Network Working Group
Internet Draft
<draft-desruisseaux-ischedule-01>
Updates: 4871 (if approved)
Intended status: Standards Track
Expires: September 9, 2010

C. Daboo
Apple
B. Desruisseaux
Oracle
March 8, 2010

# Internet Calendar Scheduling Protocol (iSchedule)

## draft-desruisseaux-ischedule-01

## Abstract

This document defines the Internet Calendar Scheduling Protocol (iSchedule), which is a binding from the iCalendar Transport-independent Interoperability Protocol (iTIP) to the Hypertext Transfer Protocol (HTTP) to enable interoperability between calendaring and scheduling systems over the Internet.

## Status of This Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as “work in progress”.

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/1id-abstracts.txt>.

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>.

This Internet-Draft will expire in September 9, 2010.

## Copyright Notice



[1] http://trustee.ietf.org/license-info

# Table of Contents

# 1. Introduction

This binding document provides the transport specific information necessary to convey iCalendar Transport-independent Interoperability Protocol (iTIP) [RFC5546] messages over the Hypertext Transfer Protocol (HTTP) [RFC2616].

The Internet Calendar Scheduling Protocol (iSchedule) enables interoperability between different calendaring and scheduling systems. Calendaring and scheduling systems that provide support for iSchedule allow their users to perform scheduling transactions such as schedule, reschedule, respond to scheduling request or cancel scheduled calendar components, as well as search for busy time information with users of other calendaring and scheduling systems on the Internet.

iSchedule leverages the DomainKeys Identified Mail (DKIM) service [RFC4871] to provide end-to-end domain-level authentication based on message content and transparent to end users.

Discussion of this Internet-Draft is taking place on the mailing list <https://www.ietf.org/mailman/listinfo/ischedule>.

## 1.1 Motivations

The iCalendar Message-Based Interoperability Protocol (iMIP) [I-D.ietf-calsify-rfc2447bis], has proven to be insufficient to allow users to seamlessly perform the same scheduling operations with users of other calendaring and scheduling systems on the Internet as with users of their own system. This section clarifies the motivations for a binding from the iCalendar Transport-independent Interoperability Protocol (iTIP) [RFC5546] to a transport that allows synchronous end-to-end connectivity.

A binding to an email-based transport is clearly inadequate to search for busy time information since users need and expect to get an immediate response. As such, some calendaring and scheduling systems allow users to publish their free busy information in a resource accessible to others on the Internet. In the absence of a standardized mechanism to locate the resource that provides the free busy information of a user, one thus needs to know the location of this resource in addition to the calendar user address of the users one wish to schedule with.

With an email-based transport, the transparent processing of incoming scheduling messages on the server is only possible when the calendaring and scheduling system is integrated with the email system. Commonly, the processing of incoming scheduling messages occurs on the client instead and requires user intervention, which yield the following consequences:

1. The processing of incoming scheduling messages and the corresponding updates to the calendar only occurs when the client is active. As such, free busy information may be inaccurate (e.g., user still appears busy when the organizer actually rescheduled or canceled the meeting).
2. Calendaring and scheduling systems generally restrain the number of updates sent to users to reduce the number of messages that will clutter their email inbox. As a result, attendees rarely obtain up to date participation status of other attendees.
3. The client becomes responsible for verification of the authenticity and integrity of the scheduling message.

## 1.2 Related Memos

Implementers will need to be familiar with other documents that, along with this document, form a framework for Internet calendaring and scheduling standards.

This document specifies a binding from iTIP to HTTP.

- iCalendar [RFC5545] specifies a core specification of objects, data types, properties and property parameters;
- iTIP [RFC5546] specifies an interoperability protocol for scheduling between different implementations.

Furthermore, implementers will need to be familiar with the DomainKeys Identified Mail (DKIM) service defined in [RFC4871]. An overview of DKIM can be found in [RFC5585].

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

## 1.3 Notational Conventions

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 [RFC2119].

The Augmented BNF (ABNF) syntax used by this document to describe protocol elements is defined in [RFC5234].

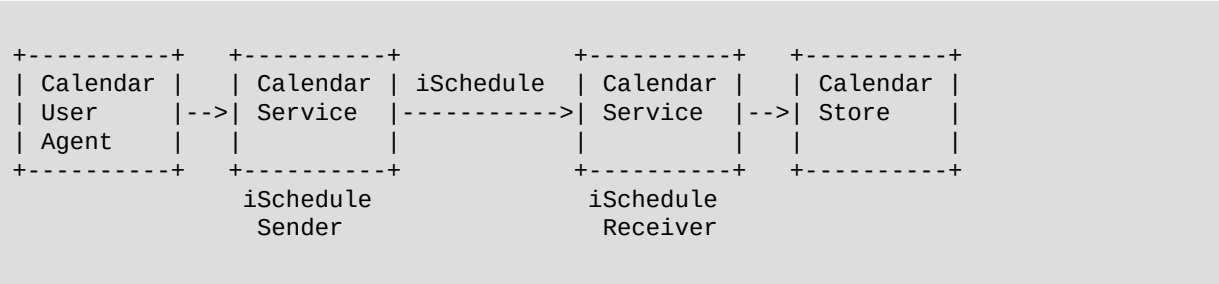
Definitions of XML elements in this document use XML element type declarations (as found in XML Document Type Declarations), described in Section 3.2 of [W3C.REC-xml-20081126].

The namespace "urn:ietf:params:xml:ns:ischedule" is reserved for the XML elements defined in this specification, or in other Standards Track IETF RFCs written to extend iSchedule. It MUST NOT be used for proprietary extensions.

Note that the XML declarations used in this document are incomplete, in that they do not include namespace information. Thus, the reader MUST NOT use these declarations as the only way to validate iSchedule XML element types.

## 2. iSchedule Model

The iSchedule design can be pictured as:




When an iSchedule Sender has a scheduling message to transmit, it determines the iSchedule Receivers to which to delivers the message to and sends the appropriate iSchedule requests. The responsability of an iSchedule Sender is to transfer scheduling messages to one or more iSchedule Receivers, or report its failure to do so.

