Network Working Group
Internet-Draft
Updates: 6376 (if approved)
Intended status: Standards Track
Expires: November 3, 2013

C. Daboo
Apple
B. Desruisseaux
Oracle
May 2, 2013

Internet Calendar Scheduling Protocol (iSchedule)
draft-desruisseaux-ischedule-05

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.

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 [RFC6376] to provide end-to-end domain-level authentication based on message content that is 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) [RFC6047], 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 wishes 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 and requires user intervention, which yields the following consequences:

1. The processing of incoming scheduling messages and the corresponding updates to the calendar only occur when the client is active. As a result, 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 [RFC6376]. 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. Terminology

This specification reuses much of the same terminology as iCalendar [RFC5545], iTIP [RFC5546], HTTP [RFC2616], and DKIM [RFC6376]. Additional terms used by this specification are:

Scheduling message: An iCalendar [RFC5545] object conforming to the requirements of iTIP [RFC5546].

Originator: The calendar user who is sending a scheduling message to one or more other calendar users.

Recipient: A calendar user to whom a scheduling message is being sent.

iSchedule Sender: The iSchedule service responsible for sending scheduling messages.

iSchedule Receiver: The iSchedule service responsible for receiving scheduling messages.

1.4. 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].

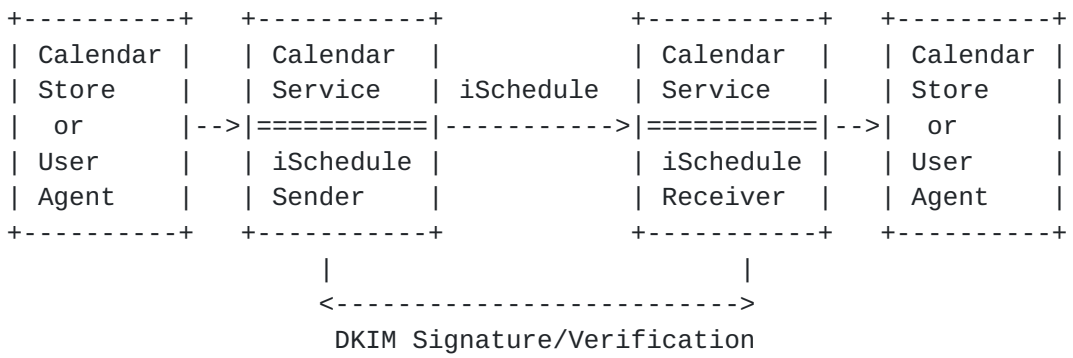
Definitions of XML elements in this document use XML element type declarations (as found in XML Document Type Declarations), described in Section 3.2 of [W3C.REC-xml-20081126].

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. When XML element types in this namespace are referenced in this document outside of the context of an XML fragment, the string "IS:" will be prefixed to the element type names.

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 deliver the message and sends the appropriate iSchedule message. The iSchedule Receiver verifies the authenticity and content of the iSchedule message and delivers it to the Calendar Service.

The means by which a Calendar Store or 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], [RFC6638] 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 Overview

This section provides an overview of the various steps involved for iSchedule Senders and Receivers to transmit scheduling messages between Calendar Services. It references later sections describing the precise details of each step.

3.1. iSchedule Sender Actions

A Calendar Service will generate an iTIP [RFC5546] scheduling message for transmission. It will additionally provide details of the Originator and Recipients. The Calendar Service will "submit" the scheduling message and details to the iSchedule Sender, through a process that is outside the scope of this document.

The iSchedule Sender MUST verify the authenticity of the Originator

and the Originator's authorization to send the scheduling message. In particular the "ORGANIZER" iCalendar property value MUST match the Originator calendar user address. The process by which this authentication and authorization is done is outside the scope of this document.

For each Recipient, the iSchedule Sender will attempt to lookup a matching iSchedule Receiver to which the iSchedule message can be sent, following the rules in Section 4. After determining the iSchedule Receiver to use, the iSchedule Sender MUST check the capabilities of the iSchedule Receiver to ensure it will be able to accept the scheduling message that needs to be sent, as per Section 5.

The iSchedule Sender MUST group together Recipients for whom the iSchedule Receiver is the same, so that a single scheduling message is sent for multiple Recipients, within the limits of the IS:max-recipients (Section 9.2.1.10) value specified in the iSchedule Receiver's capabilities.

