Network Working Group — T. Hare
Internet-Draft — Individual
Expires: November 17, 2005 — May 16, 2005

Guideline for use of XML with iCalendar elements
draft-hare-xcalendar-03

Status of this Memo

Copyright Notice

Abstract

This memo defines a guideline for using XML to represent calendaring information that corresponds to the iCalendar, Internet Calendaring and Scheduling Core Object Specification defined by [RFC 2445] and the protocols defined by [RFC2446], [RFC2447] and [CAP]. This memo applies to all [RFC 2445] extensions and modifications.

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

Table of Contents

1. Introduction

The Extended Markup Language (XML) as defined in [XML] has gained widespread attention as a "web friendly" syntax for representing and exchanging documents and data on the Internet. This interest includes requests for and discussion of possible document type definitions (DTD) and name-spaces for IETF standard   formats such as that defined by [RFC 2445], [RCF 2446], and [RFC 2447].

This memo defines how XML can be used to represent iCalendar objects,and how iCalendar namespaces can be used with other XML documents. An example DTD is provided although use of a DTD is not required.

This memo does not try to enforce any specific one-to-one mapping between XML   objects and iCalendar objects, but instead attempts to document the method whereby XML developers can provide interoperability with iCalendar.

NOTE: The [RFC 2445] is the definitive reference for the definition of iCalendar semantics. This memo only provides a guideline for representing such semantics in XML. This memo does not introduce any new semantics for items already defined by [RFC 2445]. [RFC 2446], [RFC 2447], and [CAP] are the references for protocols   for the exchange of iCalendar objects. This memo does not introduce any new protocols or functions beyond those in the respective documents.

An attempt has been made to guide the developer in use of XML to represent iCalendar semantics, allowing XML-based applications to make use of the iCalendar [RFC2445] and related protocols [RFC2446], [RFC2447], and [CAP] semantics and to provide interoperability between XML-based applications and iCalendar-compliant   applications.

The publication of XML version 1.0 was followed by publication of two World-wide Web Consortium (W3C) recommendations relevant to this memo. The first was a recommendation on "Namespaces in XML" and the other was a recommendation on "Extensible Stylesheet Language" and "Extensible Stylesheet Language Transformation" (XSL and XSLT). An XML name-space is a collection of names, identified by a URI. An XSL transformation(XSLT)is a document in the Extensible Stylesheet Language (XSL) which provides a method for transforming an XML document into some other form. In anticipation of the use of XML namespaces, this memo includes the definition of URIs to be used to identify the namespaces for iCalendar [RFC 2445], iTIP [RFC 2446], iMIP [RFC 2447] and CAP elements. XML applications that conform to this memo and also use namespaces MAY include other   non-iCalendar namespaces.

Documents MAY include a Document Type Definition (DTD), an XML schema, or may reference external versions of either. This memo allows documents containing   iCalendar XML objects to be constructed with either. DTDs and Schemas are outside   the scope of this memo. A document containing a DTD or schema MAY include   definitions for calendar elements. Any document conforming to this memo MUST provide an XSL transformation which will render those calendar elements into   standard iCalendar/iTIP/iMIP/CAP (as appropriate) elements.

2. Using XML For Representing iCalendar

XML is a simplified version of the text markup syntax defined by ISO 8879, Standard Generalized Markup Language (SGML). XML was published as a proposed recommendation [XML] by the World-wide Web Consortium (W3C) on February 10, 1998.

3. XML and XSL Dependencies

This memo specifies the XML representation for the standard iCalendar elements defined by [RFC 2445], [RFC 2446], [RFC2447], and [CAP]. There are no XML dependencies other than the [XML] and the [XMLNS] recommendations.

This memo requires that conforming documents include a reference to an [XSL] stylesheet for transforming the document into standard iCalendar format. How the transformation is done is left to the implementor. Providing an XSL transform into iCalendar objects does not preclude providing other transforms.

4. Working With Standard and XML iCalendar Representations

This memo provides a guideline for using alternative, XML representations for the standard iCalendar elements defined in [RFC 2445]. These alternative representations SHOULD provide the same semantics as that defined in the standard format. It is the goal of this memo to allow all [RFC 2445] extensions and modifications to be translated into and from this XML format.

5. Conversion

This memo requires any compliant document to be transformable into standard iCalendar information. It is recognized that such conversion MAY be asymmetric, since compliant documents MAY include information which is not representable in iCalendar and which would be lost during any "round trip" conversions. This does not preclude implementation of "round-trippable" transformations, but they are not required.

To formalize and standardize the interchange of iCalendar information through XML, each conforming document MUST include reference to an XSL stylesheet which can transform the document into a standard iCalendar [RFC 2445] document of MIME content-type "text/calendar".

6. Mixed Use of Both Representations

As previously indicated, conversion between the XML and standard representations of iCalendar is a straightforward process using XSL transformations. In addition, mixed use of both representations is also possible using MIME objects.

While MIME multipart content-types can be used to provide a mix of both the standard and XML representations, this is NOT required. Instead, each document MUST include a reference to an XSL stylesheet which can transform the XML representation into standard iCalendar (and possibly, iTIP, iMIP, or CAP) syntax.

With the use of the MIME multipart content-types, compound MIME entities containing a mix of the standard and XML representations can be specified. Internet applications conforming to this memo MAY send both the standard and XML representation of the iCalendar objects, to provide compatibility with Internet applications which cannot process the required XSL transformation.

7. Using Data Types

Strong "data typing" is an integral design principle to the iCalendar format. Strong data typing in iCalendar means that the format type for each property   value is well known. Within [RFC 2445], the data type is called the "value type". The standard format defined by [RFC 2445] specifies a default value type for each calendar and component property. In addition, many of the property definitions allow for the specification of alternate value types. The required XSL transformation in this memo MUST create iCalendar elements with proper types. Consult iCalendar [RFC 2445] for documentation of value types.

8. Namespaces

[XMLNS] defines "Namespaces in XML" to be a collection of names, identified by a URI, which are used in XML documents as element types and attribute names. The [XML] specification does not include a definition for namespaces, but does set down some guidelines for experimental naming of namespaces.

XML namespaces allow multiple markup vocabulary in a single document. Documents and applications conforming to this memo MAY use multiple namespaces with the iCalendar, iTIP, iMIP, and CAP namespaces. Multiple namespaces MUST be used for the different iCalendar and protocol elements. This requirement is intended to provide clarity in the document, by discriminating calendar object elements from protocol elements.

The document at the namespace URI does not contain any defintions but serves as a unique identifier to allow specification of different namespaces. Applications complying with this memo MUST use the following URIs for namespace elements:

iCalendar:	http://www.ietf.org/rfc/rfc2445.txt
iTIP:	http://www.ietf.org/rfc/rfc2446.txt
iMIP:	http://www.ietf.org/rfc/rfc2447.txt
CAP:	http://www.ietf.org/internet-drafts/ draft-ietf-calsch-cap-13.txt

NOTE: the URI for CAP will be replaced with the RFCxxxx reference when CAP is completed.

The following is an example of a well-formed but invalid "xdoc" document type that includes elements and attribute lists from the iCalendar and iTIP namespaces.

```
                    <?xml version="1.0" encoding="UTF-8"?>
                    <xdoc>
                      <iCal:vcalendar iTIP:method="publish"
                              xmlns:iCal="http://www.ietf.org/rfc/RFC2445.txt"
                              xmlns:iTIP="http://www.ietf.org/rfc/RFC2446.txt">
                      <iCal:vevent>
                      <!--- VEVENT elements go here -->
                      </iCal:vevent>

                      <!-- Remainder of the XML document, each element from the -
->
                      <!-- iCalendar namespace with the "iCal:" prefix. -->

                    </xdoc>
```

The semantics of the "xmlns" attribute, and any attribute with "xmlns:" as a prefix, is as specified in [XMLNS]. It is used to declare a namespace in XML.

iCalendar provides for "experimental" elements. These elements are represented in iCalendar with element names beginning with "X-". Any experimental element in a document which conforms to this memo MUST be represented by a namespace different than those used for iCalendar, iTIP, iMIP, or CAP. This requirement is intended to simplify implementation of extensions and experimental items.

