

The following is an example of this property:

```
PRODID:-//ABC Corporation//NONSGML My Product//EN
```

The data type for this property is TEXT.

4.5.4 Source

This property is identified by the property name SOURCE. This property is defined by the [MIME DIR] memo. In this memo, the property identifies the URL for the source of the iCalendar object. The URL is useful for accessing the iCalendar object using a calendar access protocol.

The property is defined by the following notation:

```
source     = "SOURCE" ":" [ws] url CRLF
```

The following are examples of this property:

```
SOURCE:http://xyz.corp.com/corp-cals/1997-events.or4

SOURCE:http://xyz.corp.com/calendars/~jdoe
```

The data type for this property is URL.

4.5.5 Source Name

This property is identified by the property name NAME. This property is defined by the [MIME DIR] memo. The property identifies the displayable, presentation name for the source of the iCalendar

object. The source name is a useful text to associate in the user-interface of an application with the value in the SOURCE property.

The property is defined by the following notation:

```
name       = "NAME" ":" [ws] text CRLF
```

The following is an example of this property:

```
NAME:1997 Events Calendar for XYZ Corporation
```

The data type for this property is TEXT.

4.5.6 Version

This property is identified by the property name VERSION. This property specifies the identifier corresponding to the highest version number or the minimum and maximum range of the MIME Calendaring and Scheduling Content Type specification supported by the implementation that created the iCalendar object. A value of "2.0" corresponds to this memo. This calendar property MUST appear within the iCalendar object but can only appear once.

The property is defined by the following notation:

```
version    = "VERSION" ":" [ws] vervalue CRLF

vervalue   = "2.0"         ;This memo
           / maxver
           / (minver ";" [ws] maxver)

minver     = <A IANA registered iCalendar version identifier>
;Minimum iCalendar version used to create the iCalendar object

maxver     = <A IANA registered iCalendar version identifier>
;Maximum iCalendar version used to create the iCalendar object
```

The following is an example of this property:

```
VERSION:2.0
```

The data type for this property is TEXT.

4.6 Component Properties

The following properties MAY appear within calendar components, as specified by each component property definition.

4.6.1 Attachment

This property is identified by the property name ATTACH. The property provides the capability to associate an external object with a calendar component. For example, a document to be reviewed at a scheduled event or the description of the process steps for a to-do. The property MAY only be specified within "VEVENT", "VTODO", or "VJOURNAL" calendar components. This property MAY be specified multiple times within an iCalendar object.

The property is defined by the following notation:

```
attach     = "ATTACH" ":" [ws] url CRLF
```

The following are examples of this property:

```
ATTACH:CID:jsmith.part3.960817T083000.xyzMail@host1.com
```



```
ATTACH:FTP://xyzCorp.com/pub/reports/r-960812.ps
```

The data type for this property is URL.

4.6.2 Attendee

This property is identified by the property name ATTENDEE. The property defines an attendee within a calendar component. The property MAY only be specified within the "VEVENT", "VTODO", "VJOURNAL" and "VFREEBUSY" calendar components.

The property has the property parameters TYPE, for the type of attendee, ROLE, for the intended role of the attendee; STATUS, for the status of the attendee s participation; RSVP, for indicating whether the favor of a reply is requested; EXPECT, to indicate the

expectation of the attendee s participation by the originator; MEMBER, to indicate the groups that the attendee belongs to; DELEGATED-TO, to indicate the people that the original request was delegated to; and DELEGATED-FROM, to indicated whom the request was delegated from.

A recipient delegated a request MUST inherit the RSVP and EXPECT values from the attendee that delegated the request to them.

Multiple attendees MAY be specified by including multiple "ATTENDEE" properties within the MIME calendaring entity.

The property data type default is CAL-ADDRESS. The property data type MAY also be set to URL. This provides a useful mechanism to allow more than just the address of the attendee to be referenced. If the value is a URL, then it MUST be able to be resolved to a calendar address.

The property is defined by the following notation:

```
  attendee   = "ATTENDEE" [";" [ws] paramlist]
               [";" [ws] attparamlist] ":" [ws]
               (cal-address / URL) CRLF
  ;Value MUST match default or explicit data type

  attparamlist      = (attparam *(";" attparam)

  attparam   = typeparm / roleparm / statusparm / rsvpparm
             / expectparm / memberparm / deletoparm / delefromparm

  typeparm   = "TYPE" "="
             ("INDIVIDUAL"   ; An individual
             / "GROUP"       ; A group of individuals
             / "RESOURCE"    ; A physical resource
             / "ROOM"        ; A room resource
             / "UNKNOWN")    ; Otherwise not known
  ;Default value is INDIVIDUAL
```



```
  roleparm   = "ROLE" "="
             ("ATTENDEE"     ; Indicates a regular attendee
             / "OWNER"       ; Indicates owner of event or to-do
             / "ORGANIZER"   ; Indicates organizer of event or to-do
             / "DELEGATE")   ; Indicates delegate to event or to-do
```

```
    ;Default is ATTENDEE

    statusparm = "STATUS" "="
                ("NEEDS-ACTION" ; Indicates event or to-do needs action
                / "ACCEPTED"    ; Indicates event or to-do accepted
                / "DECLINED"    ; Indicates event or to-do not accepted
                / "TENTATIVE"   ; Indicates event or to-do tentatively
                ; accepted. Status MAY change in the future.
                / "COMPLETED"   ; Indicates to-do was completed.
                ; COMPLETED property has date/time completed.
                / "IN-PROCESS"  ;Indicates to-do is in the process of
                ; being completed.
                / "DELEGATED"   ; Indicateds event or to-do delegated
                ; to another ATTENDEE
                / "CANCELLED")  ; Indicates event or to-do cancelled for
                                ; ATTENDEE
    ;Default is NEEDS-ACTION

    rsvpparm   = "RSVP" "="
                  ("TRUE"              ; Indicates response requested
                / "FALSE")             ; Indicates no response needed
    ;Default is FALSE

    expectparm = "EXPECT" "="
                ("FYI"          ; Indicates request is for your info
                / "REQUIRE"     ; Indicates presence is required
                / "REQUEST")    ; Indicates presence is requested
    ;Default is FYI

    memberparm = "MEMBER" "=" cal-address *("," cal-address)
    ; Indicates a group or mailing list

    deletoparm = "DELEGATED-TO" "=" cal-address *("," cal-address)
    ; Indicates who request delegated to

    delefromparm = "DELEGATED-FROM" "=" cal-address *("," cal-address)
    ;Indicates who request delegated from
```

The following are examples of this property s use for a to-do:

```
    ATTENDEE;ROLE=OWNER;STATUS=COMPLETED:jsmith@host1.com
    ATTENDEE;MEMBER=DEV-GROUP@host2.com:joecool@host2.com
    ATTENDEE;DELEGATED-FROM=immud@host3.com:ildoit@host1.com
```

The following is an example of this property used for specifying multiple attendees to an event:

```
    ATTENDEE;ROLE=OWNER;STATUS=ACCEPTED:John Smith <jsmith@host1.com>
    ATTENDEE;ROLE=ATTENDEE;STATUS=TENTATIVE:Henry Cabot
```

```
   <hcabot@host2.com>
  ATTENDEE;ROLE=DELEGATE;STATUS=ACCEPTED:Jane Doe <jdoe@host1.com>
```

The following is an example of this property with the value specified as an URL reference to a vCard that contains the information about the attendee:

```
  ATTENDEE;ROLE=ATTENDEE;STATUS=ACCEPTED;VALUE=URL:
   http://www.xyz.com/~myvcard.vcf
```

The following is an example of this property with "delegatee" and"delegator" information for an event:

```
  ATTENDEE;ROLE=OWNER;STATUS=ACCEPTED:John Smith <jsmith@host1.com>
  ATTENDEE;ROLE=DELEGATE;STATUS=TENTATIVE;DELEGATED-FROM=
   iamboss@host2.com:Henry Cabot<hcabot@host2.com>
  ATTENDEE;ROLE=ATTENDEE;STATUS=DELEGATED;DELEGATED-TO=
   hcabot@host2.com=iamboss(The Big Cheese)@host2.com
  ATTENDEE;ROLE=DELEGATE;STATUS=ACCEPTED:Jane Doe <jdoe@host1.com>
```

The default data type for this property is CAL-ADDRESS. The data type MAY be reset to URL; in which case the value is a location or message that contains the information that is to be used to specify the attendee address.

4.6.3 Categories

This property is identified by the property name CATEGORIES. This property defines the categories for a calendar component. The property MAY be specified within the "VEVENT", "VTODO" or "VJOURNAL" calendar component with an arbitrary text value. In the "VALARM" calendar component the property MUST be specified to declare the alarm category. More than one category MAY be specified as a list of categories separated by the COMMA character (ASCII decimal 44).

The properties is defined by the following notation:

```
  categories = "CATEGORIES" [";" [ws] paramlist] ":" [ws]
               catvalue CRLF

  catvalue   = cat1value *["," [ws] cat1value]
             / cat2value *["," [ws] cat2value]

  cat1value  = "ANNIVERSARY" / "APPOINTMENT" / "BUSINESS"
             / "EDUCATION" / "HOLIDAY" / "MEETING" / "MISCELLANEOUS"
             / "NON-WORKING HOURS" / "NOT IN OFFICE" / "PERSONAL"
```

```
                / "PHONE CALL"  / "SICK DAY" / "SPECIAL OCCASION"
                / "TRAVEL" / "VACATION" / word
     ;Used only in "VEVENT", "VTODO" and "VJOURNAL" calendar components.

     cat2value  = "AUDIO" / "DISPLAY" / "EMAIL" / "PROCEDURE"
                / x-token / iana-word
     ;Used only in "VALARM" calendar component.
```


The following are examples of this property in a "VEVENT", "VTODO" or "VJOURNAL" calendar component:

```
  CATEGORIES:APPOINTMENT,EDUCATION

  CATEGORIES:MEETING
```

The following are examples of this property in a "VALARM" calendar component:

```
  CATEGORIES:AUDIO,DISPLAY

  CATEGORIES:PROCEDURE
```

The data type for this property is TEXT.