For each group of Recipients handled by the same iSchedule Receiver, the iSchedule Sender will construct an HTTP request, as per Section 6, with the body of the HTTP request containing the iSchedule message. Note, in the case of a "VFREEBUSY" iSchedule message, the iSchedule Sender MUST ensure that iCalendar "ATTENDEE" properties in the iSchedule message match one-for-one with the Recipients listed in the HTTP request header.

After constructing the HTTP request, the iSchedule Sender MUST generate a DKIM signature for the request and include a "DKIM-Signature" (Section 8.1) request header, as per Section 7.

The iSchedule Sender then sends the HTTP request to the iSchedule Receiver handling the Recipient group, and receives the HTTP response, which will be an XML document with either an IS:schedule-response or IS:error element as the root element.

The iSchedule Sender aggregates the results for each Recipient group receiving an iSchedule message, and returns the resulting status information for each Recipient to the Calendar Service that generated the schedule message. The process by which this is done is outside the scope of this document.

3.2. iSchedule Receiver Actions

iSchedule Receivers MUST provide a capabilities document to Senders, as per Section 5.

Upon receipt of an iSchedule HTTP request, the iSchedule Receiver verifies the message as per Section 7.4.

Once the authenticity of the message is confirmed, the iSchedule Receiver delivers the scheduling message to the indicated recipients, collects and aggregates the delivery status for each recipient, and returns the result in the HTTP response body.

In the event of a processing error related to the overall request, iSchedule Receivers MUST return an error response as per Section 6.1.2.

4. iSchedule Receiver Discovery

This section describes how an iSchedule Sender can discover the host name, port, and the path to use to submit an HTTP request to an iSchedule Receiver.

For each Recipient to whom a scheduling message is being sent, the iSchedule Sender will "resolve" the associated calendar user address into a domain name, as per Section 4.1.

The iSchedule Sender then uses the extracted domain name to issue a DNS SRV query for the iSchedule service (Section 4.2) expected to be hosted at the domain.

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

Since an iSchedule Receiver is an HTTP server, an iSchedule Sender needs to supply a Request-URI in the HTTP request it makes to the iSchedule Receiver, in addition to the host name and port information. iSchedule Senders MUST use the path specified in any TXT records accompanying the SRV record (as per Section 4.3), or in the absence of a matching TXT record, MUST use the .well-known URI (as per Section 4.4).

4.1. Resolving Calendar User Addresses

To deliver a scheduling message via the iSchedule protocol, an iSchedule Sender needs to determine which iSchedule Receiver to use for a particular recipient. Each recipient's calendar user address is specified in one or more Recipient request headers.

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. However, only URIs containing a "host" element can be used to extract the necessary information to locate an iSchedule Receiver.

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

Example:

Calendar User Address: mailto:cyrus@example.com

Query SRV Record Name: _ischedules._tcp.example.com

In cases where the "domain" portion of the mailto URI contains one or more levels of sub-domain, iSchedule Senders 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 query 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.

4.2. iSchedule SRV Service Type

This specification adds an SRV service label for use with iSchedule:

ischedules: Identifies an iSchedule Receiver that uses HTTP with transport layer security ([RFC2818]).

Example: service record for iSchedule Receiver with transport layer security

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

4.3. iSchedule Service TXT Records

When SRV RRs are used to advertise iSchedule services, it is also convenient to be able to specify a "context path" in the DNS to be retrieved at the same time. To enable that, this specification uses a TXT RR that follows the syntax defined in Section 6 of [RFC6763] and defines a "path" key for use in that record. The value of the key MUST be the actual "context path" to the corresponding service on the iSchedule Receiver.

A site might provide TXT records in addition to SRV records for the service. When present, iSchedule Senders MUST use the "path" value

as the "context path" for the service in HTTP requests. When not present, iSchedule Senders use the ".well-known" URI approach described next.

Example: text record for service with TLS

_ischedules._tcp TXT path=/ischedule

4.4. iSchedule Receiver Request-URI

This specification registers a well-known URI [RFC5785] 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.