9. Emailing documents with iCalendar XML Representation

It is expected that iCalendar XML documents will need to be sent over SMTP/MIME email. The "text/xml" and "application/xml" content-types have been registered for XML documents.

All documents conforming to this memo SHOULD be sent as content-type "text/xml" or "application/xml". When iCalendar elements may be mixed with others, it is not   practical for an MUA to determine, without opening the document, if iCalendar XML elements exist within the document.

If a part of a MIME multi-part message contains only XML-represented iCalendar objects, and it is wished to provide the ability for an application to determine content based upon the MIME headers, the content-types "text/xml+calendar" or   "application/xml+calendar" MAY be used.

Internet applications conforming to this memo MUST include in any iCalendar XML document that is sent, the XSL stylesheet reference to be used to provide transformation from the XML representation to the standard representation. This   restriction guarantees that a standard iCalendar object can be produced from the iCalendar XML document.

Internet applications conforming to this memo MAY send the iCalendar XML document in a "multipart/alternative" MIME entity that also contains an equivalent iCalendar object in the standard format defined by [RFC 2445], to provide compatibility with applications which cannot process XML or XSL transformations.

An XML application comforming to the guidelines in this memo MUST be able to receive and properly process the "application/xml" document contained within a   "multipart" message content-type, and MUST be capable of performing the XSL transformation of the iCalendar elements of the document into a standard iCalendar document.

10. iCalendar XML Representation and File Systems

The iCalendar XML documents will be stored in file systems. The accepted practice for file extensions for XML documents is the text "XML". However, IF a document contains only XML representations of iCalendar data, then for file association with applications that can directly process this document type, it is RECOMMENDED that the file extension be the text "xcs".

11. Example Usage

The following sections provide various examples of documents using iCalendar elements in XML.