4.6.4 Classification

This property is identified by the property name CLASS. This property defines the access classification for a calendar component. The property MAY only be specified in a "VEVENT", "VTODO" or "VJOURNAL" calendar component. The property MAY only be specified once.

An access classification is only one component of the general security system within a calendar application. It provides a method of capturing the scope of the access the calendar owner intends for information within an individual calendar entry. The access classification of an individual iCalendar component is useful when measured along with the other security components of a calendar system (e.g., calendar user authentication, authorization, access rights, access role, etc.). Hence, the semantics of the individual access classifications can not be completely defined by this memo alone. Additionally, due to the "blind" nature of most exchange processes using this memo, these access classifications can not serve as an enforcement statement for a system receiving an iCalendar

object . Rather, they provide a method for capturing the intention of the calendar owner for the access to the calendar component. . The [ICMS] provides a broader description of the security system within a calendar application.

The property is defined by the following notation:

```
class      = "CLASS" [";" [ws] paramlist] ":" [ws]
             classvalue CRLF

classvalue = "PUBLIC" / "PRIVATE" / "CONFIDENTIAL" / x-token
;Default is PUBLIC
```

The following is an example of this property:

```
CLASS:PUBLIC
```

The data type for this property is TEXT.

4.6.5 Comment

This property is identified by the property name COMMENT. This property specifies non-processing information intended to provide a comment to the calendar user. The property MAY be specified in any of the calendar components. The property MAY be specified multiple times.

The property is defined by the following notation:

```
comment    = "COMMENT" ":" [ws] text CRLF
```

The following is an example of this property:

```
COMMENT:The meeting really needs to include both ourselves
  and the customer.  We can t hold this  meeting without them
  As a matter of fact\, the venue for the meeting ought to be at
  their site. - - John
```

The data type for this property is TEXT.

4.6.6 Contact

This property is defined by the property name CONTACT. The property

is used to represent contact information or alternately a reference to contact information associated with the calendar component. The property MAY only be specified in the "VEVENT", "VTODO" and "VJOURNAL" calendar components. The property value consists of textual contact information. Alternately, the value type for the property can be reset such that the property references the URL to the contact information.

The property is defined by the following notation:

```
contact    = "CONTACT" [";" [ws] paramlist] ":" [ws]
             text / url CRLF
```

The following is an example of this property referencing textual contact information:

```
CONTACT:Jim Dolittle\, ABC Industries\, +1-919-555-1234
```

The following is an example of this property referencing a LDAP URL to a directory entry containing the contact information:

```
CONTACT;VALUE=URL:"ldap://host.com:6666/o=3DABC%20Industries,
  c=3DUS??(cn=3DBJim%20Dolittle)"
```

The following is an example of this property referencing a MIME body part containing the contact information, such as a vCard embedded in a [MIME-DIR] content-type:

```
CONTACT;VALUE=URL:<part3.msg970930T083000SILVER@host.com>
```


The default data type for this property is TEXT. The data type MAY be reset to URL in order to specify a reference to the contact information.

4.6.7 Date/Time Completed

This property is identified by the property name COMPLETED. This property defines the date and time that a to-do was actually completed. The property MAY be specified once in a "VTODO" component. The date and time is an UTC value.

The property is defined by the following notation:

```
  completed  = "COMPLETED" ":" [ws] date-time CRLF
```

The following is an example of this property:

```
  COMPLETED:19960401T235959Z
```

The data type for this property is DATE-TIME.

4.6.8 Date/Time Created

This property is identified by the property name CREATED. This property specifies the date and time that the calendar information was created by the Organizer. The property MAY be specified in any of the calendar components. The property MAY only be specified once. The date and time is an UTC value.

The property is defined by the following notation:

```
  created    = "CREATED" ":" [ws] date-time CRLF
```

The following is an example of this property:

```
  CREATED:19960329T133000Z
```

The data type for this property is DATE-TIME.

4.6.9 Date/Time Due

This property is identified by the property name DUE. This property defines the date and time that a to-do is expected to be completed. The time can either be in local time, local time with UTC offset or UTC time. The property MAY only be specified in a "VTODO" calendar component and only once. The value MUST be a date/time after the DTSTART value, if specified.

The property is defined by the following notation:

```
  due        = "DUE" ":" [ws] date-time CRLF
```

The following is an example of this property:


```
  DUE:19960401T235959Z
```

The type for this property is DATE-TIME.

4.6.10 Date/Time End

This property is identified by the property name DTEND. This property MAY be specified within the "VEVENT", "VFREEBUSY", and "VTIMEZONE" calendar components.

Within the "VEVENT" calendar component, this property defines the end date and time for the event. The property is REQUIRED in "VEVENT" calendar components. The time can either be in local time, local time with UTC offset or UTC time. The local time is only to be used to specify date and time values that do not need to be fixed. A recipient MUST assume their own time zone for data and time values that do not include time zone information. The value MUST be later in time than the value of the "DTSTART" property.

Within the "VFREEBUSY" calendar component, this property defines the end date and time for the free or busy time information. The time MUST be specified in local time with UTC offset or UTC time. The value MUST be later in time than the value of the "DTSTART" property.

The property is defined by the following notation:

```
dtend      = "DTEND" ":" [ws] date-time CRLF
```

The following is an example of this property:

```
DTEND:19960401T235959Z
```

The data type for this property is DATE-TIME.

4.6.11 Date/Time Stamp

This property is identified by the property name DTSTAMP. This property specifies an UTC date/time stamp. The property indicates the date/time that the iCalendar object instance was created. This property MUST be included in "VEVENT", "VTODO", "VJOURNAL" and "VFREEBUSY" calendar components to permit the recipient to know when the iCalendar object was created.

This property is also useful to protocols such as [IMIP] that have inherent latency issues with the delivery of content. This property will assist in the proper sequencing of messages containing iCalendar objects.

This property is different than the "CREATED" and "LAST-MODIFIED" properties. These two properties are used to specify when the calendar service information was created and last modified. This is different than when the iCalendar object representation of the

calendar service information was created or last modified.

The property is defined by the following notation:

```
dtstamp    = "DTSTAMP" ":" [ws] date-time CRLF
```

The value type for this property is DATE-TIME. The value MUST be a UTC date/time value.

4.6.12 Date/Time Start

This property is identified by the property name DTSTART. This property MAY be specified within the "VEVENT", "VFREEBUSY", and "VTIMEZONE" calendar components.

Within the "VEVENT" calendar component, this property defines the start date and time for the event. The property is REQUIRED in "VEVENT" calendar components. The time can either be in local time, local time with UTC offset or UTC time. The local time is only to be used to specify date and time values that do not need to be fixed. A recipient MUST assume their own time zone for data and time values that do not include time zone information. Events MAY have a start date/time but no end date/time. In that case, the event does not take up any time.

Within the "VFREEBUSY" calendar component, this property defines the start date and time for the free or busy time information. The time MUST be specified in local time with UTC offset or UTC time.

Within the "VTIMEZONE" calendar component, this property defines the effective start date and time for a time zone specification. This property is REQUIRED within "VTIMEZONE" calendar components. The time MUST be specified as a UTC time.

The property is defined by the following notation:

```
dtstart    = "DTSTART" [";" [ws] paramlist] ":" [ws] (date-time /
             date) CRLF
```

The following is an example of this property:

```
DTSTART:19960401T235959-0600
```

The default data type for this property is DATE-TIME. The data type MAY be overridden to be DATE.

4.6.13 Daylight

This property is identified by the property name DAYLIGHT. This property MAY only be specified in a "VTIMEZONE" calendar component. This property specifies whether Daylight Saving Time (i.e., value is TRUE) or Standard Time (i.e., value is FALSE) is in effect for the time zone. The default value is FALSE or Standard Time.

The property is defined by the following notation:


```
daylight   = "DAYLIGHT" ":" [ws] boolean CRLF
;Default value is FALSE
```

The following is an example of this property:

```
DAYLIGHT:TRUE              ;Specifies DST in effect in time zone
```

The data type for this property is BOOLEAN.

4.6.14 Description

This property is identified by the property name DESCRIPTION. This property provides a more complete description of the calendar component, than that provided by the "SUMMARY" property. The property MAY be specified in the "VEVENT", "VTODO" and "VJOURNAL" calendar components. The property MAY be specified multiple times only within a "VJOURNAL" calendar component.

The property is defined by the following notation:

```
Description        = "DESCRIPTION" [";" [ws] paramlist] ":" [ws]
                     text CRLF
```

The following is an example of the property with formatted line breaks in the property value:

```
DESCRIPTION:Meeting to provide technical review for "Phoenix"
  design.\n Happy Face Conference Room. Phoenix design team
  MUST attend this meeting.\n RSVP to team leader.
```

The following is an example of the property with folding of long lines:

```
DESCRIPTION:Last draft of the new novel is to be completed
  for the editor s proof today.
```

The data type for this property is TEXT.

4.6.15 Duration

This property is identified by the property name DURATION. The property specifies a duration of time. The property MAY be specified in a "VEVENT" calendar component in order to specify a duration of the event, instead of an explicit end date/time. The property MAY be specified in a "VTODO" calendar component in order to specify a duration for the to-do. The property MAY be specified in a "VFREEBUSY" calendar component in order to specify the amount of free time being requested. The property MAY be specified in an "VALARM" calendar component in order to specify the period between repeating alarms.