5. iSchedule Receiver Capabilities

iSchedule Receivers supporting the features described in this document MUST allow iSchedule Senders to query their capabilities by accepting GET requests targeted at the Request-URI found during discovery (Section 4). The response body for a successful GET request targeted at this URI MUST be an XML document with IS: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]).

iSchedule Receivers SHOULD use normal HTTP expiration mechanisms (as per Section 13.1.3 of [RFC2616]) to ensure caches do not cache the capabilities response for too long. iSchedule Senders SHOULD use normal HTTP conditional GET requests when re-checking capabilities to avoid re-transferring already cached data.

iSchedule Senders SHOULD use the information in the capabilities to determine whether the iSchedule Receiver supports a version of the protocol that the iSchedule Sender can use, and if not, not issue any iSchedule requests with scheduling messages to the iSchedule Receiver. iSchedule Senders SHOULD verify that the scheduling message to be sent to the iSchedule Receiver is in line with the restrictions on scheduling messages indicated by the capabilities before sending the scheduling message.

5.1. Example: Querying iSchedule Receiver Capabilities

```
   >> Request <<

   GET /.well-known/ischedule?action=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
   iSchedule-Capabilities: 123
   ETag: "afasdf-132afds"

   <?xml version="1.0" encoding="utf-8" ?>
   <query-result xmlns="urn:ietf:params:xml:ns:ischedule">
     <capabilities>
       <serial-number>123</serial-number>
       <versions>
         <version>1.0</version>
       </versions>
       <scheduling-messages>
         <component name="VEVENT">
           <method name="REQUEST"/>
           <method name="ADD"/>
           <method name="REPLY"/>
           <method name="CANCEL"/>
         </component>
         <component name="VTODO">
           <method name="REQUEST"/>
           <method name="ADD"/>
           <method name="REPLY"/>
           <method name="CANCEL"/>
         </component>
         <component name="VFREEBUSY">
           <method name="REQUEST"/>
         </component>
       </scheduling-messages>
       <calendar-data-types>
         <calendar-data-type
          content-type="text/calendar" version="2.0"/>
       </calendar-data-types>
       <attachments>
         <inline/>
         <external/>
```

```
      </attachments>
      <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>
    </capabilities>
  </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. Note, this describes the HTTP request prior to generating a DKIM signature as per Section 7.

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 (i.e., an iCalendar object that follows the iTIP semantic).

The submitted scheduling message will be delivered to the Recipients, with status information about per-recipient delivery returned in the HTTP response. However, when the scheduling message is a request for free-busy time, the iSchedule Receiver will immediately execute the free-busy request for the Recipients and return per-recipient iCalendar data in the response for successful free-busy queries.

Every POST request MUST include the "iSchedule-Version" (Section 11.2.2) general header.

Every POST request SHOULD include the "iSchedule-Message-ID" (Section 11.2.4) request header.

Every POST request MUST include the "Cache-Control" HTTP general header containing the cache-directives "no-cache" and "no-transform" to prevent intermediary caches from caching or transforming responses.

Every POST request MUST include a single "Originator" (Section 11.2.5) 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" iCalendar property or one of the specified "ATTENDEE" iCalendar properties in the scheduling message, depending on the specified "METHOD" iCalendar property value as summarized in the following table:

Method	Originator Requirement
PUBLISH	MUST match ORGANIZER
REQUEST	MUST match ORGANIZER (see Note 1)
REPLY	MUST match ATTENDEE
ADD	MUST match ORGANIZER
CANCEL	MUST match ORGANIZER
REFRESH	MUST match ATTENDEE
COUNTER	MUST match ATTENDEE
DECLINECOUNTER	MUST match ORGANIZER

Table 1

Note 1: iTIP does allow an Attendee to forward a "METHOD:REQUEST" scheduling message to another attendee. However, due to complexity of managing the authorization of such requests, this specification does not allow scheduling message forwarding.

Every POST request MUST include one or more "Recipient" (Section 11.2.6) 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. The value of the "Recipient" request header MUST match the value of the "ORGANIZER" iCalendar property or one of the specified "ATTENDEE" iCalendar properties in the scheduling message, depending on the specified "METHOD" iCalendar property value as summarized in the following table:

Method	Recipient Requirement
PUBLISH	None (see Note 1)
REQUEST	MUST match ATTENDEE (see Note 1)
REPLY	MUST match ORGANIZER
ADD	MUST match ATTENDEE (see Note 1)
CANCEL	MUST match ATTENDEE (see Note 1)
REFRESH	MUST match ORGANIZER
COUNTER	MUST match ORGANIZER
DECLINECOUNTER	MUST match ATTENDEE

Table 2