11.1 Example DTD

The following is a DTD which can be used to represent iCalendar [RFC 2445] objects. While this DTD represents the iCalendar objects as currently defined, this document does not imply that this is the only way to represent them, nor that this is the best DTD to do so. This is provided as an example, only.

```
<?xml version="1.0" encoding="UTF-8"?>

   <!-- ******************* -->
   <!-- Entity declarations -->
   <!-- ******************* -->

   <!ENTITY % attr.altrep "
   altrep ENTITY #IMPLIED
   ">

   <!ENTITY % attr.cn "
   cn CDATA ''
   ">

   <!ENTITY % attr.cutype "
   cutype NMTOKEN 'INDIVIDUAL'
   ">
   <!-- Valid name tokens are "INDIVIDUAL", "GROUP", "RESOURCE" -->
   <!-- "ROOM", "UNKNOWN", a non-standard "X-" name or another -->
   <!-- IANA registered name. -->

   <!ENTITY % attr.delegated-from "
   delegated-from CDATA #IMPLIED
   ">
   <!-- delegated-from value is a calendar user address -->

   <!ENTITY % attr.delegated-to "
   delegated-to CDATA #IMPLIED
   ">
   <!-- delegated-to value is one or more calendar user addresses -->

   <!ENTITY % attr.dir "
   dir ENTITY #IMPLIED
   ">
   <!-- dir value is a URI to a directory entry -->
```

```
      <!ENTITY % attr.fmttype "
      fmttype CDATA #REQUIRED
      ">
      <!-- fmttype value is any IANA registered content type -->

      <!ENTITY % attr.fbtype "
      fbtype NMTOKEN 'BUSY'
      ">

      <!-- Valid token values are "FREE", "BUSY", "BUSY-UNAVAILABLE", -->
      <!-- "BUSY-TENTATIVE", a non-standard "X-" name or another -->
      <!-- IANA registered name. -->

      <!ENTITY % attr.language "
      language CDATA #IMPLIED
      ">
      <!-- language value is a valid RFC 1766 language string -->

      <!ENTITY % attr.member "
      member CDATA #IMPLIED
      ">
      <!-- member value is one or more calendar user addresses -->

      <!ENTITY % attr.partstat "
      partstat NMTOKEN 'NEEDS-ACTION'
      ">
      <!-- Valid token value for VEVENT: "NEEDS-ACTION", "ACCEPTED", -->
      <!-- "DECLINED", "TENTATIVE", "DELEGATED", a non-standard "X- -->
      <!-- name or another IANA registered name. -->

      <!-- Valid token value for VTODO: "NEEDS-ACTION", "ACCEPTED", -->
      <!-- "DECLINED", "TENTATIVE", "DELEGATED", "COMPLETED", -->
      <!-- "IN-PROGRESS, a non-standard "X- name or another IANA -->
      <!-- registered name. -->
      <!-- Valid token value for VJOURNAL: "NEEDS-ACTION", "ACCEPTED", -->
      <!-- "DECLINED", a non-standard "X- name or another IANA -->
      <!-- registered name. -->

      <!ENTITY % attr.range "
      range NMTOKEN 'THISONLY'
      ">
      <!-- Valid token values are "THISONLY" or "THISANDPRIOR" or -->
      <!-- "THISANDFUTURE" -->

      <!ENTITY % attr.related "
      related NMTOKEN 'START'
      ">
      <!-- Valid token values are "START" or "END" -->
```

```
<!ENTITY % attr.reltype "
reltype NMTOKEN 'PARENT'
">
<!-- Valid token values are "PARENT", "CHILD", SIBLING", -->
<!-- a non-standard "X-" name or any IANA registered name. -->

<!ENTITY % attr.role "
role NMTOKEN 'REQ-PARTICIPANT'
">

<!-- Valid token values are "CHAIR", "REQ-PARTICIPANT", -->
<!-- "OPT-PARTICIPANT", "NON-PARTICIPANT", a non-standard "X-" -->
<!-- name or any IANA registered name. -->

<!ENTITY % attr.rsvp "
rsvp NMTOKEN 'FALSE'
">
<!-- Valid token values are "TRUE" or "FALSE", -->

<!ENTITY % attr.sent-by "
sent-by CDATA #IMPLIED
">
<!-- sent-by value is a calendar user address -->

<!ENTITY % attr.tzid "
tzid CDATA #IMPLIED
">
<!-- tzid value is a time zone identifier -->

<!ENTITY % cal.comp "
vevent | vtodo | vjournal | vfreebusy | vtimezone
">

<!ENTITY % vevent.opt1 "
class | created | description | dtstamp | dtstart | geo |
last-modified | location | organizer | priority | recurrence-id |
sequence | status | summary | transp | uid | url |
(dtend | duration)
">
<!-- These properties may only appear once in a VEVENT -->
<!ENTITY % vevent.optm "
attach | attendee | categories | comment | contact |
exdate | exrule | rdate | related-to | resources | request-status |
rrule
">
<!-- These properties may appear one or more times in a VEVENT -->

<!ENTITY % vtodo.opt1 "
```

```
      class | completed | created | description | dtstamp | dtstart |
      geo | last-modified | location | organizer | percent | priority |
      recurrence-id | sequence | status | summary | uid | url |
      (due | duration)
      ">
      <!-- These properties may only appear once in a VTODO -->

      <!ENTITY % vtodo.optm "
      attach | attendee | categories | comment | contact |
      exdate | exrule | request-status | related-to | resources |
      rdate | rrule

      ">
      <!-- These properties may appear one or more times in a VTODO -->

      <!ENTITY % vjournal.opt1 "
      class | created | description | dtstart | dtstamp | last-modified |
      organizer | recurrence-id | sequence | status | summary | uid | url
      ">
      <!-- These properties may only appear once in a VJOURNAL -->

      <!ENTITY % vjournal.optm "
      attach | attendee | categories | comment | contact |
      exdate | exrule | related-to | rdate | rrule | request-status
      ">
      <!-- These properties may appear one or more times in a VJOURNAL -->

      <!ENTITY % vfreebusy.opt1 "
      contact | dtstamp | dtstart | dtend | duration |
      organizer | uid | url
      ">
      <!-- These properties may only appear once in a VFREEBUSY -->

      <!ENTITY % vfreebusy.optm "
      attendee | comment | freebusy | request-status
      ">
      <!-- These properties may appear one or more times in a -->
      <!-- VFREEBUSY -->

      <!ENTITY % vtimezone.man "
      tzid
      ">
      <!-- These properties must appear in a VTIMEZONE -->

      <!ENTITY % vtimezone.opt1 "
      last-modified | tzurl
      ">
      <!-- These properties may only appear once in a VTIMEZONE -->
      <!ENTITY % vtimezone.mann "
      (standard | daylight), (standard | daylight)*
      ">
      <!-- These properties must appear in a VTIMEZONE and may -->
      <!-- appear multiple times -->

      <!ENTITY % standard.man "
      dtstart | tzoffsetto | tzoffsetfrom
      ">
      <!-- These properties must appear in a STANDARD, but only once -->

      <!ENTITY % standard.optm "

      comment | rdate | rrule | tzname
      ">
      <!-- These properties may appear one or more times in a STANDARD -->

      <!ENTITY % daylight.man "
      dtstart | tzoffsetto | tzoffsetfrom
      ">
      <!-- These properties must appear in a DAYLIGHT, but only once -->

      <!ENTITY % daylight.optm "
      comment | rdate | rrule | tzname
      ">
      <!-- These properties may appear one or more times in a DAYLIGHT -->

      <!ENTITY % audio.man "
      action, trigger
      ">
      <!-- These properties must appear in an audio VALARM. -->

      <!ENTITY % audio.optx "
      duration | repeat
      ">
      <!-- These properties may appear once in an audio VALARM. If one -->
      <!-- appears, then both must appear. -->

      <!ENTITY % audio.opt1 "
      attach
      ">
      <!-- These properties may appear once in an audio VALARM. -->

      <!ENTITY % valarm.audio "
      (%audio.man;), (%audio.optx;)*, (%audio.opt1;)
      ">

      <!ENTITY % display.man "
      action, description, trigger
      ">
      <!-- These properties must appear in a display VALARM. -->
      <!ENTITY % display.optx "
      duration | repeat
      ">
      <!-- These properties may appear once in a display VALARM. If -->
      <!-- one appears, then both must appear. -->

      <!ENTITY % valarm.display "
      (%display.man;), (%display.optx;)*
      ">

      <!ENTITY % email.man "

      action, description, summary, trigger
      ">
      <!-- These properties must appear in an email VALARM. -->

      <!ENTITY % email.optx "
      duration | repeat
      ">
      <!-- These properties may appear once in an email VALARM. If one -->
      <!-- appears, then both must appear. -->

      <!ENTITY % email.optm "
      attach
      ">
      <!-- These properties may appear one or more times in an email -->
      <!-- VALARM. -->

      <!ENTITY % email.mann "
      attendee
      ">
      <!-- These properties must appear in an email VALARM. The may -->
      <!-- appear more than once. -->

      <!ENTITY % valarm.email "
      (%email.man;), (%email.optx;)*, (%email.optm;)*,
      (%email.mann;)*
      ">

      <!ENTITY % procedure.man "
      action, attach, trigger
      ">
      <!-- These properties must appear in an audio VALARM. -->

      <!ENTITY % procedure.optx "
      duration | repeat
      ">
      <!-- These properties may appear once in an procedure VALARM. -->
      <!-- If one appears, then both must appear. -->

      <!ENTITY % procedure.opt1 "
      description
      ">
      <!-- These properties may appear once in a procedure VALARM -->
      <!ENTITY % valarm.procedure "
      (%procedure.man;), (%procedure.optx;)*, (%procedure.opt1;)?
      ">

      <!-- ******************************************** -->
      <!-- iCalendar value type notation declarations   -->
      <!-- ******************************************** -->

      <!-- NOTE: The "XCAL" text in the following NOTATION values
      will be replaced with the text "RFC xxxx", where "xxxx" is the RFC
      number, when this memo is published as a RFC. -->