The means by which a Calendar User Agent instructs a Calendar Service, acting as an iSchedule Sender, to transmit scheduling messages is outside the scope of this document. A Calendar Service could provide support for a standard calendar access protocol, such as CalDAV [RFC4791], [I-D.desruisseaux-caldav-sched] or any other protocol, to allow a Calendar User Agent to perform scheduling operations with users of other Calendar Services.

Likewise, the actual processing of scheduling messages received by a Calendar Service, acting as an iSchedule Receiver, is also outside the scope of this document. Some Calendar Service implementations may decide to process some or all received scheduling messages, while other implementations may decide to leave that work to Calendar User Agent implementations.

# 3. iSchedule Intermediaries

From the end-to-end view, an iSchedule request is sent to an iSchedule Receiver and a response is returned to the iSchedule Sender. In practice, this may not always be the case. An iSchedule request may travel through several iSchedule intermediaries.

iSchedule intermediaries can be used for different purposes, namely:

- Dispatch iSchedule request to the appropriate iSchedule Receivers for each specified Recipient; Users of the same domain could actually be hosted on different iSchedule Receivers.
- Dispatch iSchedule request to the appropriate iSchedule Receivers according to the calendar component type specified in the requests. Different iSchedule Receivers could be responsible of handling, VEVENT, VTODO, VJOURNAL and VFREEBUSY requests.
- Scan iSchedule requests, particularly attachments, for virus.

iSchedule intermediaries are REQUIRED to identify their hostname and the version number of the preceding server from which the request or response arrived. iSchedule intermediaries append this information to the "iSchedule-Via" general header, in sequential order, as the request travels between Sender and Receiver.

For example, an iSchedule request might be submitted to an iSchedule Receiver with the following "iSchedule-Via" header:

```
iSchedule-Via: 1.0 ischedule.example.com:443 (VendorX/2.0),
               1.0 cal.internal.example.com:80 (VendorZ/4.3)
```

# 4. iSchedule Receiver Discovery

This section describes how an iSchedule Sender can discover the host name, the port as well as the Request-URI to use to submit a request to an iSchedule Receiver.

## 4.1 iSchedule SRV Service Types

This specification adds two SRV service labels for use with iSchedule:

| | |
|---|---|
| | Identifies an iSchedule Receiver that uses HTTP without transport layer security ([RFC2818]). |
| ischedules: | Identifies an iSchedule Receiver that uses HTTP with transport layer security ([RFC2818]). |

Example: service record for server without transport layer security

```
_ischedule._tcp.example.com. IN SRV 0 1  80 ischedule.example.com.
```

Example: service record for server with transport layer security

```
_ischedules._tcp.example.com. IN SRV 0 1 443 ischedule.example.com.
```

## 4.2 iSchedule Receiver Request-URI

This specification registers a well-known URI [I-D.nottingham-site-meta] for the iSchedule service, namely, "ischedule" (see Section 11.3.1). iSchedule Receivers MUST support requests targeted at this well-known URI. iSchedule Senders MUST handle HTTP redirects on this well-known URI.

## 4.3 Resolving Calendar User Addresses

To deliver a scheduling message via the iSchedule protocol, an iSchedule Sender needs to determine what iSchedule Receiver it needs to deliver a scheduling message to for a particular Recipient. Each Recipient's calendar user address is specified in the Recipient request header.

A calendar user address as defined by iCalendar is simply a URI. This is typically a mailto URI, but could potentially be any URI type.

To get the SRV record name to query for a given mailto URI, the "domain" portion of the mailto URI MUST be extracted and appended to the service label "_ischedule._tcp." or "_ischedules._tcp.".

Example:

```
  Calendar User Address:  mailto:cyrus@example.com

  Query SRV Record Names: _ischedules._tcp.example.com
                          _ischedule._tcp.example.com
```

In cases where the "domain" portion of the mailto URI contains one or more levels of sub-domain, clients MAY choose to remove successive levels of "sub-domain" if queries for that sub-domain fail to return any SRV records. For example, a mailto URI with the full domain "host.calendar.example.com" would first trigger a querying using the domain "host.calendar.example.com", then if that failed, the domain "calendar.example.com" would be tried, then if that failed the domain "example.com" would be tried.

## 4.4 Using the SRV Record Result

As defined in [RFC2782] the result of an SRV record lookup will be a target host name and a port. An iSchedule Sender uses these to contact the iSchedule Receiver. iSchedule Senders MUST honor the full behavior of SRV records as defined by [RFC2782], in particular the TTL, Priority and Weight options in the record, as well as handling multiple records being returned.

Since an iSchedule server is an HTTP server, an iSchedule client needs to supply a Request-URI in the HTTP request it makes to the server, in addition to the host name and port information. When SRV records are being used there is no way to specify the Request-URI in the SRV record. As a result clients MUST use "/" as the Request-URI for the iSchedule server identified by an SRV record.

# 5. iSchedule Receiver Capabilities

iSchedule Receivers supporting the features described in this document MUST allow iSchedule Sender to query their capabilities by accepting GET requests targeted at the Request-URI "/.well-known/ischedule?query=capabilities". The response body for a successful GET request targeted at this URI MUST be an XML document with query-result as its root element.

Informative rationale: The GET method was favored over the POST method to allow iSchedule Senders to query capabilities with "conditional GET" requests (see Section 9.3 of [RFC2616]).