Note 1: iTIP does allow an Organizer to send scheduling message to calendar users who are not listed as Attendees, e.g., to inform other calendar users of an event taking place. However, due to complexity of managing the authorization of such requests, this specification does not allow such scheduling messages.

The Content-Type general header MUST include the type parameters "component" and "method" defined in [RFC5545]. The value of the "component" MUST correspond to the iCalendar component type (e.g., "VEVENT") specified in the scheduling message. The value of the "method" parameter MUST be the same as the value of the "METHOD" iCalendar property in the scheduling message.

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 an XML document with IS:schedule-response as its root 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.

Every POST response MUST include the "Cache-Control" HTTP general header containing the cache-directives "no-cache" and "no-transform" to prevent intermediary caches from caching or transforming responses.

6.1.2. Failed Schedule Response

When there is an overall, as opposed to per-recipient, failure of the POST request, the iSchedule Receiver SHOULD return an XML document with IS:error as its root element. The child elements of the IS:

error element are used to indicate an error code and description, primarily meant for service administrators.

The following XML elements are error codes which can be used within an IS:error element to represent errors:

IS:version-not-supported: The POST request was either missing an "iSchedule-Version" header, or had an "iSchedule-Version" header value for a version not supported by the iSchedule Receiver, as advertised in the IS:versions capability.

IS:invalid-calendar-data-type: The resource submitted in the POST request was not a supported media type (i.e. text/calendar) for scheduling or free-busy messages;

IS:invalid-calendar-data: The resource submitted in the POST request was not valid data for the media type being specified;

IS:invalid-scheduling-message: The resource submitted in the POST request did not obey all restrictions specified for the POST request, violating the IS:scheduling-message capability element, or the requirements of iTIP;

IS:verification-failed: The POST request failed DKIM verification;

IS:originator-missing: The POST request did not include an "Originator" request header specifying the calendar user address of the Originator of the scheduling message.

IS:too-many-originators: The POST request contained more than one "Originator" request header.

IS:originator-invalid: The "Originator" header in the POST request did not include a valid calendar user address for the Originator of the scheduling message.

IS:originator-denied: The calendar user identified by the "Originator" header in the POST request is not allowed to use this service.

IS:recipient-missing: The POST request did not include one or more valid "Recipient" request headers specifying the calendar user address of users to whom the scheduling message will be delivered.

IS:recipient-mismatch: The POST request did not include "Recipient" request header values which exactly match the list of "ATTENDEE" property values in a "VFREEBUSY" request.

IS:max-recipients: The POST request had too many calendar user addresses specified in "Recipient" request headers, violating the IS:max-recipients capability.

IS:attachment-type-not-supported: The scheduling message submitted in the POST request had iCalendar data with "ATTACH" properties whose value type is not supported, violating the IS:attachments capability.

IS:max-content-length: The scheduling message submitted in the POST request had iCalendar data violating the IS:max-content-length capability.

IS:min-date-time: The scheduling message submitted in the POST request had iCalendar data violating the IS:min-date-time capability.

IS:max-date-time: The scheduling message submitted in the POST request had iCalendar data violating the IS:max-date-time capability.

IS:max-instances: The scheduling message submitted in the POST request had iCalendar data violating the IS:max-instances capability.

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. Typically a 403 response code would be used when an XML document with an IS:error element as its root is also returned.

200 (OK) - The command succeeded.

400 (Bad Request) - The Sender has provided an invalid scheduling message, or invalid iSchedule request headers.

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) [RFC6376]. DKIM defines a domain-level digital signature authentication framework for email, using public-key cryptography, with the domain name service (DNS) as one possible key server technology.

This specification extends the applicability of DKIM to the HTTP protocol, with a specific "profile" for use with iSchedule messages. Additionally, DKIM support is REQUIRED for all iSchedule requests, and iSchedule Receivers MUST reject any messages which cannot be verified according to the requirements of DKIM. This is a much stronger requirement than the email use of DKIM, which has to deal with legacy systems.