      <!NOTATION BINARY PUBLIC "-//IETF//NOTATION XCAL/Value Type/Binary//EN">

      <!NOTATION BOOLEAN PUBLIC "-//IETF//NOTATION XCAL/Value Type/Boolean//EN">

      <!NOTATION CALADR PUBLIC "-//IETF//NOTATION XCAL/Value Type/Calendar
      User Address//EN">

      <!NOTATION DATE PUBLIC "-//IETF//NOTATION XCAL/Value Type/Date//EN">

      <!NOTATION DATE-TIME PUBLIC "-//IETF//NOTATION XCAL/Value
      Type/Date-Time//EN">

      <!NOTATION DURATION PUBLIC "-//IETF//NOTATION XCAL/Value
      Type/Duration//EN">

      <!NOTATION FLOAT PUBLIC "-//IETF//NOTATION XCAL/Value Type/Float//EN">

      <!NOTATION INTEGER PUBLIC "-//IETF//NOTATION XCAL/Value Type/Integer//EN">

      <!NOTATION PERIOD PUBLIC "-//IETF//NOTATION XCAL/Value
      Type/Period of Time//EN">

      <!NOTATION RECUR PUBLIC "-//IETF//NOTATION XCAL/Value
      Type/Recurrence Rule//EN">

      <!NOTATION TEXT PUBLIC "-//IETF//NOTATION XCAL/Value Type/Text//EN">
```

```
   <!NOTATION TIME PUBLIC "-//IETF//NOTATION XCAL/Value Type/Time//EN">

   <!NOTATION URI PUBLIC "-//IETF//NOTATION XCAL/Value Type/URI//EN">

   <!NOTATION UTC-OFFSET PUBLIC "-//IETF//NOTATION XCAL/Value
   Type/UTC-Offset//EN">

   <!NOTATION X-NAME PUBLIC "-//IETF//NOTATION XCAL/Value Type/X-Name//EN">

   <!-- ************************************************* -->
   <!-- iCalendar property element/attribute declarations -->
   <!-- ************************************************* -->

   <!ELEMENT br EMPTY>
   <!-- Signifies a new line in the TEXT value content information -->

   <!-- Description component properties element type declarations -->

   <!ELEMENT attach (extref | b64bin)>
   <!-- extref holds a reference to an external entity that -->
   <!-- has the attachment. b64bin holds the inline BASE64 encoded -->
   <!-- binary data for the attachment as defined in RFC 2045. -->

   <!ELEMENT extref EMPTY>
   <!ATTLIST extref
   uri ENTITY #REQUIRED>

   <!ELEMENT b64bin (#PCDATA)>
   <!ATTLIST b64bin
   %attr.fmttype;
   value NOTATION (BINARY) #IMPLIED>

   <!ELEMENT categories (item)*>

   <!ELEMENT item (#PCDATA)>
   <!ATTLIST item
   %attr.language;
   value NOTATION (TEXT) #IMPLIED>

   <!ELEMENT class (#PCDATA)>
   <!ATTLIST class
   %attr.language;
   value NOTATION (TEXT) #IMPLIED>

   <!ELEMENT comment (#PCDATA)*>
   <!ATTLIST comment
   %attr.language;
   %attr.altrep;
```

```
      value NOTATION (TEXT) #IMPLIED>

      <!ELEMENT description (#PCDATA)*>
      <!ATTLIST description
      %attr.language;
      %attr.altrep;
      value NOTATION (TEXT) #IMPLIED>

      <!ELEMENT geo (lat, lon)>

      <!ELEMENT lat (#PCDATA)>
      <!ATTLIST lat value NOTATION (FLOAT) #IMPLIED>
      <!-- A decimal degree float number to 6 decimal places -->

      <!ELEMENT lon (#PCDATA)>
      <!ATTLIST lon value NOTATION (FLOAT) #IMPLIED>
      <!-- A decimal degree float number to 6 decimal places -->

      <!ELEMENT location (#PCDATA)>
      <!ATTLIST location
      %attr.language;
      %attr.altrep;
      value NOTATION (TEXT) #IMPLIED>

      <!ELEMENT percent (#PCDATA)>
      <!ATTLIST percent
      value NOTATION (INTEGER) #IMPLIED>

      <!ELEMENT priority (#PCDATA)>
      <!ATTLIST priority
      value NOTATION (INTEGER) #IMPLIED>

      <!ELEMENT resources (#PCDATA)>
      <!ATTLIST resources
      %attr.language;
      %attr.altrep;
      value NOTATION (TEXT) #IMPLIED>

      <!ELEMENT status (#PCDATA)>
      <!ATTLIST status
      %attr.language;
      %attr.altrep;
      value NOTATION (TEXT) #IMPLIED>
      <!-- Text value must match the valid values for the particular -->
      <!-- calendar component. -->

      <!ELEMENT summary (#PCDATA)>
      <!ATTLIST summary
```

```
%attr.language;
%attr.altrep;
value NOTATION (TEXT) #IMPLIED >

<!-- Data and time component property element type declarations -->

<!ELEMENT dtstart (#PCDATA)>
<!ATTLIST dtstart
%attr.tzid;
value NOTATION (DATE-TIME | DATE) "DATE-TIME">

<!ELEMENT dtend (#PCDATA)>
<!ATTLIST dtend
%attr.tzid;
value NOTATION (DATE-TIME | DATE) "DATE-TIME">

<!ELEMENT due (#PCDATA)>
<!ATTLIST due
%attr.tzid;
value NOTATION (DATE-TIME | DATE) "DATE-TIME">

<!ELEMENT completed (#PCDATA)>
<!ATTLIST completed
value NOTATION (DATE-TIME) #IMPLIED>

<!ELEMENT duration (#PCDATA)>
<!ATTLIST duration
value NOTATION (DURATION) #IMPLIED>

<!ELEMENT freebusy (#PCDATA)>
<!ATTLIST freebusy
%attr.fbtype;
value NOTATION (PERIOD) #IMPLIED>

<!ELEMENT transp (#PCDATA)>
<!ATTLIST transp
value NOTATION (TEXT) #IMPLIED>
<!-- Text value must be one of the valid enumerations. -->

<!-- Time zone component property element type declarations -->

<!ELEMENT tzid (#PCDATA)>
<!ATTLIST tzid
value NOTATION (TEXT) #IMPLIED>

<!ELEMENT tzname (#PCDATA)>
<!ATTLIST tzname
%attr.language;
```

```
      value NOTATION (TEXT) #IMPLIED>

      <!ELEMENT tzoffsetfrom (#PCDATA)>
      <!ATTLIST tzoffsetfrom
      value NOTATION (UTC-OFFSET) #IMPLIED>

      <!ELEMENT tzoffsetto (#PCDATA)>
      <!ATTLIST tzoffsetto
      value NOTATION (UTC-OFFSET) #IMPLIED>

      <!ELEMENT tzurl EMPTY>
      <!ATTLIST tzurl
      uri ENTITY #REQUIRED>

      <!-- Relationship component property element type declarations -->

      <!ELEMENT attendee (#PCDATA)>
      <!ATTLIST attendee
      %attr.language;
      %attr.cn;
      %attr.role;
      %attr.partstat;
      %attr.rsvp;
      %attr.cutype;
      %attr.member;
      %attr.delegated-to;
      %attr.delegated-from;
      %attr.sent-by;
      %attr.dir;
      value NOTATION (CALADR) #IMPLIED>

      <!ELEMENT contact (#PCDATA)*>
      <!ATTLIST contact
      %attr.language;
      %attr.altrep;
      value NOTATION (TEXT) #IMPLIED>

      <!ELEMENT organizer (#PCDATA)>
      <!ATTLIST organizer
      %attr.language;
      %attr.cn;
      %attr.sent-by;
      %attr.dir;
      value NOTATION (CALADR) #IMPLIED>

      <!ELEMENT recurrence-id (#PCDATA)>
      <!ATTLIST recurrence-id
      %attr.tzid;
      %attr.range;
      value NOTATION (DATE-TIME | DATE) "DATE-TIME">

      <!ELEMENT related-to (#PCDATA)>
      <!ATTLIST related-to
      %attr.reltype;
      value NOTATION (TEXT) #IMPLIED>

      <!ELEMENT url EMPTY>
      <!ATTLIST url
      uri ENTITY #REQUIRED>

      <!ELEMENT uid (#PCDATA)>
      <!ATTLIST uid
      value NOTATION (TEXT) #IMPLIED>

      <!-- Recurrence component property element type declarations -->

      <!ELEMENT exdate (#PCDATA)>
      <!ATTLIST exdate
      %attr.tzid;
      value NOTATION (DATE-TIME | DATE) "DATE-TIME">

      <!ELEMENT exrule (#PCDATA)>
      <!ATTLIST exrule
      value NOTATION (RECUR) #IMPLIED>

      <!ELEMENT rdate (#PCDATA)>
      <!ATTLIST rdate
      %attr.tzid;
      value NOTATION (DATE-TIME | DATE) "DATE-TIME">

      <!ELEMENT rrule (#PCDATA)>
      <!ATTLIST rrule
      value NOTATION (RECUR) #IMPLIED>

      <!-- Alarm component property element type declarations -->

      <!ELEMENT action (#PCDATA)>
      <!ATTLIST action
      value NOTATION (TEXT) #IMPLIED>
      <!-- Text value must be a valid enumeration -->

      <!ELEMENT repeat (#PCDATA)>
      <!ATTLIST repeat
      value NOTATION (INTEGER) #IMPLIED>

      <!ELEMENT trigger (#PCDATA)>
```

```
<!ATTLIST trigger
%attr.related-to;
value NOTATION (DURATION | DATE-TIME) "DURATION">

<!-- Change management component property element type -->
<!-- declarations -->

<!ELEMENT created (#PCDATA)>
<!ATTLIST created
value NOTATION (DATE-TIME) #IMPLIED>

<!ELEMENT dtstamp (#PCDATA)>
<!ATTLIST dtstamp
value NOTATION (DATE-TIME) #IMPLIED>

<!ELEMENT last-modified (#PCDATA)>
<!ATTLIST last-modified
value NOTATION (DATE-TIME) #IMPLIED>

<!ELEMENT sequence (#PCDATA)>
<!ATTLIST sequence
value NOTATION (INTEGER) #IMPLIED>

<!-- Miscellaneous component property element type declarations -->

<!ELEMENT request-status (#PCDATA)>
<!ATTLIST request-status
%attr.language;
value NOTATION (TEXT) #IMPLIED>

<!-- iCalendar object element type declarations -->

<!ELEMENT iCalendar (vcalendar+)>

<!ELEMENT vcalendar (%cal.comp;)*>
<!ATTLIST vcalendar
%attr.language;
xmlns CDATA #FIXED 'http://www.ietf.org/internet-drafts/draft-
hare-xcalendar-01.txt'
calscale CDATA "GREGORIAN"
method CDATA "PUBLISH"
version CDATA #REQUIRED
prodid CDATA #IMPLIED>
<!-- version - Must be "2.0" if document conforms to this spec. -->
<!-- calscale - Calendar scale. Default is GREGORIAN. -->
<!-- method - C&S method. Default is iTIP PUBLISH. -->
<!-- prodid - ISO 9070 FPI for product that generated iCalendar. -->
```

```
      <!-- "vevent" element type declaration -->
      <!ELEMENT vevent ((%vevent.opt1;)*, (%vevent.optm;)*, valarm*)>

      <!-- "vtodo" element type declaration -->
      <!ELEMENT vtodo ((%vtodo.opt1;)*, (%vtodo.optm;)*, valarm*)>

      <!-- "vjournal" element type declaration -->
      <!ELEMENT vjournal ((%vjournal.opt1;)*, (%vjournal.optm;)*)>

      <!-- "vfreebusy" element type declaration -->
      <!ELEMENT vfreebusy ((%vfreebusy.opt1;)*, (%vfreebusy.optm;)*)>

      <!-- "vtimezone" element type declaration -->
      <!ELEMENT vtimezone (%vtimezone.man;, (%vtimezone.opt1;)*,
      (%vtimezone.mann;)*)>

      <!ELEMENT standard (((%standard.man;)*), (%standard.optm;)*)>

      <!ELEMENT daylight (((%daylight.man;)*), (%daylight.optm;)*)>

      <!ELEMENT valarm ((%valarm.audio;) | (%valarm.display;) |
      (%valarm.email;) | (%valarm.procedure;))>
```