## 5.1 Example: Querying iSchedule Receiver Capabilities

>> Request <<

```
GET /.well-known/ischedule?query=capabilities HTTP/1.1
Host: cal.example.com
```

>> Response <<

```
HTTP/1.1 200 OK
Date: Mon, 15 Dec 2008 09:32:12 GMT
Content-Type: application/xml; charset=utf-8
Content-Length: xxxx
iSchedule-Version: 1.0
ETag: "afasdf-132afds"

<?xml version="1.0" encoding="utf-8" ?>
<query-result xmlns="urn:ietf:params:xml:ns:ischedule">
  <capability-set>
    <supported-version-set>
      <version>1.0</version>
    </supported-version-set>
    <supported-scheduling-message-set>
      <comp name="VEVENT">
        <method name="REQUEST"/>
        <method name="ADD"/>
        <method name="REPLY"/>
        <method name="CANCEL"/>
      </comp>
      <comp name="VTODO"/>
      <comp name="VFREEBUSY"/>
    </supported-scheduling-message-set>
    <supported-calendar-data-type>
      <calendar-data-type content-type="text/calendar" version="2.0"/>
    </supported-calendar-data-type>
    <supported-attachment-values>
      <inline-attachment/>
      <external-attachment/>
    </supported-attachment-values>
    <supported-recipient-uri-scheme-set>
      <scheme>mailto</scheme>
    </supported-recipient-uri-scheme-set>
    <max-content-length>102400</max-content-length>
    <min-date-time>19910101T000000Z</min-date-time>
    <max-date-time>20381231T000000Z</max-date-time>
    <max-instances>150</max-instances>
    <max-recipients>250</max-recipients>
    <administrator>mailto:ischedule-admin@example.com</administrator>
  </capability-set>
</query-result>
```

# 6. Scheduling

This section defines how an iSchedule Sender can use the HTTP POST method to submit a scheduling message to an iSchedule Receiver.

## 6.1 POST Method

The POST method submits a scheduling message to one or more recipients by targeting the request at the Request-URI of an iSchedule Receiver. The request body of a POST method MUST contain a scheduling message or free-busy message (e.g., an iCalendar object that follows the iTIP semantic).

A submitted scheduling message will be delivered to the calendar user addresses specified in the Recipient request header. A submitted free-busy message will be immediately executed and a free-busy response returned.

Every POST request MUST include the iSchedule-Version request header.

Every POST request MUST include a single Originator request header that specifies the calendar user address of the originator of the scheduling message. The value of the Originator request header MUST match the value of the ORGANIZER property or one of the specified ATTENDEE property depending on the specify METHOD property value as summarized in the following table:

| **Method** | **Originator Requirement** |
|---|---|
| PUBLISH | None |
| REQUEST | MUST match ORGANIZER or ATTENDEE |
| REPLY | MUST match ATTENDEE |
| ADD | MUST match ORGANIZER |
| CANCEL | MUST match ORGANIZER |
| REFRESH | None |
| COUNTER | MUST match ATTENDEE |
| DECLINECOUNTER | MUST match ORGANIZER |

Every POST request MUST include one or more Recipient request headers. The value of this header is a list of one or more calendar user addresses and corresponds to the set of calendar users who will have the scheduling message delivered to them.

### 6.1.1 Schedule Response

A POST request may deliver a scheduling message to one or more calendar users specified in the Recipient request header. Since the behavior of each recipient may vary, it is useful to get response status information for each recipient in the overall POST response. This specification defines a new XML response to convey multiple recipient status.

A response to a POST method that indicates status for one or more recipients MUST be a schedule-response XML element. This MUST contain one or more response elements for each recipient, with each of those containing elements that indicate which recipient they correspond to, the scheduling status of the request for that recipient, any error codes and an optional description.

In the case of a free-busy request, the response elements can also contain calendar-data elements which contain free busy information (e.g., an iCalendar VFREEBUSY component) indicating the busy state of the corresponding recipient, assuming that the free-busy request for that recipient succeeded.

TODO: Define the response body for a failed request.

(supported-calendar-data-type): The resource submitted in the POST request MUST be a supported media type (i.e. text/calendar) for scheduling or free-busy messages;

(valid-calendar-data): The resource submitted in the POST request MUST be valid data for the media type being specified (i.e. valid iCalendar object);

(valid-scheduling-message): The resource submitted in the POST request MUST obey all restrictions specified for the POST request (e.g., the scheduling message follows the restrictions of iTIP);

(originator-specified): The POST request MUST include a valid Originator request header specifying the calendar user address of the originator of the scheduling message.

(recipient-specified): The POST request MUST include one or more valid Recipient request headers specifying the calendar user address of users to whom the scheduling message will be delivered.

(originator-reply): The calendar user identified by the Originator request header in the POST request MUST have previously received the scheduling message that is being replied to when the scheduling message is an incoming scheduling message;

(max-content-length): The request body submitted in the POST request MUST have an octet size less than or equal to the value of the max-content-length capability of the iSchedule Receiver.

### 6.1.2 Status Codes for use with the POST method

The following are examples of response codes one would expect to be used for this method. Note, however, that unless explicitly prohibited any 2/3/4/5xx series response code may be used in a response.

200 (OK) - The command succeeded.

400 (Bad Request) - The Sender has provided an invalid scheduling message.

403 (Forbidden) - The Sender cannot submit a scheduling message to the specified Request-URI.

404 (Not Found) - The URL in the Request-URI was not present.

507 (Insufficient Storage) - The server did not have sufficient space to record the scheduling message.

# 7. iSchedule Domain-Level Authentication

iSchedule uses and extends the mechanism defined by DomainKeys Identified Mail (DKIM) [RFC4871]. DKIM defines a domain-level digital signature authentication framework for email, using public-key cryptography, with the domain name service (DNS) as its key server technology.

The following sections describes how the DomainKeys Identified Mail (DKIM) service can fit into a scheduling service.

## 7.1 Signature Content

The following HTTP headers MUST be included in the signature of a message:

- Content-Type
- Host
- Originator
- Recipient

To allow iSchedule messages to transit via HTTP intermediaries, hop-by-hop headers, such as the following HTTP/1.1 headers MUST NOT be included in the signature of a message:

- Connection
- Keep-Alive
- Proxy-Authenticate
- Proxy-Authorization
- TE
- Trailers
- Transfer-Encoding
- Upgrade

## 7.2 Canonicalization

Transformations can be applied to the message body during transit in order to safely transfer it between the iSchedule Sender and the iSchedule Receiver. Notably, one or more transfer-codings may be applied to an entity.

To avoid breaking the DKIM signature of a message, the signer/verifier MUST compute the hash of a message body against the entity-body, that is, with no transfer-encoding applied to the message body.

### 7.2.1 The "simple-http" Header Canonicalization Algorithm

TODO.

### 7.2.2 The "simple-http" Body Canonicalization Algorithm

TODO.

## 7.3 Key Management

DKIM allows an Administrative Management Domain (ADMD) to constrain the use of a key to specific service types. By default, keys are not constrained to specific service types. As such, the same key could be used to sign email and calendar messages.