The property is defined by the following notation:


```
duration   = "DURATION" ":" [ws] duration CRLF
```

The following is an example of this property that specifies an interval of time of 1 hour and zero minutes and zero seconds:

```
DURATION:PT1H0M0S
```

The following is an example of this property that specifies an interval of time of 15 minutes.

```
DURATION:PT15M
```

The data type for this property is DURATION.

4.6.16 Exception Date/Times

This property is identified by the property name EXDATE. This property defines the list of date/time exceptions for a recurring "VEVENT", "VTODO" or "VJOURNAL" calendar component. The times can

either be in local time, local time with UTC offset or UTC time.

The exception dates, if specified, is used in computing the recurrence set. The recurrence set is the complete set of recurrence instances for a calendar component. The recurrence set is generated by considering the initial "DTSTART" property along with the "RRULE", "RDATE", "EXDATE" and "EXRULE" properties contained within the iCalendar object. The "DTSTART" property defines the first instance in the recurrence set. Multiple instances of the "RRULE" and "EXRULE" properties MAY also be specified to define more sophisticated recurrence sets. The final recurrence set is generated by gathering all of the start date-times generated by any of the specified "RRULE" and "RDATE" properties, and excluding any start date and times which fall within the union of start date and times generated by any specified "EXRULE" and "EXDATE" properties. This implies that start date and times within exclusion related properties (i.e., "EXDATE" and "EXRULE") take precedence over those specified by inclusion properties (i.e., "RDATE" and "RRULE"). Where duplicate instances are generated by the "RRULE" and "RDATE" properties, only one recurrence is considered. Duplicate instances are ignored.

The property is defined by the following notation:

```
exdate     = "EXDATE" [";" [ws] paramlist] ":" [ws] [date-time /
             date] *("," [ws] date-time/date) CRLF
;Values MUST match the specified value type.
```

The following is an example of this property:

```
EXDATE:19960402T010000Z,19960403T010000Z,19960404T010000Z
```

The data type for this property is DATE-TIME. The data type MAY be reset to DATE.

4.6.17 Exception Rule

This property is identified by the property name EXRULE. This property defines a rule or repeating pattern for an exception to a recurrence set. This property MAY only be specified in the "VEVENT", "VTODO" or "VJOURNAL" calendar components.

The exception rule, if specified, is used in computing the recurrence

set. The recurrence set is the complete set of recurrence instances for a calendar component. The recurrence set is generated by considering the initial "DTSTART" property along with the "RRULE", "RDATE", "EXDATE" and "EXRULE" properties contained within the iCalendar object. The "DTSTART" defines the first instance in the recurrence set. Multiple instances of the "RRULE" and "EXRULE" properties MAY also be specified to define more sophisticated recurrence sets. The final recurrence set is generated by gathering all of the start date-times generated by any of the specified "RRULE" and "RDATE" properties, and excluding any start date and times which fall within the union of start date and times generated by any specified "EXRULE" and "EXDATE" properties. This implies that start date and times within exclusion related properties (i.e., "EXDATE" and "EXRULE") take precedence over those specified by inclusion properties (i.e., "RDATE" and "RRULE"). Where duplicate instances are generated by the "RRULE" and "RDATE" properties, only one recurrence is considered. Duplicate instances are ignored.

The property is defined by the following notation:

```
  exrule     = "EXRULE" [";" [ws] paramlist] ":" [ws] recur CRLF
```

The following are examples of this property. Except every other week, on Tuesday and Thursday for 4 occurrences:

```
  EXRULE:FREQ=WEEKLY;COUNT=4;INTERVAL=2;BYDAY=TU,TH
```

Except daily for 10 occurrences:

```
  EXRULE:FREQ=DAILY;COUNT=10
```

Except yearly in June and July for 8 occurrences:

```
  EXRULE:FREQ=YEARLY;COUNT=8;BYMONTH=6,7
```

The data type for this property is RECUR.

4.6.18 Free/Busy Time

This property is identified by the property name FREEBUSY. The property defines one or more free or busy time intervals. These time periods MAY be specified as either a start and end date-time or a start date-time and duration.

The date and time is either local time with UTC offset or a UTC value.

The "FREEBUSY" property MAY include the "TYPE" property parameter to specify whether the information defines a free or busy time interval. The default "TYPE" property parameter value is BUSY. The property MAY also include the "STATUS" property parameter to provide added information about the busy time. For example, if the busy time is associated with a time interval that would be unavailable for scheduling - - in any case - - or busy time that has been tentatively scheduled. The default "STATUS" property parameter value is BUSY. The property is defined by the following notation:

```
freebusy   = "FREEBUSY" [";" [ws] paramlist] [";" [ws] fbparmlist]
             ":" [ws] fbvalue CRLF

fbparmlist = fbparam *(";" [ws] fbparam)

fbparam    = fbtype / fbstatus

fbtype     = "TYPE" "=" ("FREE" or "BUSY")
;Default is BUSY

fbstatus   = "STATUS" "="
             "BUSY"        ;Represents busy time interval.
           / "UNAVAILABLE" ;Represents busy, but unavailable
                           ;interval for cheduling; such as
                           ;out-of-office or non-working hours.
           / "TENTATIVE"   ;Represents busy, but tentatively
                           ;scheduled interval.
;Default is BUSY

fbvalue    = period *["," [ws] period]
;Value MUST match default or explicit data type
```

The following are some examples of this property:

```
FREEBUSY;STATUS=UNAVAILABLE:19970308T160000Z/PT8H30M

FREEBUSY;TYPE=FREE:19970308T160000Z/PT3H,19970308T200000Z/PT1H
```

"FREEBUSY" properties within the "VFREEBUSY" calendar component should be sorted in ascending order, based on start time and then end time, with the earliest periods first.

The "FREEBUSY" property MAY specify more than one value, separated by the COMMA character (ASCII decimal 44). In such cases, the "FREEBUSY" property values should all be of the same "STATUS" property parameter type (e.g., all values of a particular "STATUS" listed together in a

single property).

The data type for this property is PERIOD.

4.6.19 Geographic Position

This property is identified by the property name GEO. This property specifies information related to the global position for a "VEVENT" or "VTODO" calendar component. The property value specifies latitude and longitude, in that order (i.e., "LAT LON" ordering). The longitude represents the location east and west of the prime meridian as a positive or negative real number, respectively. The latitude represents the location north and south of the equator as a positive or negative real number, respectively. The longitude and latitude values MUST be specified as decimal degrees and should be specified to six decimal places. This will allow for granularity within a meter of the geographical position. The simple formula for converting degrees-minutes-seconds into decimal degrees is:

```
decimal = degrees + minutes/60 + seconds/3600.
```

The property is defined by the following notation:

```
geo        = "GEO" ":" [ws] geovalue CRLF

geovalue   = float ";" [ws] float
;Latitude and Longitude components
```

The following is an example of this property:

```
GEO:37.386013;-122.082932
```

The default data type for this property is FLOAT.

4.6.20 Last Modified

This property is identified by the property name LAST-MODIFIED. The property specifies the date and time that the calendar information was last revised. This property MAY be specified in the "VEVENT", "VTODO", "VJOURNAL" or "VFREEBUSY" calendar components. The data and time MUST be a UTC value.

The property is defined by the following notation:

```
last-mod   = "LAST-MODIFIED" ":" [ws] date-time CRLF
```

The following is are examples of this property:

```
LAST-MODIFIED:19960817T133000Z
```

The data type for this property is DATE-TIME.

4.6.21 Location

This property is identified by the property name LOCATION. The property defines the intended location for the "VEVENT" or "VTODO"

calendar component. The property MAY only be specified within a "VEVENT" or "VTODO" calendar component.

The property is defined by the following notation:

```
location   = "LOCATION [";" [ws] paramlist] ":" [ws] locavalue
             CRLF

locavalue  = text / url    ;The value MUST be the same type as the
                           ;default or explicit data type.
```

The following are some examples of this property:

```
LOCATION:Conference Room - F123, Bldg. 002

LOCATION;VALUE=URL:http://www.xyzcorp.com/~jsmith.vcf
```

The default data type for this property is TEXT. The data type MAY be reset to URL. In the case of the data type being URL, the property value MAY reference a vCard object. This provides a useful mechanism to specify a location in terms of its electronic business card.

4.6.22 Percent Complete

This property is identified by the property name PERCENT-COMPLETE. This property is used by an assignee or delegatee of a to-do to convey the percent completion of a to-do to the Organizer. The

property MAY only be specified once and within a "VTODO" calendar component. The property value is an integer between zero and one hundred. A value of "0" indicates the to-do has not yet been started. A value of "100" indicates that the to-do has been completed. Integer values in between indicate the percent partially complete.

When a to-do is assigned to multiple individuals, the property value indicates the percent complete for that portion of the to-do assigned to the assignee or delegatee. For example, if a to-do is assigned to both individuals "A" and "B". A reply from "A" with a percent complete of "70" indicates that "A" has completed 70% of the to-do assigned to them. A reply from "B" with a percent complete of "50" indicates "B" has completed 50% of the to-do assigned to them.

The property is defined by the following notation:

```
percent = "PERCENT-COMPLETE" ":" [ws] integer CRLF
```

The following is an example of this property to show 39% completion:

```
PERCENT-COMPLETE:39
```

The data type for this property is INTEGER.

4.6.23 Priority

This property is identified by the property name PRIORITY. The property defines the priority for event or to-do. The property MAY only be specified within a "VEVENT" or "VTODO" calendar component. The value is an integer. A value of zero (ASCII decimal 48) specifies an undefined priority. A value of one (ASCII decimal 49) is the highest priority. A value of two (ASCII decimal 50) is the second highest priority. Subsequent numbers specify a decreasing ordinal priority.