11.2 An example XSL transformation

The following is an example of an XSL transformation which can convert an xCalendar document into an iCalendar [RFC 2445] text object. Again, this is an   example and not presented as the only or best way to accomplish the   transformation.

```
   <?xml version="1.0" encoding="UTF-8"?>
   <xsl:stylesheet version="1.0"
           xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
           xmlns:ical="http://www.ietf.org/rfc/rfc2445.txt"
           xmlns:itip="http://www.ietf.org/rfc/rfc2446.txt"
           xmlns="http://www.ietf.org/rfc/rcf2445.txt">
   <!-- =================================================================== -->
   <!-- xcal2ics.xsl XSL transformation of xCalendar documents to iCalendar file -->
   <!-- Tim Hare, April 2004                                                -->
   <!-- Tim Hare, August 2004 -  updated to wrap at 75 chars                -->
   <!-- Tim Hare, November 2004 - removed unused template, cleanup          -->
   <!-- Tim Hare, May 2005 - added function to quote some characters        -->
   <!-- This transformation may be freely used, attribution is appreciated  -->
   <!-- Concepts and inspiration due to rdf2ical from Masahide Kanzaki      -->
   <!-- who should also receive attribution
-->
```

```
   <!-- ==================================================================== -->
   <!-- define output format for iCalendar for use in result-document instr.     -->
   <xsl:output method="text" media-type="text/calendar" />
   <xsl:template match="icalendar|iCalendar|ical:icalendar|ical:iCalendar">
           <xsl:apply-templates select="vcalendar|ical:vcalendar" />
   </xsl:template>
   <xsl:template match="vcalendar|ical:vcalendar">
                   <xsl:call-template name="emit_text">
                   <!-- parameter "line" is entire result of here to end of with-param --
>
                   <xsl:with-param name="line">
                           <xsl:text>BEGIN:VCALENDAR</xsl:text>
                   </xsl:with-param>
                   </xsl:call-template>
                           <xsl:if test="not(@method) and
                                   not(@Method) and
                                     not(@ical:method) and
                                             not(@ical:Method)">
                                   <xsl:call-template name="emit_text">
                                   <xsl:with-param name="line">
                                           <xsl:text>METHOD:PUBLISH</xsl:text>
                                   </xsl:with-param>
                                   </xsl:call-template>
                           </xsl:if>
                           <xsl:for-each select="@*">
                                   <xsl:call-template name="emit_text">
                                   <xsl:with-param name="line">
                                           <xsl:call-template name="stringcap">
                                           <xsl:with-param name="s" select="local-name()"
/>

                                           </xsl:call-template>
                                           <xsl:text>:</xsl:text>
                                           <xsl:call-template name="stringcap">
                                           <xsl:with-param name="s" select="current()" />
                                           </xsl:call-template>
                                   </xsl:with-param>
                                   </xsl:call-template>
                           </xsl:for-each>
                   <xsl:apply-templates select="./*" />
                   <xsl:call-template name="emit_text">
                   <xsl:with-param name="line">
                           <xsl:text>END:VCALENDAR</xsl:text>
                   </xsl:with-param>
                   </xsl:call-template>
   </xsl:template>
   <xsl:template name="eachcomponent"
match="vevent|vtodo|vjournal|valarm|ical:vevent|ical:vtodo|ical:vjournal|ical:valarm">
           <xsl:call-template name="emit_text">
           <xsl:with-param name="line">
                   <xsl:text>BEGIN:</xsl:text>
                   <xsl:call-template name="stringcap">
                   <xsl:with-param name="s" select="local-name()" />
                   </xsl:call-template>
           </xsl:with-param>
           </xsl:call-template>
                   <xsl:for-each select="./*" >
                                   <xsl:call-template name="eachproperty" />
                   </xsl:for-each>
                   <xsl:call-template name="emit_text">
                   <xsl:with-param name="line">
                           <xsl:text>END:</xsl:text>
                           <xsl:call-template name="stringcap">
                           <xsl:with-param name="s" select="local-name()" />
                           </xsl:call-template>
                   </xsl:with-param>
                   </xsl:call-template>
   </xsl:template>
   <xsl:template name="eachproperty">
           <xsl:if test="local-name() != 'categories' and
                             local-name() != 'CATEGORIES' and
                             local-name() != 'geo' and
                             local-name() != 'GEO' and
                             local-name() != 'resources' and
                             local-name() != 'RESOURCES'">
           <!-- this block outputs a property with all of its parameters -->
                   <xsl:call-template name="emit_text">
                   <xsl:with-param name="line">
                           <xsl:call-template name="stringcap">
                           <xsl:with-param name="s" select="local-name()" />
                           </xsl:call-template>
                           <xsl:for-each select="@*">
                                   <xsl:call-template name="eachparam">
                                   <xsl:with-param name="p" select="local-name()" />
                                   </xsl:call-template>
                           </xsl:for-each>
                           <xsl:text>:</xsl:text>
                           <!-- call replace-text to quote newline, semicolon, colon, reverse
solidus, and comma -->
                           <xsl:call-template name="quote-some">
                           <xsl:with-param name="text"><xsl:value-of select="." /></xsl:with-param>
                     </xsl:call-template>
                   </xsl:with-param>
                   </xsl:call-template>
           <!-- end of regular properties ouput block                    -->
           </xsl:if>
           <xsl:if test="local-name() = 'categories' or
                             local-name() = 'CATEGORIES' or
                             local-name() = 'resources' or
                             local-name() = 'RESOURCES'">
           <!-- this block outputs a Categories or Resources property     -->
```

```
                   <xsl:call-template name="emit_text">
                   <xsl:with-param name="line">
                           <xsl:call-template name="stringcap">
                                   <xsl:with-param name="s" select="local-name()" />
                           </xsl:call-template>
                           <xsl:for-each select="@*">
                                   <xsl:call-template name="eachparam">
                                   <xsl:with-param name="p" select="local-name()" />
                                   </xsl:call-template>
                           </xsl:for-each>
                           <xsl:text>:</xsl:text>
                           <xsl:for-each select="./item|./ITEM">
                                   <xsl:call-template name="stringcap">
                                   <xsl:with-param name="s" select="current()" />
                                   </xsl:call-template>
                                   <xsl:if test="position() != last()">
                                           <xsl:text>,</xsl:text>
                                   </xsl:if>
                           </xsl:for-each>
                   </xsl:with-param>
                   </xsl:call-template>
           <!-- end of Categories or Resources property output block       -->
           </xsl:if>
           <xsl:if test="local-name() = 'geo' or
                             local-name() = 'GEO'">
           <!-- this block outputs a Geo property                          -->
                   <xsl:call-template name="emit_text">
                   <xsl:with-param name="line">
                           <xsl:call-template name="stringcap">
                           <xsl:with-param name="s" select="local-name()" />
                           </xsl:call-template>
                           <xsl:text>:</xsl:text>
                           <xsl:value-of select="lat" />
                           <xsl:text>;</xsl:text>
                           <xsl:value-of select="lon" />
                   </xsl:with-param>
                   </xsl:call-template>
           <!-- end of Geo property output block                            -->
           </xsl:if>
   </xsl:template>
   <xsl:template name="eachparam">
           <xsl:param name="p" />
           <xsl:text>;</xsl:text>
           <xsl:call-template name="stringcap">
                   <xsl:with-param name="s" select="string($p)"/>
           </xsl:call-template>
           <xsl:text>=</xsl:text>
           <xsl:value-of select="." />
   </xsl:template>
   <!-- Capitalize and output a string -->
   <xsl:template name="stringcap">
   <xsl:param name="s"/>
   <xsl:value-of select="translate($s,'abcdefghijklmnopqrstuvwxyz','ABCDEFGHIJKLMNOPQRSTUVWXYZ')"/>
   </xsl:template>
   <!-- Emit text, 75 character max per line, recursively -->
   <xsl:template name="emit_text">
           <xsl:param name="limit" select="number(75)"/> <!-- default limit is 75 " -->
           <xsl:param name="line"/>
           <xsl:value-of select="substring(normalize-space($line),1,$limit)" />
           <!-- Output the newline string -->
           <xsl:text>&#13;&#10;</xsl:text>
           <xsl:if test="string-length($line) > $limit">
                   <xsl:text> </xsl:text>
                   <xsl:call-template name="emit_text">
                           <xsl:with-param name="limit" select="($limit - 1)" /> <!-- use 74 allow
for space -->
                           <xsl:with-param name="line" select="substring($line,($limit + 1))" />
                   </xsl:call-template>
           </xsl:if>
   </xsl:template>
   <!-- ACK: ASPN replace-string function used to model this
     http://aspn.activestate.com/ASPN/Cookbook/XSLT/Recipe/65426
     via Dan Connolly, W3C

     all characters to be quoted are in this template rather than doing multiple calls to the
template
     in the hopes that this will improve performance.
     -->
   <xsl:template name="quote-some">
       <!-- rfc2445#sec4.3.11 says to quote DQUOTE, newline, semicolon, colon,
                             reverse solidus, comma -->
       <xsl:param name="text">
       <xsl:choose>
         <xsl:when test="contains($text,'&quot')">
           <xsl:variable name="before" select="substring-before($text,'&quot')
           <xsl:variable name="before" select="substring-after($text,'&quot')/>
           <xsl:value-of select="$before"/>
           <xsl:value-of select="\&quot" />
           <xsl:call-template name="quote-some">
           <xsl:with-param name="text" select="&after" />
           </xsl:call-template>
         </xsl:when>
         <xsl:when test="contains($text,'&#10')">
           <xsl:variable name="before" select="substring-before($text,'&#10')
           <xsl:variable name="before" select="substring-after($text,'&#10')/>
           <xsl:value-of select="$before"/>
           <xsl:value-of select="\n" />
           <xsl:call-template name="quote-some">
           <xsl:with-param name="text" select="&after" />
           </xsl:call-template>
         </xsl:when>
         <xsl:when test="contains($text,';')">
           <xsl:variable name="before" select="substring-before($text,';')/>
           <xsl:variable name="before" select="substring-after($text,';')/>
           <xsl:value-of select="$before"/>
           <xsl:value-of select="\;" />
           <xsl:call-template name="quote-some">
           <xsl:with-param name="text" select="&after" />
           </xsl:call-template>
          </xsl:when>
         <xsl:when test="contains($text,':')">
           <xsl:variable name="before" select="substring-before($text,':')/>
           <xsl:variable name="before" select="substring-after($text,':')/>
           <xsl:value-of select="$before"/>
           <xsl:value-of select="\:" />
           <xsl:call-template name="quote-some">
           <xsl:with-param name="text" select="&after" />
           </xsl:call-template>
          </xsl:when>
         <xsl:when test="contains($text,'\')">
           <xsl:variable name="before" select="substring-before($text,'\')/>
           <xsl:variable name="before" select="substring-after($text,'\')/>
           <xsl:value-of select="$before"/>
           <xsl:value-of select="\\" />
           <xsl:call-template name="quote-some">
           <xsl:with-param name="text" select="&after" />
           </xsl:call-template>
          </xsl:when>
         <xsl:when test="contains($text,',')">
           <xsl:variable name="before" select="substring-before($text,',')/>
           <xsl:variable name="before" select="substring-after($text,',')/>
           <xsl:value-of select="$before"/>
           <xsl:value-of select="\," />
           <xsl:call-template name="quote-some">
           <xsl:with-param name="text" select="&after" />
           </xsl:call-template>
          </xsl:when>
       <xsl:otherwise>
           <xsl:value-of select="$text"/>
       </xsl:otherwise>
       </xsl:choose>
    </xsl:template>
   </xsl:stylesheet>
```