This specification defines a new service type "calendar" to constrain the use of a key to calendaring and scheduling services:

| Service Type | Description |
|---|---|
| calendar | calendaring and scheduling service (not necessarily limited to iSchedule) |

## 7.4 Delegation of Signing Authority

An Administrative Management Domain (ADMD) MAY delegate signing authority to other ADMD by publishing a Procuration record. The DNS is proposed as the initial mechanism for Procuration records.

### 7.4.1 Publication of Procuration Record

Signing Procuration records are published using the DNS TXT resource record type.

_procuration._domainkey.aaa.example TXT "v=1.0 d=bbb.example"

TODO: Detailed record syntax specification. Clarify how an ADMD can delegate signing authority to multiple ADMD.

### 7.4.2 Procuration Lookup Procedure

If the Signing Domain IDentifier (SDID) contained in the "d=" tag of DKIM-Signature request header specifies a different domain than the value of the Originator request header, the iSchedule Receiver MUST provide a way to verify whether the owner of the domain of the Originator authorize the domain specified in the SDID to sign on its behalf.

The iSchedule Receiver MAY query DNS for a TXT record corresponding to the Originator's domain prefixed by "_procuration._domainkey." (note the trailing dot).

If the result of this query is a NOERROR response (rcode=0 in [RFC1035]) with an answer that is a single record that is a valid Procuration record, use that record, and the algorithm terminates.

If the result of the query is NXDOMAIN or NOERROR with zero records, there is no Procuration record. If the result of the query contains more than one record, or a record that is not a valid Procuration record, the Procuration result is undefined.

If a query results in a "SERVFAIL" error response (rcode=2 in [RFC1035]), the algorithm terminates without returning a result; possible actions include queuing the message or returning an iSchedule error indicating a temporary failure.

# 8. HTTP Headers

This section defines the syntax and semantics of additional HTTP/1.1 header fields.

The header's syntax uses the optional whitespace (OWS) rule defined as follows:

```
OWS = *( [ CRLF ] WSP )
```

## 8.1 DKIM-Signature Request Header

The DKIM-Signature request header MUST be specified by the iSchedule Sender on all scheduling requests to specify all of the signature and key-fetching data.

```
DKIM-Signature   = "DKIM-Signature" ":" OWS DKIM-Signature-v
DKIM-Signature-v = tag-list
; tag-list is defined in Section 3.2 of <xref target="RFC4871"/>
```

## 8.2 iSchedule-Version General Header

The iSchedule-Version general header field MUST be specified by the iSchedule Sender on requests, and by the iSchedule Receiver on responses.

```
iSchedule-Version      = "iSchedule-Version" ":" OWS
                         iSchedule-Version-v
iSchedule-Version-v    = iSchedule-Version-elem
                         *( OWS "," OWS iSchedule-Version-elem )
iSchedule-Version-elem =  1*DIGIT "." 1*DIGIT
```

## 8.3 iSchedule-Via General Header

The iSchedule-Via general header field MUST be used by iSchedule intermediaries to indicate the intermediate protocols and recipients between the iSchedule Sender and the iSchedule Receiver on requests.

```
iSchedule-Via      = "iSchedule-Via" ":" OWS iSchedule-Via-v
iSchedule-Via-v    = iSchedule-Via-elem
                       *( OWS "," OWS iSchedule-Via-elem )
iSchedule-Via-elem = ( received-protocol received-by [ comment ] )

; received-protocol as defined in [RFC2616]
; received-by as defined in [RFC2616]
; comment as defined in [RFC2616]
```

## 8.4 Originator Request Header

The Originator request header value is a URI which specifies the calendar user address of the originator of the scheduling message. Note that the absoluteURI rule is defined in [RFC3986].

```
Originator   = "Originator" ":" OWS Originator-v
Originator-v = absoluteURI
```

## 8.5 Recipient Request Header

The Recipient request header value is a URI which specifies the calendar user address of the recipients to which the POST method should deliver the submitted scheduling message. Note that the absoluteURI rule is defined in [RFC3986].

```
Recipient      = "Recipient" ":" OWS Recipient-v
Recipient-v    = Recipient-elem *( OWS "," OWS Recipient-elem )
Recipient-elem = absoluteURI
```

# 9. XML Element Definitions

TODO: Re-write XML element definitions using the RELAX NG Compact Syntax [RELAX-NG].

## 9.1 schedule-response XML Element

| | |
|---|---|
| Name: | schedule-response |
| Namespace: | urn:ietf:params:xml:ns:ischedule |
| Purpose: | Contains the set of responses for a POST method request. |
| Description: | See Section 6.1.1. |
| Definition: | |

```
<!ELEMENT schedule-response (response*)>
```

### 9.1.1 response XML Element

| | |
|---|---|
| Name: | response |
| Namespace: | urn:ietf:params:xml:ns:ischedule |
| Purpose: | Contains a single response for a POST method request. |
| Description: | See Section 6.1.1. |
| Definition: | |

```
<!ELEMENT response (recipient,
                    request-status,
                    calendar-data?,
                    error?,
                    responsedescription?)>
```

#### 9.1.1.1 recipient XML Element

| | |
|---|---|
| Name: | recipient |
| Namespace: | urn:ietf:params:xml:ns:ischedule |
| Purpose: | The calendar user address (recipient header value) that the enclosing response for a POST method request is for. |
| Description: | See Section 6.1.1. |
| Definition: | |

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

#### 9.1.1.2 request-status XML Element

| | |
|---|---|
| Name: | request-status |
| Namespace: | urn:ietf:params:xml:ns:ischedule |
| Purpose: | The iTIP REQUEST-STATUS property value for this response. |
| Description: | See Section 6.1.1. |
| Definition: | |

```
<!ELEMENT request-status (#PCDATA)>
```

#### 9.1.1.3 calendar-data XML Element

Name: calendar-data
Namespace: urn:ietf:params:xml:ns:ischedule
Purpose: An iCalendar object in a response to a seach for busy time information.
Description: See Section 6.1.1.
Definition:

```
<!ELEMENT calendar-data (#PCDATA)>
```