The property is specified by the following notation:

```
priority   = "PRIORITY" ":" [ws] integer CRLF
;Default is zero
```

The following is an example of this property:

PRIORITY:2

The data type for this property is INTEGER.

4.6.24 Recurrence Date/Times

This property is identified by the property name RDATE. This property defines the list of date/times for a recurrence set. The property MAY only appear within the "VEVENT", "VTODO", "VJOURNAL" or "VTIMEZONE" calendar components. This property MAY appear along with the "RRULE" property to define an aggregate set of repeating occurrences. When they both appear in an iCalendar object, the recurring events are defined by the union of occurrences defined by both the "RDATE" and "RRULE". The times can either be in local time, local time with UTC offset or UTC based time. If local time is used, the "VTIMEZONE" calendar component MUST be included in the iCalendar object, otherwise the local time value will be interpreted relative to the time zone of the recipient. The period values for "RDATE" property are specified using a specific start and a specific end basic format (period-explicit) or the period with a specific start and a specific duration basic format (period-start).

The recurrence dates, if specified, is used in computing the recurrence set. The recurrence set is the complete set of recurrence instances for a calendar component. The recurrence set is generated by considering the initial "DTSTART" property along with the "RRULE", "RDATE", "EXDATE" and "EXRULE" properties contained within the iCalendar object. The "DTSTART" property defines the first instance in the recurrence set. Multiple instances of the "RRULE" and "EXRULE" properties MAY also be specified to define more sophisticated recurrence sets. The final recurrence set is generated by gathering all of the start date-times generated by any of the specified "RRULE" and "RDATE" properties, and excluding any start date and times which fall within the union of start date and times generated by any specified "EXRULE" and "EXDATE" properties. This implies that start

date and times within exclusion related properties (i.e., "EXDATE" and "EXRULE") take precedence over those specified by inclusion properties (i.e., "RDATE" and "RRULE"). Where duplicate instances are generated by the "RRULE" and "RDATE" properties, only one recurrence is considered. Duplicate instances are ignored.

The property is defined by the following notation:

```
rdate      = "RDATE" [";" [ws] paramlist] ":" [ws] rdvalue
             *("," [ws] rdvalue) CRLF

rdvalue    = date-time / date / period
;Value MUST match the specified data type
```

The following are examples of this property:

```
RDATE:19970714T083000-0400

RDATE;VALUE=PERIOD:19960403T020000Z/19960403T040000Z,
 19960404T010000Z/PT3H

RDATE;VALUE=DATE:19970101,19970120,19970217,19970421
 19970526,19970704,19970901,19971014,19971128,19971129,19971225
```

The default data type for this property is DATE-TIME. The data type MAY be reset to DATE or PERIOD.

4.6.25 Recurrence ID

This property is identified by the property name RECURRENCE-ID. This property identifies a specific instance of a recurring "VEVENT", "VTODO" or "VJOURNAL" calendar component. The property value is the effective value of the "DTSTART" property of the recurrence instance. The time of day component for the value MUST be either an UTC or a local time with UTC offset time format, unless the original calendar object was expressed as a "floating" calendar object; that is in local time with no time zone calendar component specified. If the value of the "DTSTART" property is a DATE type value, then the value MUST be the calendar date for the recurrence instance.

The date/time value is set to the time when the original recurrence instance would occur - - meaning that if the intent is to change a Friday meeting to Thursday, the date/time is still set to the original Friday meeting.

The "RECURRENCE-ID" property is used in conjunction with the "UID" property to identify a particular instance of a recurring event, to-do or journal.

The property is defined by the following notation:

```
recurid    = "RECURRENCE-ID" [";" [ws] rangeparm] [";" [ws]
             paramlist] ":" [ws] [date-time / date]
```

rangeparm = "RANGE" "=" ("THISANDPRIOR" / "THISANDFUTURE")

The default value for the range parameter is the single recurrence instance only.

The following are examples of this property:

RECURRENCE-ID:19960401T235959Z

RECURRENCE-ID;RANGE=THISANDFUTURE:19960120T120000Z

This default data type for this property is DATE-TIME. The data type MAY be reset to DATE.

4.6.26 Recurrence Rule

This property is identified by the property name RRULE. This property defines a rule or repeating pattern for a recurring events, to-dos, or time zone definitions. The property MAY be specified in the "VEVENT", "VTODO", "VJOURNAL" or "VTIMEZONE" calendar components. The data type for this property is RECUR.

The recurring rule, if specified, is used in computing the recurrence set. The recurrence set is the complete set of recurrence instances for a calendar component. The recurrence set is generated by considering the initial "DTSTART" property along with the "RRULE", "RDATE", "EXDATE" and "EXRULE" properties contained within the iCalendar object. The "DTSTART" property defines the first instance in the recurrence set. Multiple instances of the "RRULE" and "EXRULE" properties MAY also be specified to define more sophisticated recurrence sets. The final recurrence set is generated by gathering all of the start date-times generated by any of the specified "RRULE" and "RDATE" properties, and excluding any start date and times which fall within the union of start date and times generated by any specified "EXRULE" and "EXDATE" properties. This implies that start date and times within exclusion related properties (i.e., "EXDATE" and "EXRULE") take precedence over those specified by inclusion properties (i.e., "RDATE" and "RRULE"). Where duplicate instances are generated by the "RRULE" and "RDATE" properties, only one recurrence is considered. Duplicate instances are ignored.

The "DTSTART" and "DTEND" property pair or "DTSTART" and "DURATION" property pair, specified within the iCalendar object defines the first instance of the recurrence. When used with a recurrence rule, the "DTSTART" and "DTEND" properties MUST be specified in local time and the appropriate set of "VTIMEZONE" calendar components MUST be

included. For detail on the usage of the "VTIMEZONE" calendar component, see the "VTIMEZONE" calendar component definition.

Any duration associated with the iCalendar object applies to all members of the generated recurrence. Any modified duration for specific recurrences would have to be explicitly specified using the "RDATE" property.

This property is defined by the following notation:

```
rrule      = "RRULE" [";" [ws] paramlist] ":" [ws] recur CRLF
```

Examples of this property include the following. All examples assume the Eastern United States time zone.

Daily for 10 occurrences:

```
DTSTART:19970902T090000-0400
RRULE:FREQ=DAILY;COUNT=10

==> (1997 9AM EDT)September 2-11
```

Daily until 12/24/97:

```
DTSTART:19970902T090000-0400
RRULE:FREQ=DAILY;UNTIL=19971224T000000Z

==> (1997 9AM EDT)September 2-30;October 1-25
    (1997 9AM EST)October 26-31;November 1-30;December 1-24
```

Every other day - forever:

```
DTSTART:19970902T090000-0400
RRULE:FREQ=DAILY;INTERVAL=2
==> (1997 9AM EDT)September2,4,6,8...24,26,28,30;
     October 2,4,6...20,22,24
    (1997 9AM EST)October 26,28,30;November 1,3,5,7...25,27,29;
     Dec 1,3,...
```

Every 10 days, 5 occurrences:

```
DTSTART:19970902T090000-0400
RRULE:FREQ=DAILY;INTERVAL=10;COUNT=5

==> (1997 9AM EDT)September 2,12,22;October 2,12
```

Everyday in January, for 3 years:

```
DTSTART:19980101T090000-0400
RRULE:FREQ=YEARLY;UNTIL:20000131T090000-0400;BYMONTH=1;BYDAY=SU,
 MO,TU,WE,TH,FR,SA
or
RRULE:FREQ=DAILY;UNTIL:20000131T090000-0400;BYMONTH=1

==> (1998 9AM EDT)January 1-31
    (1999 9AM EDT)January 1-31
    (2000 9AM EDT)January 1-31
```

Weekly for 10 occurrences

```
DTSTART:19970902T090000-0400
RRULE:FREQ=WEEKLY;COUNT=10
```



```
==> (1997 9AM EDT)September 2,9,16,23,30;October 7,14,21
    (1997 9AM EST)October 28;November 4
```

Weekly until 12/24/94

```
DTSTART:19970902T090000-0400
RRULE:FREQ=WEEKLY;UNTIL=19971224T000000Z

==> (1997 9AM EDT)September 2,9,16,23,30;October 7,14,21
    (1997 9AM EST)October 28;November 4,11,18,25;
                  December 2,9,16,23,24
```

Every other week - forever:

```
DTSTART:19970902T090000-0400
RRULE:FREQ=WEEKLY;INTERVAL=2;WKST=SU

==> (1997 9AM EDT)September 2,16,30;October 14
    (1997 9AM EST)October 28;November 11,25;December 9,23
    (1998 9AM EST)January 6,20;February
...
```