11.3 A well-formed and valid iCalendar XML document

The following is a simple example of a iCalendar XML document. This document is both a well-formed and valid XML document. It also contains the required reference to the XSL transformation. The iCalendar object specifies an appointment.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE iCalendar PUBLIC "-//IETF//DTD XCAL/iCalendar XML//EN"
"http://www.ietf.org/internet-drafts/draft-hare-xcalendar-01.txt">
<?xml-stylesheet href="xcal2ics.xsl" type="text/xsl" output="text" ?>
<iCalendar>
<vcalendar method="PUBLISH"
      version="2.0"
      prodid="-//HandGen//NONSGML vGen v1.0//EN">
<vevent>
     <uid>19981116T150000@cal10.host.com</uid>
     <dtstamp>19981116T145958Z</dtstamp>
     <summary>Project XYZ Review</summary>
     <location>Conference Room 23A</location>
     <dtstart>19981116T163000Z</dtstart>
     <dtend>19981116T190000Z</dtend>
     <categories>
          <item>Appointment</item>
     </categories>
</vevent>
</vcalendar>
</iCalendar>
```

11.4 Including binary content in attachments

The following is an example of a valid iCalendar XML document that also   includes an external reference to an attachment. The iCalendar object specifies   a meeting invitation with an attachment.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE iCalendar PUBLIC "-//IETF//DTD XCAL/iCalendar XML//EN"
"http://www.ietf.org/internet-drafts/draft-hare-xcalendar-01.txt"
<?xml-stylesheet href="xcal2ics.xsl" type="text/xsl" output="text" ?>

<iCalendar>
 <vcalendar>
  <method>REQUEST</method>
  <version>2.0</version>
  <prodid>-//HandGen//NONSGML vGen v1.0//EN</prodid>
  <vevent>
   <uid>19981211T133000@cal1.host.com</uid>
   <dtstamp>19981211T132928Z</dtstamp>
   <organizer>cap://host.com/jim</organizer>
   <dtstart>19981212T150000Z</dtstart>
   <dtend>19981212T160000Z</dtend>
   <summary>Department Meeting</summary>
   <location>Conference Room 23A</location>
   <attendee role="CHAIR">jim@host.com</attendee>
   <attendee role="REQ-PART"
             rsvp="TRUE">MAILTO:joe@host.com</attendee>
   <attendee role="REQ-PART"
             rsvp="TRUE">MAILTO:steve@host.com</attendee>
   <attach>http://host.com/pub/photos/holiday.jpg</attach>
  </vevent>
 </vcalendar>
</iCalendar>
```

11.5 Including binary content inline

The following is an example of a well-formed and valid iCalendar XML document that includes an attachment as inline binary content. The iCalendar   object specifies a meeting invitation with an attachment.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE iCalendar PUBLIC "-//IETF//DTD XCAL/iCalendar XML//EN"
"http://www.ietf.org/internet-drafts/draft-hare-xcalendar-01.txt">
<?xml-stylesheet href="xcal2ics.xsl" type="text/xsl" output="text" ?>
<iCalendar>
<vcalendar>
 <method>REQUEST</method>
 <version>2.0</version>
 <prodid>-//HandGen//NONSGML vGen v1.0//EN</prodid>
 <vevent>
  <uid>19981211T133000@cal1.host.com</uid>
  <dtstamp>19981211T132928Z</dtstamp>
  <organizer>MAILTO:jim@host.com</organizer>
  <dtstart>19981212T150000Z</dtstart>
  <dtend>19981212T160000Z</dtend>
  <summary>Department Meeting</summary>
  <location>Conference Room 23A</location>
  <attendee role="CHAIR">MAILTO:jim@host.com</attendee>
  <attendee role="REQ-PART"
            rsvp="TRUE">MAILTO:joe@host.com</attendee>
  <attendee role="REQ-PART"
            rsvp="TRUE">MAILTO:steve@host.com</attendee>
  <attach fmttype="IMAGE/JPEG">MIICajCCAdOgAwIBAgI
  CBEUwDQEEBQAwdzELMAkGA1UEBhMCVVMxLDAqBgNVBAoTI05ldHNjYXB
  lIEjYXRpb25z...and so on...IENvcnBvc==</attach>
 </vevent>
 </vcalendar>
</iCalendar>
```

11.6 iCalendar XML document with multiple iCalendar objects

The following is an example of a well-formed and valid iCalendar XML document that includes more than one iCalendar object.