#### 9.1.1.4 error XML Element

Name: error
Namespace: urn:ietf:params:xml:ns:ischedule
Purpose: Error responses sometimes need more information to indicate what went wrong.
Description: See Section 6.1.1.
Definition:

```
<!ELEMENT error ANY>
```

#### 9.1.1.5 responsedescription XML Element

Name: responsedescription
Namespace: urn:ietf:params:xml:ns:ischedule
Purpose: Contains information about a status response
Description: See Section 6.1.1.
Definition:

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

### 9.2 query-result XML Element

Name: query-result
Namespace: urn:ietf:params:xml:ns:ischedule
Purpose: Contains result of a query request.
Description: A generic container for the result of a query request, such as a query of the capabilities of an iSchedule Receiver.
Definition:

```
<!ELEMENT query-result (capability-set)>
```

#### 9.2.1 capability-set XML Element

Name: capability-set
Namespace: urn:ietf:params:xml:ns:ischedule
Purpose: Contains iSchedule Receiver capabilities.
Description: The capability-set element contains capabilities of the iSchedule Receiver.

Definition:

```
<!ELEMENT capability-set (supported-version-set,
                          supported-scheduling-message-
set,
                          supported-calendar-data-type,
                          supported-attachment-values,
                          supported-recipient-uri-
scheme-set,
                          max-content-length,
                          min-date-time,
                          max-date-time,
                          max-instances,
                          max-recipients,
                          administrator) >
```

#### 9.2.1.1 supported-version-set XML Element

| | |
|---|---|
| Name: | supported-version-set |
| Namespace: | urn:ietf:params:xml:ns:ischedule |
| Purpose: | Identifies the iSchedule versions supported by the iSchedule Receiver. |
| Description: | An iSchedule Receiver MAY advertise support for multiple versions of the iSchedule protocol. |

Definition:

```
<!ELEMENT supported-version-set (version)+>
```

##### 9.2.1.1.1 version XML Element

| | |
|---|---|
| Name: | version |
| Namespace: | urn:ietf:params:xml:ns:ischedule |
| Purpose: | Specifies an iSchedule version. |

Definition:

```
<!ELEMENT version (#PCDATA)>
<!-- PCDATA value: version number -->
```

#### 9.2.1.2 supported-scheduling-message-set XML Element

| | |
|---|---|
| Name: | supported-scheduling-message-set |
| Namespace: | urn:ietf:params:xml:ns:ischedule |
| Purpose: | Identifies the type of supported scheduling messages. |
| Description: | An iSchedule Receiver could advertise that it only provides support for event and free-busy scheduling messages, and not for to-do scheduling messages, with this capabilities. |

Definition:

```
<!ELEMENT supported-scheduling-message-set (comp)+>
```

#### 9.2.1.2.1 comp XML Element

Name: comp
Namespace: urn:ietf:params:xml:ns:ischedule
Purpose: Identifies a calendar component type.
Description:
Definition:

```
<!ELEMENT comp (method)*>

<!ATTLIST comp name CDATA #REQUIRED>
<!-- name value: a calendar component name -->
```

##### 9.2.1.2.1.1 method XML Element

Name: method
Namespace: urn:ietf:params:xml:ns:ischedule
Purpose: Specifies an iCalendar method type.
Description:
Definition:

```
<!ELEMENT method EMPTY>

<!ATTLIST method name CDATA #REQUIRED>
<!-- name value: a method type -->
```

#### 9.2.1.3 supported-calendar-data-type XML Element

Name: supported-calendar-data-type
Namespace: urn:ietf:params:xml:ns:ischedule
Purpose:
Description:
Definition:

```
<!ELEMENT supported-calendar-data-type (calendar-
data-type)+>
```

#### 9.2.1.3.1 calendar-data-type XML Element

Name: calendar-data-type
Namespace: urn:ietf:params:xml:ns:ischedule
Purpose: Specified a supported media type for scheduling messages.
Description:

Definition:

```
<!ELEMENT calendar-data-type EMPTY>

<!ATTLIST calendar-data-type content-type CDATA
 "text/calendar"
                                    version CDATA "2.0">
<!-- content-type value: a MIME media type -->
<!-- version value: a version string -->
```

#### 9.2.1.4 supported-attachment-values XML Element

Name: supported-attachment-values
Namespace: urn:ietf:params:xml:ns:ischedule
Purpose: Identifies the attachment values supported.
Description:
Definition:

```
<!ELEMENT supported-attachment-values (inline-
attachment?,
                                                external-
attachment?)>
```

##### 9.2.1.4.1 inline-attachment XML Element

Name: inline-attachment
Namespace: urn:ietf:params:xml:ns:ischedule
Purpose: Specifies inline attachment as a supported attachment value.
Description:
Definition:

```
<!ELEMENT inline-attachment EMPTY>
```

##### 9.2.1.4.2 external-attachment XML Element

Name: external-attachment
Namespace: urn:ietf:params:xml:ns:ischedule
Purpose: Specifies external attachment as a supported attachment value.
Description:
Definition:

```
<!ELEMENT external-attachment EMPTY>
```

#### 9.2.1.5 supported-recipient-uri-scheme-set XML Element

Name: supported-recipient-uri-scheme-set
Namespace: urn:ietf:params:xml:ns:ischedule
Purpose: Identifies the supported URI schemes supported in the Recipient HTTP request header.
Description:

Definition:

```
<!ELEMENT supported-recipient-uri-scheme-set (scheme
+)>
```

#### 9.2.1.5.1 scheme XML Element

Name: scheme

Namespace: urn:ietf:params:xml:ns:ischedule

Purpose: Specifies a supported URI scheme.

Description:

Definition:

```
<!ELEMENT scheme (#PCDATA)>
<!-- PCDATA value: URI scheme (e.g., mailto) -->
```

### 9.2.1.6 max-content-length XML Element

Name: max-content-length

Namespace: urn:ietf:params:xml:ns:ischedule

Purpose: Specifies the maximum size allowed for a scheduling message in octets.