Weekly on Tuesday and Thursday for 5 weeks:

```
DTSTART:19970902T090000-0400
RRULE:FREQ=WEEKLY;UNTIL=19971007T000000-0400;WKST=SU;BYDAY=TU,TH
```

```
  or
  RRULE:FREQ=WEEKLY;COUNT=10;WKST=SU;BYDAY=TU,TH

  ==> (1997 9AM EDT)September 2,4,9,11,16,18,23,25,30;October 2

Every other week on Monday, Wednesday and Friday until 12/24/97, but
starting on Tuesday, 9/2/97:

  DTSTART:19970902T090000-0400
  RRULE:FREQ=WEEKLY;INTERVAL=2;UNTIL=19971224T000000-0400;WKST=SU;
   BYDAY=MO,WE,FR
  ==> (1997 9AM EDT)September 2,3,5,15,17,19,29;October 1,3,13,15,17
      (1997 9AM EST)October 27,29,31;November 10,12,14,24,26,28;
                    December 8,10,12,22,24

Every other week on Tuesday and Thursday, for 8 occurrences:

  DTSTART:19970902T090000-0400
  RRULE:FREQ=WEEKLY;INTERVAL=2;COUNT=8;WKST=SU;BYDAY=TU,TH

  ==> (1997 9AM EDT)September 2,4,16,18,30;October 2,14,16

Monthly on the 1st Friday for ten occurrences:

  DTSTART:19970905T090000-0400
  RRULE:FREQ=MONTHLY;COUNT=10;BYDAY=1FR


Dawson/Stenerson                  57                  Expires MAY 1998


Internet Draft       C&S Core Object Specification     October 22, 1997



  ==> (1997 9AM EDT)September 5;October 3
      (1997 9AM EST)November 7;Dec 5
      (1998 9AM EST)January 2;February 6;March 6;April 3
      (1998 9AM EDT)MAY 1;June 5

Monthly on the 1st Friday until 12/24/94:

  DTSTART:19970905T090000-0400
  RRULE:FREQ=MONTHLY;UNTIL=19971224T000000Z;BYDAY=1FR

  ==> (1997 9AM EDT)September 5;October 3
      (1997 9AM EST)November 7;December 5

Every other month on the 1st and last Sunday of the month for 10
occurrences:

  DTSTART:19970907T090000-0400
```

```
  RRULE:FREQ=MONTHLY;INTERVAL=2;COUNT=10;BYDAY=1SU,-1SU

  ==> (1997 9AM EDT)September 7,28
      (1997 9AM EST)November 2,30
      (1998 9AM EST)January 4,25;March 1,29
      (1998 9AM EDT)MAY 3,31
```

Monthly on the second to last Monday of the month for 6 months:

```
  DTSTART:19970922T090000-0400
  RRULE:FREQ=MONTHLY;COUNT=6;BYDAY=-2MO

  ==> (1997 9AM EDT)September 22;October 20
      (1997 9AM EST)November 17;December 22
      (1998 9AM EST)January 19;February 16
```

Monthly on the third to the last day of the month, forever:

```
  DTSTART:19970928T090000-0400
  RRULE:FREQ=MONTHLY;BYMONTHDAY=-3

  ==> (1997 9AM EDT)September 28
      (1997 9AM EST)October 29;November 28;December 29
      (1998 9AM EDT)January 29;February 26
  ...
```

Monthly on the 2nd and 15th of the month for 10 occurrences:

```
  DTSTART:19970902T090000-0400
  RRULE:FREQ=MONTHLY;COUNT=10;BYMONTHDAY=2,15

  ==> (1997 9AM EDT)September 2,15;October 2,15
      (1997 9AM EST)November 2,15;December 2,15
      (1998 9AM EST)January 2,15
```

Monthly on the first and last day of the month for 10 occurrences:


```
  DTSTART:19970930T090000-0400
  RRULE:FREQ=MONTHLY;COUNT=10;BYMONTHDAY=1,-1

  ==> (1997 9AM EDT)September 30;October 1
      (1997 9AM EST)October 31;November 1,30;December 1,31
      (1998 9AM EST)January 1,31;February 1
```

Every 18 months on the 10th thru 15th of the month for 10

occurrences:

```
  DTSTART:19970910T090000-0400
  RRULE:FREQ=MONTHLY;INTERVAL=18;COUNT=10;BYMONTHDAY=10,11,12,13,14,
   15

  ==> (1997 9AM EDT)September 10,11,12,13,14,15
      (1999 9AM EST)March 10,11,12,13
```

Every Tuesday, every other month:

```
  DTSTART:19970902T090000-0400
  RRULE:FREQ=MONTHLY;INTERVAL=2;BYDAY=TU

  ==> (1997 9AM EDT)September 2,9,16,23,30
      (1997 9AM EST)November 4,11,18,25
      (1998 9AM EST)January 6,13,20,27;March 3,10,17,24,31
  ...
```

Yearly in June and July for 10 occurrences:

```
  DTSTART:19970610T090000-0400
  RRULE:FREQ=YEARLY;COUNT=10;BYMONTH=6,7

  ==> (1997 9AM EDT)June 10;July 10
      (1998 9AM EDT)June 10;July 10
      (1999 9AM EDT)June 10;July 10
      (2000 9AM EDT)June 10;July 10
      (2001 9AM EDT)June 10;July 10
  Note: Since none of the BYDAY, BYMONTHDAY or BYYEARDAY components
  are specified, the day is gotten from DTSTART
```

Every other year on January, February, and March for 10 occurrences:

```
  DTSTART:19970310T090000-0400
  RRULE:FREQ=YEARLY;INTERVAL=2;COUNT=10;BYMONTH=1,2,3

  ==> (1997 9AM EST)March 10
      (1999 9AM EST)January 10;February 10;March 10
      (2001 9AM EST)January 10;February 10;March 10
      (2003 9AM EST)January 10;February 10;March 10
```

Every 3rd year on the 1st, 100th and 200th day for 10 occurrences:

```
  DTSTART:19970101T090000-0400
  RRULE:FREQ=YEARLY;INTERVAL=3;COUNT=10;BYYEARDAY=1,100,200
```

```
  ==> (1997 9AM EST)January 1
      (1997 9AM EDT)April 10;July 19
      (2000 9AM EST)January 1
      (2000 9AM EDT)April 9;July 18
      (2003 9AM EST)January 1
      (2003 9AM EDT)April 10;July 19
      (2006 9AM EST)January 1

Every 20th Monday of the year, forever:

  DTSTART:19970519T090000-0400
  RRULE:FREQ=YEARLY;BYDAY=20MO

  ==> (1997 9AM EDT)MAY 19
      (1998 9AM EDT)MAY 18
      (1999 9AM EDT)MAY 17
  ...

Monday of Week No. 20, forever:

  DTSTART:19970512T090000-0400
  RRULE:FREQ=YEARLY;BYWEEKNO=20;BYDAY=MO

  ==> (1997 9AM EDT)MAY 12
      (1998 9AM EDT)MAY 11
      (1999 9AM EDT)MAY 17
  ...

Every Thursday in March, forever:

  DTSTART:19970313T090000-0400
  RRULE:FREQ=YEARLY;BYMONTH=3;BYDAY=TH

  ==> (1997 9AM EST)March 13,20,27
      (1998 9AM EST)March 5,12,19,26
      (1999 9AM EST)March 4,11,18,25
  ...

Every Thursday, but only during summer months, forever:

  DTSTART:19970605T090000-0400
  RRULE:FREQ=YEARLY;BYDAY=TH;BYMONTH=6,7,8

  ==> (1997 9AM EDT)June 5,12,19,26;July 3,10,17,24,31;
                    August 7,14,21,28
      (1998 9AM EDT)June 4,11,18,25;July 2,9,16,23,30;
                    August 6,13,20,27
      (1999 9AM EDT)June 3,10,17,24;July 1,8,15,22,29;
                    August 5,12,19,26
```

```
     ...
```

Every Friday the 13th, forever:


```
     DTSTART:19970902T090000-0400
     EXDATE:19970902T090000-0400
     RRULE:FREQ=MONTHLY;BYDAY=FR;BYMONTHDAY=13

     ==> (1998 9AM EST)February 13;March 13;November 13
         (1999 9AM EDT)August 13
         (2000 9AM EDT)October 13
     ...
```

The first Saturday that follows the first Sunday of the month, forever:

```
     DTSTART:19970913T090000-0400
     RRULE:FREQ=MONTHLY;BYDAY=SA;BYMONTHDAY=7,8,9,10,11,12,13

     ==> (1997 9AM EDT)September 13;October 11
         (1997 9AM EST)November 8;December 13
         (1998 9AM EST)January 10;February 7;March 7
         (1998 9AM EDT)April 11;MAY 9;June 13...
     ...
```

Every four years, the first Tuesday after a Monday in November, forever (U.S. Presidential Election day):

```
     DTSTART:19961105T090000-0400
     RRULE:FREQ=YEARLY;INTERVAL=4;BYMONTH=11;BYDAY=TU;BYMONTHDAY=2,3,4,
      5,6,7,8

     ==> (1996 9AM EST)November 5
         (2000 9AM EST)November 7
         (2004 9AM EST)November 2
     ...
```

The 3rd instance into the month of one of Tuesday, Wednesday or Thursday, for the next 3 months:

```
     DTSTART:19970904T090000-0400
     RRULE:FREQ=MONTHLY;COUNT=3;BYDAY=TU,WE,TH;BYSETPOS=3

     ==> (1997 9AM EDT)September 4;October 7
```

(1997 9AM EST)November 5

The 2nd to last weekday of the month:

```
DTSTART:19970929T090000-0400
RRULE:FREQ=MONTHLY;BYDAY=MO,TU,WE,TH,FR;BYSETPOS=-2

==> (1997 9AM EDT)September 29
    (1997 9AM EST)October 30;November 27;December 30
    (1998 9AM EST)January 29;February 26;March 30
...
```

Every 3 hours from 9AM to 5PM on a specific day:


```
DTSTART:19970902T090000-0400
RRULE:FREQ=HOURLY;INTERVAL=3;UNTIL=19970902T170000-0400

==> (September 2, 1997 EDT)09:00,12:00,15:00
```

Every 15 minutes for 6 occurrences:

```
DTSTART:19970902T090000-0400
RRULE:FREQ=MINUTELY;INTERVAL=15;COUNT=6

==> (September 2, 1997 EDT)09:00,09:15,09:30,09:45,10:00,10:15
```

Every hour and a half for 4 occurrences:

```
DTSTART:19970902T090000-0400
RRULE:FREQ=MINUTELY;INTERVAL=90;COUNT=4

==> (September 2, 1997 EDT)09:00,10:30;12:00;13:30
```

Every 20 minutes from 9AM to 4:40PM every day:

```
DTSTART:19970902T090000-0400
RRULE:FREQ=DAILY;BYHOUR=9,10,11,12,13,14,15,16;BYMINUTE=0,20,40
or
RRULE:FREQ=MINUTELY;INTERVAL=20;BYHOUR=9,10,11,12,13,14,15,16

==> (September 2, 1997 EDT)9:00,9:20,9:40,10:00,10:20,
                           ... 16:00,16:20,16:40
    (September 3 1997 EDT)9:00,9:20,9:40,10:00,10:20,
                          ...16:00,16:20,16:40
...
```

An example where the days generated makes a difference because of WKST:

```
DTSTART:19970805T090000-0400
RRULE:FREQ=WEEKLY;INTERVAL=2;COUNT=4;BYDAY=TU,SU;WKST=MO

==> (1997 EDT)Aug 5,10,19,24

changing only WKST from MO to SU, yields different reults...

DTSTART:19970805T090000-0400
RRULE:FREQ=WEEKLY;INTERVAL=2;COUNT=4;BYDAY=TU,SU;WKST=SU
==> (1997 EDT)August 5,17,19,31
```

4.6.27 Related To

This property is identified by the property name RELATED-TO. The property is used to represent relationships or references between one calendar component and another. The property MAY only be specified in the "VEVENT", "VTODO", "VJOURNAL" or "VFREEBUSY" calendar components. The property value consists of the persistent, globally unique

identifier of another calendar component. This value would be represented in a calendar component by the "UID" property. The property value always points to another calendar component that has a "parent" relationship to the referencing object.