```
      <?xml version="1.0" encoding="UTF-8"?>
      <!DOCTYPE iCalendar PUBLIC "-//IETF//DTD XCAL/iCalendar XML//EN"
      "http://www.ietf.org/internet-drafts/draft-hare-xcalendar-01.txt">
      <?xml-stylesheet href="xcal2ics.xsl" type="text/xsl" output="text" ?>
      <iCalendar annotation="IT Conference Prep">
      <vcalendar>
       <method>PUBLISH</method>
       <version>2.0</version>
       <prodid>-//HandGen//NONSGML vGen v1.0//EN</prodid>
       <vtodo>
        <uid>19981009T233000@cal1.host.com</uid>
        <dtstamp>19981009T232928Z</dtstamp>
        <dtstart>19981010T000000Z</dtstart>
        <due>19981010T235959Z</due>
        <summary>Register for conference</summary>
        <priority>2</priority>
       </vtodo>
      </vcalendar>
      <vcalendar>
       <version>2.0</version>
       <prodid>-//HandGen//NONSGML vGen v1.0//EN</prodid>
       <method>PUBLISH</method>
       <vevent>
        <uid>19981009T233010@cal1.host.com</uid>
        <dtstamp>19981009T233000Z</dtstamp>
        <dtstart>19981120T133000Z</dtstart>
        <dtend>19981122T183000Z</dtend>
        <summary>IT Conference</summary>
        <location>Downtowner Hotel</location>
       </vevent>
      </vcalendar>
      </iCalendar>
```

11.7 Using the iCalendar namespace

The following is an example of a snippet of a XML document that includes elements from the iCalendar name-space.

```
      <x xmlns:xcal="http://www.ietf.org/internet-drafts/
      draft-hare-xcalendar-01.txt"
      xmlns:pdi="http://pdi.org/schema">
      <xcal:dtstart>19981123T133000Z</xcal:dtstart>
      <xcal:dtend>19981123T203000Z</xcal:dtend>
      <pdi:idnum>1234567</pdi:idnum>
      <pdi:usage>999.99</pdi:usage>
      </x>
```

11.8 Publish meeting information

The following is a snippet of an iCalendar XML document that publishes information about a meeting.

```
<iCalendar>
 <vcalendar>
  <version>2.0</version>
  <prodid>-//hacksw/handcal//NONSGML 1.0//EN</prodid>
  <method>PUBLISH</method>
  <vevent>
   <uid>19970901T130000Z-123401@host.com</uid>
   <dtstamp>19970901T130000Z</dtstamp>
   <dtstart>19970903T163000Z</dtstart>
   <dtend>19970903T190000Z</dtend>
   <summary>Annual Employee Review</summary>
   <class>PRIVATE</class>
   <categories>Business,Human Resources</categories>
  </vevent>
 </vcalendar>
</iCalendar>
```

11.9 Publish transparent annual event

The following is a snippet of an iCalendar XML document that publishes information about an annually repeating event that is transparent to busy time searches.

```
<iCalendar>
 <vcalendar>
  <version>2.0</version>
  <prodid-//hacksw/handcal//NONSGML 1.0//EN</prodid>
  <method>PUBLISH</publish>
  <vevent>
   <uid>19990101T125957Z-123403@host.com</uid>
   <dtstamp>19990101T130000Z</dtstamp>
   <dtstart value="DATE">19991102</dtstart>
   <summary>Our Blissful Anniversary</summary>
   <class>CONFIDENTIAL</class>
   <transp>TRANSPARENT</transp>
   <categories>Anniversary,Personal,Special Occasion</categories>
   <rrule>FREQ=YEARLY</rrule>
  </vevent>
 </vcalendar>
</iCalendar>
```

11.10 Meeting invitation

The following is a snippet of an iCalendar XML document that specifies an invitation for a meeting. The meeting occurs on the first Monday of each year for five years.

```
<iCalendar>
 <vcalendar>
 <method>REQUEST</method>
 <version>2.0</version>
 <prodid>-//hacksw/handcal//NONSGML 1.0//EN</prodid>
 <vevent>
  <uid>19981220T130000Z-123403@host.com</uid>
  <dtstamp>19981220T130050Z</dtstamp>
  <organizer>MAILTO:corprel@host.com</organizer>
  <dtstart>19990104T140000Z</dtstart>
  <dtend>19990104T220000Z</dtend>
  <summary>Annual Stockholders Meeting</summary>
  <location>One Corporate Drive, Wilmington, DL</location>
  <attendee role="CHAIR">MAILTO:mrbig@host.com</attendee>
  <attendee cutype="GROUP"
            rsvp="TRUE">CAP:host.com/stockholders</attendee>
  <categories>Business,Meeting,Special Occasion</categories>
  <rrule>FREQ=YEARLY;COUNT=5;BYDAY=1MO</rrule>
  </vevent>
 </vcalendar>
</iCalendar>
```

11.11 Assign a to-do

The following is a snippet of an iCalendar XML document for a to-do.

```
      <iCalendar>
       <vcalendar>
        <method>REQUEST</method>
        <version>2.0</version>
        <prodid>-//hacksw/handcal//NONSGML 1.0//EN</prodid>
        <vtodo>
         <uid>19990104T133402@ical1.host.com</uid>
         <dtstamp>19990104T133410Z</dtstamp>
         <dtstart value="DATE">19990104</dtstart>
         <due value="DATE">19990129</due>
         <organizer>MAILTO:dboss@host.com</organizer>
         <summary>Periodic Self Review</summary>
         <description>Complete your self review.
         Contact me if you questions.</description>
         <priority>1</priority>
         <class>CONFIDENTIAL</class>
         <attendee>CAP:dilbert@host.com</attendee>
        </vtodo>
       </vcalendar>
      </iCalendar>
```

11.12 Publish busy time

The following is an iCalendar XML document that publishes busy time information. The default value for the "method" attribute is "PUBLISH" and does   not need to be specified in this example.

```
      <iCalendar>
       <vcalendar>
        <version>2.0</version>
        <prodid>-//hacksw/handcal//NONSGML 1.0//EN</prodid>
        <vfreebusy>
         <uid>19980313T133000@ical1.host.com</uid>
         <dtstamp>19990104T133010Z</dtstamp>
         <organizer>CAP:host.com/jsmith</organizer>
         <dtstart>19980313T141711Z</dtstart>
         <dtend>19980410T141711Z</dtend>
         <url>jsmith.ifb</url>
         <freebusy>19980314T233000Z/19980315T003000Z</freebusy>
         <freebusy>19980316T153000Z/19980316T163000Z</freebusy>
         <freebusy>19980318T030000Z/19980318T040000Z</freebusy>
        </vfreebusy>
       </vcalendar>
      </iCalendar>
```

11.13 Request busy time

The following is a snippet of an iCalendar XML document that requests a calendar user's busy time information.