Description:

Definition:

```
<!ELEMENT max-content-length (#PCDATA)>
<!-- PCDATA value: a numeric value (positive integer)
-->
```

### 9.2.1.7 min-date-time XML Element

Name: min-date-time

Namespace: urn:ietf:params:xml:ns:ischedule

Purpose: Specifies a DATE-TIME value indicating the earliest date and time in UTC that the server is willing to accept for any DATE or DATE-TIME value in a scheduling message.

Description:

Definition:

```
<!ELEMENT min-date-time (#PCDATA)>
<!-- PCDATA value: an iCalendar format DATE-TIME value
in UTC -->
```

### 9.2.1.8 max-date-time XML Element

Name: max-date-time

Namespace: urn:ietf:params:xml:ns:ischedule

Purpose: Specifies a DATE-TIME value indicating the latest date and time in UTC that the server is willing to accept for any DATE or DATE-TIME value in a scheduling message.

Description:

Definition:

```
<!ELEMENT max-date-time (#PCDATA)>
<!-- PCDATA value: an iCalendar format DATE-TIME value
in UTC -->
```

#### 9.2.1.9 max-instances XML Element

| | |
|---|---|
| Name: | max-instances |
| Namespace: | urn:ietf:params:xml:ns:ischedule |
| Purpose: | Specifies the maximum number of recurrence instances allowed in a scheduling message. |
| Description: | |
| Definition: | |

```
<!ELEMENT max-instances (#PCDATA)>
<!-- PCDATA value: a numeric value (positive integer)
-->
```

#### 9.2.1.10 max-recipients XML Element

| | |
|---|---|
| Name: | max-recipients |
| Namespace: | urn:ietf:params:xml:ns:ischedule |
| Purpose: | Specifies the maximum number of recipients allowed for a scheduling message. |
| Description: | |
| Definition: | |

```
<!ELEMENT max-recipients (#PCDATA)>
<!-- PCDATA value: a numeric value (positive integer)
-->
```

#### 9.2.1.11 administrator XML Element

| | |
|---|---|
| Name: | administrator |
| Namespace: | urn:ietf:params:xml:ns:ischedule |
| Purpose: | Provides contact information for the administrator of the iSchedule Receiver. |
| Description: | |
| Definition: | |

```
<!ELEMENT administrator (#PCDATA)>
<!-- PCDATA value: URI to contact administrator -->
```

# 10. Security Considerations

The process of scheduling involves the sending and receiving of scheduling messages. As a result, the security problems related to messaging in general are relevant here. In particular the authenticity of the scheduling messages needs to be verified.

Potential attacks described in the security considerations of DKIM [RFC4871] are also applicable to iSchedule.

## 10.1 Privacy

iSchedule Senders and iSchedule Receivers MUST use an HTTP connection protected with TLS [RFC5246] as defined in [RFC2818] for all transactions.

## 10.2 Authentication

iSchedule uses and extends the mechanism defined by DomainKeys Identified Mail (DKIM) [RFC4871]. DKIM defines a domain-level digital signature authentication framework for email, using public-key cryptography, with the domain name service as its key server technology.

## 10.3 DNS Considerations

DNS security issues are addressed by DNSSEC [RFC4033].

# 11. IANA Considerations

## 11.1 Namespace Registration

This specification registers a new URN to identify a new XML namespace as per [RFC3688].

### 11.1.1 iSchedule Namespace Registration

Registration request for the iSchedule namespace:

URI: urn:ietf:params:xml:ns:ischedule

Registrant Contact: See the "Authors' Addresses" section of this document.

XML: None. Namespace URIs do not represent an XML specification.

## 11.2 HTTP Headers Registration

This specification registers new headers for use with HTTP as per [RFC3864].

### 11.2.1 DKIM-Signature Request Header Registration

Header field name: DKIM-Signature

Applicable protocol: http

Status: standard

Author/Change controller: IETF

Specification document(s): this specification

Related information: none

### 11.2.2 iSchedule-Version General Header Registration

Header field name: iSchedule-Version

Applicable protocol: http

Status: standard

Author/Change controller: IETF

Specification document(s): this specification

Related information: none

### 11.2.3 iSchedule-Via General Header Registration

Header field name: iSchedule-Via

Applicable protocol: http

Status: standard

Author/Change controller: IETF

Specification document(s): this specification

Related information: none

### 11.2.4 Originator Request Header Registration

Header field name: Originator

Applicable protocol: http

Status: standard

Author/Change controller: IETF

Specification document(s): this specification

Related information: none

### 11.2.5 Recipient Request Header Registration

Header field name: Recipient

Applicable protocol: http

Status: standard

Author/Change controller: IETF

Specification document(s): this specification

Related information: none

## 11.3 Well-Known URI Registration

This specification registers a new well-known URI as per [I-D.nottingham-site-meta].

### 11.3.1 iSchedule Well-Known URI Registration

URI suffix: ischedule

Change controller: IETF.

Specification document(s): this specification

Related information: none

## 11.4 DKIM Parameters Registration

### 11.4.1 DKIM-Signature Canonicalization Algorithm Registration

This specification registers new canonicalization algorithms for the header and body of HTTP requests as per [RFC4871].

The following value should be added to the DKIM-Signature Canonicalization Header registery established in Section 7.3 of [RFC4871]:

| Type | Reference |
|---|---|
| simple-http | RFC XXXX |

The following value should be added to the DKIM-Signature Canonicalization Body registery established in Section 7.3 of [RFC4871]:

| Type | Reference |
|---|---|
| simple-http | RFC XXXX |

### 11.4.2 DKIM Service Type Registration

This specification registers a new DKIM service type to specify that a given selector MUST only be used to verify messages of calendar services (not necessarily limited to iSchedule). The following value should be added to the DKIM Service Type Registry established in Section 7.7 of [RFC4871]:

| Type | Reference |
|---|---|
| calendar | RFC XXXX |

# 12. Acknowledgments

The authors would like to thank the following individuals for contributing their ideas and support for writing this specification: Mattias Amnefelt, Mike Douglass, Tomas Hnetila, Ciny Joy, Barry Leiba, Simon Pilette, Arnaud Quillaud, Simon Vaillancourt, and Wilfredo Sanchez Vega.

The authors would also like to thank the Calendaring and Scheduling Consortium for advice with this specification, and for organizing interoperability testing events to help refine it.