A linked relationship can be specified by a series of components that each, in turn, refer to their parent component. A group relationship can be specified by a number of components that all refer to one common parent component.

Changes to a calendar component referenced by this property MAY impact the related calendar component. For example, if a group event changes its start or end date or time, then the related, dependent events will need to have their start and end dates changed in a corresponding way. This property is intended only to provide information on the relationship of calendar components. It is up to the target calendar system to maintain any property implications of this relationship.

The property is defined by the following notation:

```
  related    = "RELATED-TO" [";" [ws] paramlist] ":" [ws] uid CRLF
```

The following is an example of this property:

```
  RELATED-TO:<jsmith.part7.19960817T083000.xyzMail@host3.com>

  RELATED-TO:<19960401-080045-4000F192713-0052@host1.com>
```

The data type for this property is UID.

4.6.28 Repeat Count

This property is identified by the property name REPEAT. This property defines the number of repetitions for an alarm.

The property is defined by the following notation:

```
  repeatcnt  = "REPEAT" ":" [ws] integer CRLF
  ;Default is "1".
```

The following is an example of this property:

```
  REPEAT:4
```

The data type for the property is INTEGER.

4.6.29 Request Status

This property is identified by the property name REQUEST-STATUS. This property defines the status code returned for a scheduling request. This property is used to return status code information related to

the processing of an associated iCalendar object. The data type for this property is TEXT.

The value consists of a short return status, a longer return status description, and optionally the offending data. The components of the value are separated by the SEMICOLON character (ASCII decimal 59).

The short return status is a PERIOD character (ASCII decimal xx) separated set of integers. For example, "3.1.1". The successive

levels of integers provide for a successive level of status code granularity.

The property is defined by the following notation:

```
  Rstatus    = "REQUEST-STATUS" ":" [ws] statcode ";" [ws]
                       statdesc [";" [ws] extdata]

  Statcode   = 1*DIGIT *("." 1*DIGIT)
  ;Hierarchical, numeric return status code

  Statdesc   = *WORD
  ;Textual status description

  Extdata    = *WORD
  ;Textual exception data. For example, the offending property
  ;name and value or complete property line.
```

The following are some possible examples of this property (Note: The BACKSLASH character escapement of separator characters in a property value):

```
  REQUEST-STATUS:2.0;Success

  REQUEST-STATUS:3.1;Invalid property value;DTSTART:96-Apr-01

  REQUEST-STATUS:2.8; Success\, repeating event ignored. Scheduled
   as a single event.;RRULE\:FREQ=WEEKLY\;INTERVAL=2

  REQUEST-STATUS:4.1;Event conflict. Date/time is busy.

  REQUEST-STATUS:3.7;Invalid calendar user;ATTENDEE\:
   jsmith@host.com
```

The following are valid classes for the return status code. Individual iCalendar object methods will define specific return status codes for these classes.

Short Return Status Code	Longer Return Status Description
1.xx	Preliminary success. This class of status of status code indicates that the request has request has been initially processed but that

	completion is pending.
2.xx	Successful. This class of status code indicates that the request was completed successfuly. However, the exact status code MAY indicate that a fallback has been taken.
3.xx	Client Error. This class of status code indicates that the request was not successful. The error is the result of either a syntax or a semantic error in the client formatted request. Request should not be retried until the condition in the request is corrected.
4.xx	Scheduling Error. This class of status code indicates that the request was not successful. Some sort of error occurred within the calendaring and scheduling service, not directly related to the request itself.

4.6.30 Resources

This property is identified by the property name RESOURCES. This property defines the equipment or resources needed for the event or to-do. The property value is an arbitrary text. The property MAY only be specified in the "VEVENT" or "VTODO" calendar component. More than one resource MAY be specified as a list of resources separated by the COMMA character (ASCII decimal 44).

The property is defined by the following notation:

```
resource   = "RESOURCES" [";" [ws] paramlist] ":" [ws]
             resvalist CRLF

resvalist  = resvalue / resvalue "," [ws] resvalist

resvalue   = "CATERING" / "CHAIRS" / "COMPUTER PROJECTOR"
           / "EASEL" / "OVERHEAD PROJECTOR" / "SPEAKER PHONE"
           / "TABLE" / "TV" / "VCR" / "VIDEO PHONE" / "VEHICLE"
           / word
```

The following is an example of this property:

```
RESOURCES:EASEL,PROJECTOR,VCR
```

The data type for this property is TEXT.

4.6.31 Sequence Number

This property is identified by the property name SEQUENCE. This property defines the revision sequence of the calendar component used in a request. The property MAY only be specified in a "VEVENT",

"VTODO", "VJOURNAL" or "VFREEBUSY" calendar component. This property is needed to properly handle the receipt and processing of a sequence of calendar components that have been delivered out of order. Such is the case for store-and-forward based transports. The first request is created with a sequence number of "0" (ASCII decimal 48). It is incremented each time the ORGANIZER or OWNER issues a revision to the request. The sequence number MUST be monotonically incremented when one of the following properties in a calendar component is changed:

- "DTSTART"
- "DTEND"
- "DUE"
- "RDATE"
- "RRULE"
- "EXDATE"
- "EXRULE"

The property is defined by the following notation:

```
sequence   = "SEQUENCE" ":" [ws] integer CRLF
;Default is "0".
```

The following is an example of this property:

```
SEQUENCE:1
```

The data type for this property is INTEGER.

4.6.32 Status

This property is identified by the property name STATUS. This property defines the orignator's view of the overall status for the calendar component. This property MAY only be specified in the "VEVENT", "VTODO", "VJOURNAL" calendar components. The property is used to specify the originator's view of the general consensus for the event or the to-do. In addition, when specified in a group scheduled to-do the property is used to specify the originator's view of the completion status for the to-do. The property MAY also specify

whether the event or to-do has been cancelled.

The property is defined by the following notation:

```
  status     = "STATUS" [";" [ws] paramlist] ":" [ws] statvalue CRLF

  statvalue  = "NEEDS-ACTION"        ;Indicates to-do needs action.
             / "COMPLETED"           ;Indicates to-do completed.
             / "IN-PROCESS"          ;Indicates to-do in process of
                                     ;being completed.
             / "TENTATIVE"           ;Indicates event or to-do is
                                     ;tentative.
             / "CONFIRMED"           ;Indicates event or to-do is
                                     ;definite.
```



```
             / "CANCELLED"           ;Indicates event or to-do was
                                     ;cancelled.
```

The following is an example of this property:

```
  STATUS:TENTATIVE
```

The data type for this property is TEXT.

4.6.33 Summary

This property is identified by the property name SUMMARY. This property defines a short summary or subject for the calendar component. The property MUST be specified in the "VEVENT", "VTODO" and "VJOURNAL" calendar components. In addition, it MAY appear in the "VALARM" calendar component.

The property is defined by the following notation:

```
  summary    = "SUMMARY" [";" [ws] paramlist] ":" [ws] text CRLF
```

The following is an example of this property:

```
  SUMMARY:Department Party
```

The data type for this property is TEXT.

4.6.34 Time Transparency

This property is identified by the property name TRANSP. This property defines whether an event is transparent or not to busy time searches. This property MAY only be specified in a "VEVENT" calendar component.

The property is specified by the following notation:

```
transp     = "TRANSP" [";" [ws] paramlist] ":" [ws] transvalue CRLF

transvalue = "OPAQUE"      ;Blocks or opaque on busy time searches.
           / "TRANSPARENT" ;Transparent on busy time searches.

;Default value is OPAQUE
```

The following is an example of this property for an event that is transparent or does not block on free/busy time searches:

```
TRANSP:TRANSPARENT
```

The following is an example of this property for an event that is opaque or blocks on free/busy time searches:

```
TRANSP:OPAQUE
```


The data type for this property is TEXT.

4.6.35 Time Zone Name

This property is identified by the property name TZNAME. This property specifies the customary designation for a time zone description. This property MAY only be specified in the "VTIMEZONE" calendar component.

This property is defined by the following notation:

```
tzname     = "TZNAME" [";" [ws] paramlist] ":" [ws] text CRLF
```

The following are examples of this property:

```
TZNAME: EST
```

TZNAME: PDT

The data type for this property is TEXT.

4.6.36 Time Zone Offset

This property is identified by the property name TZOFFSET. This property specifies the offset from UTC for a time zone. This property MAY only be specified in a "VTIMEZONE" calendar component. A "VTIMEZONE" calendar component MUST include this property. The property value is a signed numeric indicating the number of hours and possibly minutes from UTC. Positive numbers represents time zones east, or ahead of UTC. Negative numbers represents time zones west of, or behind UTC.