```
      <iCalendar>
       <vcalendar>
        <method>REQUEST</method>
        <version>2.0</version>
        <prodid>-//hacksw/handcal//NONSGML 1.0//EN</prodid>
        <vfreebusy>
         <uid>19970901T083000@ical1.host.com</uid>
         <dtstamp>19970901T083000Z</dtstamp>
         <organizer>MAILTO:jane_doe@host1.com</organizer>
         <dtstart>19971015T050000Z</dtstart>
         <dtend>19971016T050000Z</dtend>
         <attendee>MAILTO:john_public@host2.com</attendee>
        </vfreebusy>
       </vcalendar>
      </iCalendar>
```

11.14 Issue a CAP command

The following is a snippet of an iCalendar XML document that issues a CAP   command to delete a UID.

```
      <iCalendar xmlns="http://www.ietf.org/internet-drafts/draft-hare-
xcalendar-01.txt"
                     xmlns:CAP="="http://www.ietf.org/internet-drafts/draft-
ietf-calsch-cap-13.txt">
       <vcalendar>
        <version>2.0</version>
        <prodid>-//hacksw/handcal//NONSGML 1.0//EN</prodid>
        <target>relcalid-22</target>
        <CAP:cmd id="random but unique per CUA">DELETE</CAP:cmd>
        <CAP:vquery>
         <CAP:query>SELECT VEVENT FROM VAGENDA WHERE UID =
'abcd12345'</CAP:query>
        </CAP:vquery>
        </vcalendar>
       </iCalendar>
```

11.15 A well-formed and valid XML document which can be transformed into iCalendar

The following is a simple example document which contains some date and time elements . This document is both a well-formed and valid XML document. It contains a DTD and references a stylesheet which transforms the schedule elements into iCalendar. The time used is a "floating" time, without timezone information, to simplify the example.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE tour [
     <!ELEMENT tourdata (performer,tourname,gig+)>
     <!ELEMENT performer (#PCDATA)>
     <!ELEMENT tourname (#PCDATA)>
     <!ELEMENT gig (venue,date,time)>
     <!ELEMENT venue (#PCDATA)>
     <!ELEMENT date (#PCDATA)>
     <!ELEMENT time (#PCDATA)]>
<?xml-stylesheet type="text/xsl" href="tour2ical.xsl" output="text">
     <tour>
     <performer>Frank Dylan</performer>
     <tourname>Rolling Blunder Revue</tourname>
     <gig>
             <venue>East Sandusky, Ohio Civic Auditorium</venue>
             <date>01/03/1997</date>
             <time>21:00:00</time>
     </gig>
     <gig>
             <venue>Sandlot, NM Lost Highway Cafe</venue>
             <date>08/09/1997</date>
             <time>12:00:00</time>
     </gig>
     </tour>
```

The required stylesheet to transform the document into an iCalendar object (tour2ical.xsl) . Note that some of the transformations, notably date and time,   would need more work to provide the robustness usually needed for applications; the simple method here is used as an example only.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
     xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="text" media-type="text/calendar" />
<xsl:template match="tour">
      BEGIN:VCALENDAR
      METHOD:PUBLISH
      <xsl:for-each select="./gig">
              BEGIN:VEVENT
      <!-- since no datestamps in original -->
      <!-- use same date & time            -->
              DTSTAMP:<xsl:call-template name="getdate" />
              DTSTART:<xsl:call-template name="getdate" />
              SUMMARY:<xsl:value-of select="../tourname" />
              DESCRIPTION:<xsl:value-of select="../performer" />
              <xsl:text> </xsl:text>
              <xsl:value-of select="./venue" />
      </xsl:for-each>
</xsl:template>
<xsl:template name="getdate">
      <xsl:value-of select="substring(date,7,4)" />
      <xsl:value-of select="substring(date,1,2)" />
      <xsl:value-of select="substring(date,4,2)" />
      <xsl:text>T</xsl:text>
      <xsl:value-of select="substring(time,1,2)" />
      <xsl:value-of select="substring(time,4,2)" />
      <xsl:value-of select="substring(time,7,2)" />
</xsl:template>
</xsl:stylesheet>
```

12. Acknowledgments

This document is based on previous work by Frank Dawson, Eric R. Plamondon, Doug Royer, and Surendra K. Reddy, whose previous XML-iCalendar work provided the basis for this document. Their previous xCalendar work also acknowledged the contributions of Greg FitzPatrick, Charles Goldfarb, Paul Hoffman, Lisa Dusseault and Thomas Rowe. The rdf2ical.xsl transformation created by Masahide Kanzaki provided inspiration and concepts for the XSL transformation in this document. The primary author of this version of the document (T. Hare), however, assumes responsibility for all content, omissions, and especially errors.

13. Security Considerations

CDATA Sections - - A XML iCalendar document may contain CDATA sections to represent content for specific element types. The CDATA section specifies arbitrary character data that is not meant to be interpreted. It is not scanned by the XML parser for markup. While this memo restricts that any CDATA section MUST NOT contain markup or other such alternate representation for the property value, in general, CDATA section from a non-conformant implementation can contain content such as HTML markup. HTML text can be used to invoke programs. Implementors should be aware that this may leave an implementation open to malicious attack that might occur as a result of executing the markup in the CDATA section.

PROCEDURAL ALARMS - - A XML iCalendar document can be created that contains a "VEVENT" and "VTODO" calendar component with "VALARM" calendar components. The "VALARM" calendar component can be of type PROCEDURE and can 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 might occur as a result of executing the attachment.

ATTACHMENTS - - A XML iCalendar document can include references to Uniform Resource Locators that can 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, since the XML objects in this memo may be exchanged via many possible mechanisms, the security considerations observed by implementations of those mechanisms should be followed for this memo.

Author's Address

Tim Hare
An individual
3048 Bell Grove Dr.
Tallahassee, FL 32308
US

Phone: (850)414-4209
Email: TimHare@comcast.net

Appendix A. Bibliography

[FPI] F. Dawson, "iCalendar Formal Public Identifier", Internet Draft, http://www.internic.net/internet-drafts/draft-calsch-icalfpi-00.txt, September 1998.

[ISO9070] "Information Technology_SGML Support Facilities_Registration Procedures for Public Text Owner Identifiers", ISO/IEC 9070, Second Edition, International Organization for Standardization, April 1991.

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

[RFC 2119] S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, http://www.ietf.org/rfc/rfc2119.txt, March 1997.

[RFC 2445] F. Dawson and D. Stenerson, "Internet Calendaring and Scheduling Core Object Specification (iCalendar)", RFC 2445, http://www.ietf.org/rfc/rfc2445.txt, November 1998.

[RFC 2446] F. Dawson, R. Hopson, S. Mansour, S. Silverberg "iCalendar Transport-   independent Interoperability Protocol", RFC 2446, http://www.ietf.org/rfc/rfc2446.txt, November 1998

[RFC 2447] F. Dawson, S. Mansour, S. Silverberg "iCalendar Message-based Interoperability Protocol", RFC 2447, http://www.ietf.org/rfc/rfc2447.txt, November 1998

[CAP] G. Babics, P. Hill, S. Mansour, D. Royer "Calendar Access Protocol",   Internet-Draft, http://www.ietf.org/internet-drafts/draft-ietf-calsch-cap-13.txt, January 2004

[Previous xCalendar work] F. Dawson, S. Reddy, D. Royer, E. Plamondon "iCalendar DTD Document (xCal)", Internet-Draft, http://www.ietf.org/internet-drafts/draft-ietf-many-xcal-03.txt, July 2002

[XML] "Extensible Markup Language (XML)", Worldwide Web Consortium, http://www.w3.org/TR/REC-xml, October 2000.

[XSL] "Extensible Stylesheet Language (XSL)", Worldwide Web Consortium, http://www.w3.org/TR/xsl, October 2001.

[XSLT] "Extensible Stylesheet Language Transformations (XSLT)", Worldwide Web Consortium, http://www.w3.org/TR/xslt, November 1999.

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.