# 13. References

## 13.1 Normative References


[I-D.nottingham-site-meta] Nottingham, M. and E. Hammer-Lahav, "Defining Well-Known URIs", Internet-Draft draft-nottingham-site-meta-05 (work in progress), December 2009.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, February 2000.

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, March 2005.

[RFC4871] Allman, E., Callas, J., Delany, M., Libbey, M., Fenton, J., and M. Thomas, "DomainKeys Identified Mail (DKIM) Signatures", RFC 4871, May 2007.

[RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[RFC5545] Desruisseaux, B., "Internet Calendaring and Scheduling Core Object Specification (iCalendar)", RFC 5545, September 2009.

[RFC5546] Daboo, C., "iCalendar Transport-Independent Interoperability Protocol (iTIP)", RFC 5546, December 2009.

[W3C.REC-xml-20081126] Maler, E., Yergeau, F., Sperberg-McQueen, C., Paoli, J., and T. Bray, "Extensible Markup Language (XML) 1.0 (Fifth Edition)", World Wide Web Consortium Recommendation REC-xml-20081126, November 2008, <http://www.w3.org/TR/2008/REC-xml-20081126>.

## 13.2 Informative References


[I-D.desruisseaux-caldav-sched] Daboo, C. and B. Desruisseaux, "CalDAV Scheduling Extensions to WebDAV", Internet-Draft draft-desruisseaux-caldav-sched-08 (work in progress), August 2009.

[I-D.ietf-calsify-rfc2447bis] Melnikov, A., "iCalendar Message-Based Interoperability Protocol (iMIP)", Internet-Draft draft-ietf-calsify-rfc2447bis-08 (work in progress), January 2010.

[RELAX-NG] Clark, J., "RELAX NG Compact Syntax", December 2001, <http://www.oasis-open.org/committees/relax-ng/compact-20021121.html>.

[RFC3864] Klyne, G., Nottingham, M., and J. Mogul, "Registration Procedures for Message Header Fields", BCP 90, RFC 3864, September 2004.

[RFC4791] Daboo, C., Desruisseaux, B., and L.M. Dusseault, "Calendaring Extensions to WebDAV (CalDAV)", RFC 4791, March 2007.

[RFC5585] Hansen, T., Crocker, D., and P. Hallam-Baker, "DomainKeys Identified Mail (DKIM) Service Overview", RFC 5585, July 2009.

## Authors' Addresses

**Cyrus Daboo**
Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
USA
EMail: cyrus@daboo.name
URI: http://www.apple.com/

**Bernard Desruisseaux**
Oracle Corporation
600 Blvd. de Maisonneuve West
Suite 1900
Montreal, QC H3A 3J2
CANADA
EMail: bernard.desruisseaux@oracle.com
URI: http://www.oracle.com/

# A. Example Scheduling Transactions

This section describes some example scheduling transactions that give a general idea of how scheduling is carried out between an iSchedule Sender and an iSchedule Receiver.

## A.1 Example: Simple Meeting Invitation

In the following example, the iSchedule Sender requests the iSchedule Receiver to deliver a meeting invitation (scheduling REQUEST) to the calendar user mailto:cyrus@example.org. The response indicates that delivery of the scheduling message was successful.

>> Request <<

```
POST /.well-known/ischedule HTTP/1.1
DKIM-Signature: a=rsa-sha256; d=example.com; s=jupiter;
 c=simple-http; q=dns/txt; t=1268069852; x=1283918400;
 h=Originator:Recipient:Host:Content-Type;
 bh=XXXxxxXXXxxxXXXxxxXXXxxxXXXxxxXXXxxx;
 b=XXXxxxXXXxxxXXXxxxXXXxxxXXXxxxXXXxxxXXXxxxXXXxxxXXXxxx
Host: cal.example.org
iSchedule-Version: 1.0
Originator: mailto:bernard@example.com
Recipient: mailto:cyrus@example.org
Content-Type: text/calendar; component=VEVENT; method=REQUEST
Content-Length: xxxx

BEGIN:VCALENDAR
VERSION:2.0
PRODID:-//Example Corp.//EN
METHOD:REQUEST
BEGIN:VEVENT
DTSTAMP:20040901T200200Z
ORGANIZER:mailto:bernard@example.com
DTSTART:20040902T130000Z
DTEND:20040902T140000Z
SUMMARY:Design meeting
UID:34222-232@example.com
ATTENDEE;PARTSTAT=ACCEPTED;ROLE=CHAIR;CUTYPE=IND
 IVIDUAL;CN=Bernard Desruisseaux:mailto:bernard@
 example.com
ATTENDEE;PARTSTAT=NEEDS-ACTION;RSVP=TRUE;ROLE=RE
 Q-PARTICIPANT;CUTYPE=INDIVIDUAL;CN=Cyrus Daboo:
 mailto:cyrus@example.org
END:VEVENT
END:VCALENDAR
```

>> Response <<

```
HTTP/1.1 200 OK
Date: Thu, 02 Sep 2004 16:53:32 GMT
Content-Type: application/xml; charset=utf-8
Content-Length: xxxx
iSchedule-Version: 1.0

<?xml version="1.0" encoding="utf-8" ?>
<schedule-response xmlns="urn:ietf:params:xml:ns:ischedule">
<response>
<recipient>mailto:cyrus@example.org</recipient>
<request-status>2.0;Success</request-status>
<responsedescription>Delivered to recipient</responsedescription>
</response>
</schedule-response>
```

## A.2 Example: Search for Busy Time Information

In the following example, the iSchedule Sender requests the iSchedule Receiver to determine the busy information of the calendar user mailto:cyrus@example.org, over the time range specified by the scheduling message sent in the request. The response includes VFREEBUSY components with the busy time of the requested calendar user.