The property is defined by the following notation:

tzoffset = "TZOFFSET" ":" [ws] utc-offset CRLF

The following are examples of this property:

TZOFFSET:-0500

TZOFFSET:+0530

The data type for this property is UTC-OFFSET.

4.6.37 Uniform Resource Locator

This property is identified by the property name URL. This property defines a Uniform Resource Locator (URL) associated with the iCalendar object. This property MAY be specified in the "VEVENT", "VTODO", "VJOURNAL", "VFREEBUSY", and "VALARM" calendar components.

The property is defined by the following notation:

url = "URL" ":" [ws] url CRLF

The following is an example of this property:

URL:http://abc.com/pub/calendars/jsmith/mytime.or3

The data type for this property is URL.

4.6.38 Unique Identifier

This property is identified by the property name UID. This property defines the persistent, globally unique identifier for the calendar component. The property MUST be specified in the "VEVENT", "VTODO" and "VJOURNAL" calendar components.

The UID itself MUST be a globally unique identifier. The generator of the identifier MUST guarantee that the identifier is unique. There are several algorithms that can be used to accomplish this. The identifier is RECOMMENDED to be the identical syntax to the [RFC 822] addr-spec. A good method to assure uniqueness is to put the domain name or a domain literal IP address of the host on which the identifier was created on the right hand side of the "@", and on the left hand side, put a combination of the current calendar date and time of day (i.e., formatted in as a DATE-TIME value) along with some other currently unique (perhaps sequential) identifier available on the system (for example, a process id number). Using a date/time value on the left hand side and a domain name or domain literal on the right hand side makes it possible to guarantee uniqueness since no two hosts should be using the same domain name or IP address at the same time. Though other algorithms will work, it is RECOMMENDED that the right hand side contain some domain identifier (either of the host itself or otherwise) such that the generator of the message identifier can guarantee the uniqueness of the left hand side within the scope of that domain.

This identifier is created by the calendar system that generates an iCalendar object. The identifier is represented as a text value. This is the method for correlating scheduling messages with the referenced event, to-do, or journal.

The property is defined by the following notation:

```
  uid        = "UID" ":" [ws] text CRLF
```

The following is an example of this property:

```
  UID:19960401T080045Z-4000F192713-0052@host1.com
```

This property is an important method for group scheduling applications to match requests with later replies, modifications or deletion requests. Calendaring and scheduling applications MUST generate this property in "VEVENT", "VTODO" and "VJOURNAL" calendar

components to assure interoperability with other group scheduling applications.

Implementations MUST be able to receive and persist values of at least 255 characters for this property.

The data type for this property is TEXT.

4.6.39 Non-standard Properties

The MIME Calendaring and Scheduling Content Type provides a "standard mechanism for doing non-standard things". This extension support is provided for implementers to "push the envelope" on the existing version of the memo. Extension properties are specified by property and/or property parameter names that have the prefix text of "X-" (the two character sequence: LATIN CAPITAL LETTER X character followed by the HYPEN-MINUS character). It is RECOMMENDED that vendors concatenate onto this sentinel another short prefix text to identify the vendor. This will facilitate readability of the extensions and minimize possible collision of names between different vendors. User agents that support this content type are expected to be able to parse the extension properties and property parameters but MAY ignore them.

The property is defined by the following notation:

```
extension  = "X-" [vendorid] "-" word [";" [ws] paramlist] ":" [ws]
             value

vendorid   = 3*char        ;Vendor identification prefix text
```

The following might be the ABC vendor s extension for an audio-clip form of subject property:

```
X-ABC-MMSUBJ;TYPE=WAVE; VALUE=URL: http://load.noise.org/mysubj.wav
```

At present, there is no registration authority for names of extension properties and property parameters. The data type for this property is TEXT. Optionally, the data type MAY be any of the other valid data types.

5. Recommended Practices

These recommended practices should be followed in order to assure consistent handling of the following cases for an iCalendar object.

1. A calendar entry with a "DTSTART" property but no "DTEND" property - The event does not take up any time. It is intended to represent an event that is associated with a given calendar date and time of day, such as an anniversary. Since the event does not take up any time, it MUST NOT be used to record busy time no matter what the value for the "TRANSP" property.

2. When the "DTSTART" and "DTEND", for "VEVENT", "VJOURNAL" and "VFREEBUSY" calendar components, and "DTSTART" and "DUE", for "VTODO" calendar components, have the same value data type (e.g., DATE-TIME), they should specify values in the same time format (e.g., local time with UTC offset).

3. A combination of "RRULE" and "RDATE" properties that produces more than one instance for a given date/time - Only one recurrence can occur on a given date/time interval. Just one instance for the date/time is recorded.

4. A particular iCalendar object method that specifies "ATTENDEE" properties with the "MEMBER" property parameter, for which the recipient has multiple memberships - Recipient should reply to only the first "MEMBER" property parameter value that it can match.

6. Registration of Content Type Elements

This section provide the process for registration of MIME Calendaring and Scheduling Content Type iCalendar object methods and new or modified properties.

6.1 Registration of New and Modified iCalendar object Methods

New MIME Calendaring and Scheduling Content Type iCalendar object methods are registered by the publication of an IETF Request for Comment (RFC). Changes to an iCalendar object method are registered by the publication of a revision of the RFC defining the method.

6.2 Registration of New Properties

This section defines procedures by which new properties or enumerated property values for the MIME Calendaring and Scheduling Content Type can be registered with the IANA. Note that non-IANA properties MAY be used by bilateral agreement, provided the associated properties names follow the "X-" convention.

The procedures defined here are designed to allow public comment and review of new properties, while posing only a small impediment to the definition of new properties.

Registration of a new property is accomplished by the following steps.

6.2.1 Define the property

A property is defined by completing the following template.

To: ietf-calendar@imc.org

Subject: Registration of text/calendar MIME property XXX

Property name:

Property purpose:

Property data type(s):

Property encoding:

Property special notes (optional):

Intended usage: (one of COMMON, LIMITED USE or OBSOLETE)

The meaning of each field in the template is as follows.

Property name: The name of the property, as it will appear in the body of an text/calendar MIME Content-Type "property: value" line to the left of the colon ":".

Property purpose: The purpose of the property (e.g., to indicate a delegate for the event or to-do, etc.). Give a short but clear description.

Property data type(s): Any of the valid data types for the property

value needs to be specified. The default data type also needs to be specified. If a new data type is specified, it needs to be declared in this section.

Property encoding: The encodings permitted for the property value. This description MUST be precise and MUST NOT violate the general encoding rules defined in this memo.

Property special notes: Any special notes about the property, how it is to be used, etc.

6.2.2 Post the Property definition

The property description MUST be posted to the new property discussion list, ietf-calendar@imc.org.

6.2.3 Allow a comment period

Discussion on the new property MUST be allowed to take place on the list for a minimum of two weeks. Consensus MUST be reached on the property before proceeding to the next step.

6.2.4 Submit the property for approval

Once the two-week comment period has elapsed, and the proposer is convinced consensus has been reached on the property, the registration application should be submitted to the Method Reviewer for approval. The Method Reviewer is appointed to the Application Area Directors and MAY either accept or reject the property registration. An accepted registration should be passed on by the

Method Reviewer to the IANA for inclusion in the official IANA method registry. The registration MAY be rejected for any of the following reasons. 1) Insufficient comment period; 2) Consensus not reached; 3) Technical deficiencies raised on the list or elsewhere have not been addressed. The Method Reviewer's decision to reject a property MAY be appealed by the proposer to the IESG, or the objections raised can be addressed by the proposer and the property resubmitted.

6.3 Property Change Control

Existing properties MAY be changed using the same process by which they were registered.

1. Define the change

2. Post the change

3. Allow a comment period

4. Submit the property for approval

Note that the original author or any other interested party MAY propose a change to an existing property, but that such changes should only be proposed when there are serious omissions or errors in the published memo. The Method Reviewer MAY object to a change if it is not backwards compatible, but is not required to do so.

Property definitions can never be deleted from the IANA registry, but properties which are no longer believed to be useful can be declared OBSOLETE by a change to their "intended use" field.

7. File extension

The file extension of "vcs" is to be used to designate a file containing calendaring and scheduling information consistent with this MIME content type.

8. Macintosh File Type Code

The file type code of "vcal" is to be used in Apple MacIntosh operating system environments to designate a file containing calendaring and scheduling information consistent with this MIME media type.

9. References

The following document are referred to within this memo.

[ICMS] "Internet Calendaring Model Specification", Internet-Draft, July 1997, ftp://ftp.ietf.org/internet-drafts/draft-ietf-calsch-mod-00.txt.

[IMIP] "iCalendar Message-based Interoperability Protocol (IMIP)", Internet Draft, October 1997, http://www.imc.org/draft-ietf-calsch-imip-02.txt.

[ISO 8601] ISO 8601, "Data elements and interchange formats- - Information interchange--Representation of dates and times", International Organization for Standardization, June, 1988. This standard is also addressed by the Internet Draft document ftp://ds.internic.net/internet-drafts/draft-newman-datetime-00.txt.

[ISO 9070] ISO/IEC 9070, "Information Technology- -SGML Support Facilities--Registration Procedures for Public Text Owner Identifiers", Second Edition, International Organization for Standardization, April, 1991.

[ITIP] "iCalendar Transport-Independent Interoperability Protocol (iTIP) : Scheduling Events, Busy Time, To-dos and Journal Entries ", Internet-Draft, October 1997, http://www.imc.org/draft-ietf-calsch-itip-01.txt.

[MIME DIR] Howes, T., Smith, M., "A MIME Content-Type for Directory Information", Internet-draft-ietf-asid-mime-direct-06.txt, July, 1997.

[RFC 822] Crocker, D., "Standard for the Format of ARPA Internet Text Messages", STD 11, RFC 822, August 1982.

[RFC 1738] Berners-Lee, T., Masinter, L., McCahill, M., "Uniform Resource Locators (URL)", RFC 1738, December 1994.

[RFC 1766] Alvestrand, H., "Tags for the Identification of Languages", March 1995.

[RFC 1872] Levinson, E., "The MIME Multipart/Related Content-type," RFC 1872, December 1995.

[RFC1983] "Internet Users' Glossary", RFC 1983, August 1996.

[RFC 2045] Freed, N., Borenstein, N., " Multipurpose Internet Mail Extensions (MIME) - Part One: Format of Internet Message Bodies", RFC 2045, November 1996.

[RFC 2046] Freed, N., Borenstein, N., " Multipurpose Internet Mail Extensions (MIME) - Part Two: Media Types", RFC 2046, November 1996.

[RFC 2047] Moore, K., "Multipurpose Internet Mail Extensions (MIME) - Part Three: Message Header Extensions for Non-ASCII Text", RFC 2047,

November 1996.

[RFC 2048] Freed, N., J. Klensin, J. Postel, "Multipurpose Internet Mail Extensions (MIME) - Part Four: Registration Procedures", RFC 2048, January 1997.

[RFC2119] "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, March 1997.

[UTF-8] "UTF-8, a transformation format of Unicode and ISO 10646",Internet-Draft, July,1996, ftp://ftp.ietf.org/internet-drafts/draft-yergeau-utf8-01.txt.

[VCARD] Internet Mail Consortium, "vCard - The Electronic Business Card Version 2.1", http://www.versit.com/pdi/vcard-21.txt, September 18, 1996.

[VCAL] Internet Mail Consortium, "vCalendar - The Electronic Calendaring and Scheduling Exchange Format", http://www.imc.org/pdi/vcal-10.txt, September 18, 1996.

[XAPIA] "XAPIA CSA, Calendaring and Scheduling Application Programming Interface (CSA) Version 1.0", X.400 API Association, November 15, 1994.

10. Acknowledgments

A hearty thanks to the IETF Calendaring and Scheduling Working Group and also the following individuals who have participated in the drafting, review and discussion of this memo:

Roland Alden, Harald T. Alvestrand, Eric Berman, Denis Bigorgne, John Binici, Bill Bliss, Philippe Boucher, Steve Carter, Andre Courtemanche, Dave Crocker, Alec Dun, John Evans, Ross Finlayson, Randell Flint, Ned Freed, Patrik Falstrom, Chuck Grandgent, Mark Handley, Steve Hanna, Paul B. Hill, Paul Hoffman, Ross Hopson, Mark Horton, Bruce Kahn, C. Harald Koch, Don Lavange, Theodore Lorek, Steve Mansour, Skip Montanaro, Keith Moore, Cecil Murray, Chris Newman, John Noerenberg, Ralph Patterson, Pete Resnick, Keith Rhodes, Robert Ripberger, John Rose, Andras Salamar, Ted Schuh, Vinod Seraphin, Derrick Shadel, Ken Shan, Andrew Shuman, Steve Silverberg, William P. Spencer, John Sun, Mark Towfiq, Robert Visnov, James L.

Weiner, Mike Weston, William Wyatt.

11. Copyright

12. Author's Address

The following address information is provided in a MIME-VCARD, Electronic Business Card, format.

The authors of this draft are:

```
BEGIN:VCARD
FN:Frank Dawson
ORG:Lotus Development Corporation
ADR;WORK;POSTAL;PARCEL:;;6544 Battleford Drive;
```



```
 Raleigh;NC;27613-3502;USA
TEL;WORK;MSG:+1-919-676-9515
TEL;WORK;FAX:+1-919-676-9564
EMAIL;INTERNET:fdawson@earthlink.net
URL:http://home.earthlink.net/~fdawson
END:VCARD
```

```
BEGIN:VCARD
FN:Derik Stenerson
ORG:Microsoft Corporation
ADR;WORK;POSTAL;PARCEL:;;One Microsoft Way;
 Redmond;WA;98052-6399;USA
TEL;WORK;MSG:+1-425-936-5522
TEL;WORK;FAX:+1-425-936-7329
EMAIL;INTERNET:deriks@Microsoft.com
END:VCARD
```

The iCalendar object is a result of the work of the Internet Engineering Task Force Calendaring and Scheduling Working Group. The chairman of that working group is:

```
BEGIN:VCARD
FN:Anik Ganguly
```

```
ORG:OnTime, Inc.
ADR;WORK;POSTAL;PARCEL:10 Floor;;21700 Northwestern Highway;
 Southfield;MI;48075;USA
TEL;WORK;MSG:+1-248-559-5955
TEL;WORK;FAX:+1-248-559-5034
EMAIL;INTERNET:anik@ontime.com
END:VCARD
```

13. iCalendar object Examples

The following examples are provided as an informational source of illustrative iCalendar objects consistent with this content type.

The following iCalendar object is specified as the content of a MIME message. The example demonstrates a possible meeting request between the originator and recipient of the message.

```
TO:jsmith@host1.com
FROM:jdoe@host1.com
MIME-VERSION:1.0
MESSAGE-ID:<id1@host1.com>
CONTENT-TYPE:text/calendar;method=request

BEGIN:VCALENDAR
METHOD:REQUEST
PRODID:-//xyz Corp//NONSGML PDA Calendar Verson 1.0//EN
VERSION:2.0
BEGIN:VEVENT
```



```
DTSTAMP:19960704T120000Z
DTSTART:19960918T143000Z
DTEND:19960920T220000Z
CATEGORIES:CONFERENCE,PROJECT
SUMMARY:Networld+Interop Conference
DESCRIPTION:Networld+Interop Conference
  and Exhibit\nAtlanta World Congress Center\n
Atlanta, Georgia
END:VEVENT
END:VCALENDAR
```

The following example message issues a meeting request that does not require any reply. The message is sent as a singular "text/calendar" content type, body part.

```
From: jsmith@host1.com
To: ietf-calendar@imc.org
Subject: First IETF-Calendar Working Group Meeting
MIME-Version: 1.0
Message-ID: <id2@host1.com>
Content-Type: text/calendar;method=request

BEGIN:VCALENDAR
METHOD:REQUEST
PRODID:-//RDU Software//NONSGML HandCal//EN
VERSION:2.0
BEGIN:VEVENT
DTSTAMP:19961022T133000Z
ATTENDEE;EXPECT=REQUEST:ietf-calendar@imc.org
DESCRIPTION:First IETF-Calendaring and Scheduling Working Group
  Meeting
CATEGORIES:MEETING
CLASS:PUBLIC
CREATED:19961022T083000
SUMMARY:IETF Calendaring Working Group Meeting
DTSTART:19961210T210000Z
DTEND:19961210T220000Z
LOCATION:San Jose, CA - Fairmont Hotel
UID:guid-1.host1.com
END:VEVENT
END:VCALENDAR
```

The following is an example of a MIME message with a single body part consisting of a text/calendar content type. The message specifies a meeting request between the originator and recipient of the message.

```
TO:jsmith@host1.com
FROM:jdoe@host1.com
MIME-VERSION:1.0
MESSAGE-ID:<id3@host1.com>
CONTENT-TYPE:text/calendar;method=request
```



```
BEGIN:VCALENDAR
METHOD:REQUEST
VERSION:2.0
```

```
  PRODID:-//ABC Corporation//NONSGML My Product//EN
  BEGIN:VEVENT
  DTSTAMP:19970324T1200Z
  SEQUENCE:0
  UID:19970324T080045Z-4000F192713-0052@host1.com
  ATTENDEE;EXPECT=REQUEST:jsmith@host1.com
  DTSTART:19970324T123000Z
  DTEND:19970324T210000Z
  CATEGORIES:CONFERENCE,PROJECT
  CLASS:PUBLIC
  SUMMARY:Calendaring Interop Conference
  DESCRIPTION:Calendaring Interop Conference and Exhibit\n
   Atlanta, Georgia
  LOCATION:Atlanta World Congress Center
  ATTACH;VALUE=URL:file:///xyzCorp.com/conf/bkgrnd.ps
  END:VEVENT
  END:VCALENDAR
```

Example of a reply to the above request, accepting the meeting.

```
  TO:jdoe@host1.com
  FROM:jsmith@host1.com
  MIME-VERSION:1.0
  MESSAGE-ID:<id4@host1.com>
  CONTENT-TYPE:text/calendar;method=reply

  BEGIN:VCALENDAR
  METHOD:REPLY
  VERSION:2.0
  PRODID:-//ABC Corporation//NONSGML My Product//EN
  BEGIN:VEVENT
  DTSTAMP:19970324T120000Z
  SEQUENCE:0
  UID:19970324T080045Z-4000F192713-0052@host1.com
  ATTENDEE;STATUS=ACCEPTED;EXPECT=REQUEST:jsmith@host1.com
  END:VEVENT
  END:VCALENDAR
```

An example of a meeting cancelation:

```
  TO:jsmith@host1.com
  FROM:jdoe@host1.com
  MIME-VERSION:1.0
  MESSAGE-ID:<id5@host1.com>
  CONTENT-TYPE:text/calendar;method=cancel

  BEGIN:VCALENDAR
  METHOD:CANCEL
  VERSION:2.0
  PRODID:-//ABC Corporation//NONSGML My Product//EN
  BEGIN:VEVENT
```

```
DTSTAMP:19970324T120000Z
UID:19970324T080045Z-4000F192713-0052@host1.com
END:VEVENT
END:VCALENDAR
```

14. Full Copyright Statement