>> Request <<

```
POST /.well-known/ischedule HTTP/1.1
DKIM-Signature: a=rsa-sha256; d=example.com; s=jupiter;
 c=simple-http; q=dns/txt; t=1268069852; x=1283918400;
 h=Originator:Recipient:Host:Content-Type;
 bh=XXXxxxXXXxxxXXXxxxXXXxxxXXXxxxXXXxxx;
 b=XXXxxxXXXxxxXXXxxxXXXxxxXXXxxxXXXxxxXXXxxxXXXxxxXXXxxx
Host: cal.example.org
iSchedule-Version: 1.0
Originator: mailto:bernard@example.com
Recipient: mailto:cyrus@example.org
Content-Type: text/calendar; component=VFREEBUSY; method=REQUEST
Content-Length: xxxx

BEGIN:VCALENDAR
VERSION:2.0
PRODID:-//Example Corp.//EN
METHOD:REQUEST
BEGIN:VFREEBUSY
DTSTAMP:20040901T200200Z
ORGANIZER:mailto:bernard@example.com
DTSTART:20040902T000000Z
DTEND:20040903T000000Z
UID:34222-232@example.com
ATTENDEE;CN=Cyrus Daboo:mailto:cyrus@example.org
END:VFREEBUSY
END:VCALENDAR
```

>> Response <<

```
HTTP/1.1 200 OK
Date: Thu, 02 Sep 2004 16:53:32 GMT
Content-Type: application/xml; charset=utf-8
Content-Length: xxxx
iSchedule-Version: 1.0

<?xml version="1.0" encoding="utf-8" ?>
<schedule-response xmlns="urn:ietf:params:xml:ns:ischedule">
<response>
<recipient>mailto:cyrus@example.org</recipient>
<request-status>2.0;Success</request-status>
<calendar-data>BEGIN:VCALENDAR
VERSION:2.0
PRODID:-//Example Corp.//EN
METHOD:REPLY
BEGIN:VFREEBUSY
DTSTAMP:20040901T200200Z
ORGANIZER:mailto:bernard@example.com
DTSTART:20040902T000000Z
DTEND:20040903T000000Z
UID:34222-232@example.com
ATTENDEE;CN=Cyrus Daboo:mailto:cyrus@example.org
FREEBUSY;FBTYPE=BUSY-UNAVAILABLE:20040902T000000Z/
 20040902T090000Z,20040902T170000Z/20040903T000000Z
FREEBUSY;FBTYPE=BUSY:20040902T120000Z/20040902T130000Z
END:VFREEBUSY
END:VCALENDAR
</calendar-data>
</response>
</schedule-response>
```

## A.3 Example: Simple Task Assignment

In the following example, the iSchedule Sender requests the iSchedule Sender to deliver a task assignment (scheduling REQUEST) to the calendar user mailto:cyrus@example.org. The response indicates that delivery of the scheduling message was successful.

>> Request <<

```
POST /.well-known/ischedule HTTP/1.1
DKIM-Signature: a=rsa-sha256; d=example.com; s=jupiter;
 c=simple-http; q=dns/txt; t=1268069852; x=1283918400;
 h=Originator:Recipient:Host:Content-Type;
 bh=XXXxxxXXXxxxXXXxxxXXXxxxXXXxxxXXXxxx;
 b=XXXxxxXXXxxxXXXxxxXXXxxxXXXxxxXXXxxxXXXxxxXXXxxxXXXxxx
Host: cal.example.org
iSchedule-Version: 1.0
Originator: mailto:bernard@example.com
Recipient: mailto:cyrus@example.org
Content-Type: text/calendar; component=VTODO; method=REQUEST
Content-Length: xxxx

BEGIN:VCALENDAR
VERSION:2.0
PRODID:-//Example Corp.//CalDAV Client//EN
METHOD:REQUEST
BEGIN:VTODO
DTSTAMP:20040901T200200Z
ORGANIZER:mailto:bernard@example.com
DUE:20070505
SUMMARY:Review Internet-Draft
UID:34222-456@example.com
ATTENDEE;PARTSTAT=NEEDS-ACTION;RSVP=TRUE;ROLE=RE
 Q-PARTICIPANT;CUTYPE=INDIVIDUAL;CN=Cyrus Daboo:
 mailto:cyrus@example.org
END:VEVENT
END:VCALENDAR
```

>> Response <<

```
HTTP/1.1 200 OK
Date: Thu, 02 Sep 2004 16:53:32 GMT
Content-Type: application/xml; charset=utf-8
Content-Length: xxxx
iSchedule-Version: 1.0

<?xml version="1.0" encoding="utf-8" ?>
<schedule-response xmlns="urn:ietf:params:xml:ns:ischedule">
<response>
<recipient>mailto:cyrus@example.org</recipient>
<request-status>2.0;Success</request-status>
<responsedescription>Delivered to recipient</responsedescription>
</response>
</schedule-response>
```

# B. Open Issues

1. As specified in Section 3.6 of DKIM, parameters to the key lookup algorithm are the type of the lookup (the "q=" tag), the domain of the signer (the "d=" tag of the DKIM-Signature header field), and the selector (the "s=" tag). Is the use of the "s=" and "d=" tags sufficient to allow "email" and "calendar" DKIM keys to be managed separately, or should a new tag be introduced to override the default DNS name to look for (e.g., "n=_calendardomainkey", to override "_domainkey")?
2. Should we forbid the "DKIM-Signature" header to be listed in the HTTP "Trailer" header? Else, should we require a "DKIM-Signature-Header" header that would specify the algorithm (a= tag) up front (along with s= and d= tags to match the proper DKIM-Signature if multiple values are provided) when the "DKIM-Signature" header is listed in the "Trailer" header? Knowning the algorithm up front would avoid going through the data twice. "Trailer: Content-MD5" doesn't have this issue since the algorithm is implicit.
3. Should the "Host" and "Recipient" request header really be REQUIRED to be signed? Should we required the actual Request-URI to be signed as well?

# C. Change Log (to be removed by RFC Editor prior to publication)

## C.1 Changes in -01

a. Introduced use of DKIM for calendaring and scheduling services.

b. The XML elements "supported-calendar-data" and "calendar-data" were renamed to "supported-calendar-data-type" and "calendar-data-type" respectively to avoid confusion with the "calendar-data" XML element being used in the "response" XML element.

c. The "recipient" XML element was redefined to accept (#PCDATA) instead of an "href" XML element.

d. The grammar of new HTTP headers is now using the ABNF syntax defined in [RFC5234].

e. Fixed various typos.