iSchedule Senders MUST only send iSchedule messages for Originators whose authenticity they have verified. iSchedule Receivers that verify a DKIM signature on an iSchedule request can assume that the iSchedule Sender is not only taking responsibility for sending the message, but has also verified the authenticity of the Originator. As such, iSchedule Receivers can reliably use the Originator information to perform their own authorization based on that value. e.g., the Calendar Service to which an iSchedule Receiver delivers a scheduling message, can apply "filtering" rules to such messages based on the guarantee that the Originator calendar user address has been verified at both the iSchedule Sender and Receiver ends.

This specification uses the syntactic elements of DKIM [RFC6376], but modified for use with HTTP. Where definitions of syntactic elements of DKIM [RFC6376] are applicable to HTTP, they will be used by reference. In cases where the HTTP definition is different, the same ABNF rule name will be used, but the value modified as appropriate.

The following sections describe 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
- iSchedule-Version
- Originator

- Recipient

The iSchedule Receiver MUST verify that the above HTTP headers are included in the signature.

iSchedule Senders and Receivers might use HTTP authentication [RFC2617] in requests, though the process through which credentials are managed are out of scope for this document. If HTTP authentication is used, then the "Authorization" HTTP request header MUST be included in the signature of the message.

The following HTTP headers, if present, SHOULD be included in the signature of a message:

- iSchedule-Message-ID
- User-Agent

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:

- Cache-Control
- Connection
- Host
- Keep-Alive
- Proxy-Authenticate
- Proxy-Authorization
- TE
- Trailer
- Transfer-Encoding
- Upgrade

The "Content-Length" header MUST NOT be signed, since the DKIM signature is generated prior to transfer encoding, and the header value represents the length after any transfer encodings have been applied.

iSchedule Senders MAY include an "x=" DKIM signature tag in the

"DKIM-Signature" header to indicate an expiration time for the signature. When doing so, the "x=" value SHOULD be set to at least 1 minute ahead of the "t=" DKIM signature tag value, to account for processing time between the iSchedule Sender and Receiver.

7.2. Canonicalization

iSchedule Senders and Receivers MUST use the "simple" body canonicalization algorithm defined in Section 3.4.3 of DKIM [RFC6376] to canonicalize the HTTP request message body used for the body hash computation. The iSchedule Sender MUST calculate the body hash prior to any HTTP transfer encodings being applied to the request message body. The iSchedule Receiver MUST calculate the body hash after any HTTP transfer encoding have been removed from the request message body.

iSchedule Senders and Receivers MUST use the new "ischedule-relaxed" header canonicalization algorithm defined below to canonicalize the HTTP request headers used for the signature computation.

7.2.1. The "ischedule-relaxed" Header Canonicalization Algorithm

The "ischedule-relaxed" header canonicalization algorithm is used to canonicalize HTTP header fields where multiple headers fields with the same name might be combined by an HTTP intermediary. The following steps MUST be applied in order:

1. Convert all header field names (not the header field values) to lowercase. For example, convert "SUBJect: AbC" to "subject: AbC".

2. Unfold all header field continuation lines; in particular, lines with terminators embedded in continued header field values (that is, CRLF sequences followed by LWS) MUST be interpreted without the CRLF. Implementations MUST NOT remove the CRLF at the end of the header field value.

3. Combine multiple header fields with the same field name into one one header field value; specifically, append each subsequent field value to the combined field value in order, separated by a comma. For example, combine "recipient:mailto:cyrus@example.com,mailto:mike@example.com" and "recipient:mailto:ken@example.org" into "recipient:mailto:cyrus@example.com,mailto:mike@example.com,mailto:ken@example.org"

4. Convert all sequences of one or more LWS characters to a single SP character. LWS characters here include those before and after a line folding boundary.

5. Delete all WS characters at the end of each unfolded header field value.

6. Delete any LWS characters remaining before and after the colon separating the header field name from the header field value. The colon separator MUST be retained.

7. Delete any LWS characters remaining before and after any commas in the header field value.

Since this canonicalization algorithm "collapses" multiple HTTP header fields into a single header field, the h= tag used in the "DKIM-Signature" request header MUST contain only a single value for each different header field name being signed.

7.3. Key Management

7.3.1. DNS-based Public Key Management

Section 3.6.2 of DKIM [RFC6376] defines a DNS-based key-binding for public key retrieval. iSchedule Senders and Receivers MUST support this method of public key retrieval. To allow public keys to be restricted to just an iSchedule service, this specification defines a new service type "ischedule" to constrain the use of a key to iSchedule:

ABNF:

```
key-s-tag-type /= "ischedule"
```

7.3.2. HTTP-based Public Key Management

This specification defines a new HTTP-based public key management method for use with DKIM [RFC6376]. The "DKIM-Signature" header "q" tag value associated with this method is "http/well-known":

ABNF:

```
sig-q-tag-method /= "http/well-known"
```

Key lookup first involves retrieving an SRV record that will provide the HTTP server host name and port to use for actual key retrieval. Then the key retrieval is done via an HTTP GET request on a .well-known resource. This new key management approach off-loads the key

handling from DNS to an HTTP server, only requiring the DNS administrator to provide the SRV records for authorized HTTP key management servers.

7.3.2.1. SRV Service Type

This specification adds one SRV service label for use with HTTP key management:

domainkey_lookup: Identifies an HTTP server from which public keys for DKIM signatures can be retrieved. The HTTP server MUST support transport layer security ([RFC2818]).

The iSchedule Receiver determines the appropriate SRV name using the "d=" DKIM signature tag value in the "DKIM-signature" header being verified:

```
_domainkey_lookup._tcp.{d}.

; {d} is the d= value from the "DKIM-Signature" header.
```

7.3.2.2. Well-known Request-URI

This specification registers a well-known URI [RFC5785] for the DKIM HTTP-based public key management information, "domainkey" (see Section 11.3.2). To retrieve information about the public key used to sign an iSchedule message, the iSchedule Sender constructs a URI of the form:

```
https://{srv-host}:{srv-port}/.well-known/domainkey/{d}/{s}

; {srv-host} is the host name from the _domainkey_lookup SRV record
; {srv-port} is the port number from the _domainkey_lookup SRV record
; {d} is the d= value from the "DKIM-Signature" header.
; {s} is the s= value from the "DKIM-Signature" header.
```

Documents retrieved from the well-known URI MUST be text documents with a media-type of "text/plain". The format of the text document is:

```
key-doc = 1*(tag-list CRLF)
; tag-list is defined in Section 3.2 of <xref target="RFC6376"/>
```

where tag-list is the unstructured textual form defined in Section 3.6.1. of [RFC6376] - i.e., the same data that would be placed in a DNS TXT record for DNS-based public key management. Note that this allows information for multiple keys to be returned for each {domain-name}, {selector} pair. iSchedule Receivers must use HTTP+TLS

(https:) to retrieve the public key information document, and follow the certificate-verification process specified in [RFC6125].

7.3.2.3. Example Lookup Procedure

Given the following (partial) DKIM-Signature header, the steps below describe how a public key is retrieved from the HTTP pubic key management system.

DKIM-Signature:q=http/well-known;d=cal.example.com;s=isched; ...

1. The iSchedule Receiver first does an SRV record lookup for "_domainkey_lookup._tcp.cal.example.com", and gets back the following example record:

_domainkey_lookup._tcp.cal.example.com. IN SRV 0 1 443 is.example.com.

2. the iSchedule Receiver then makes an HTTP request:

https://is.example.com:443/.well-known/domainkey/cal.example.com/isched

3. It parses the returned document to determine the appropriate public key to use to verify the DKIM signature.

7.3.3. Private Exchange Public Key Management

This specification defines a new public key management method for use with DKIM [RFC6376]. This method is used to indicate that the associated public key has been transferred to the recipient through some (unspecified, secure) private exchange. The "DKIM-Signature" header "q" tag value associated with this method is "private-exchange":

ABNF:

sig-q-tag-method /= "private-exchange"

This method is useful, for example, in situations where an "internal" deployment of iSchedule is being used to connect different calendar systems within an organization. It avoids the need to setup other public key discovery mechanisms when a simple, secure public key exchange can be accomplished between the system administrators.

7.4. Verification Requirements

iSchedule Receivers MUST verify the follow details:

- The HTTP request contains an "iSchedule-Version" header that matches a version of the iSchedule protocol that the iSchedule Receiver supports.

- Only one Originator HTTP request header is present in the request and it matches the appropriate iCalendar property as per Table 1.

- One or more Recipient HTTP request headers are present in the request and they match the appropriate iCalendar properties as per Table 2.

- The "DKIM-Signature" header contains valid information and the signature and body hash values can be verified correctly. Note, for the t= value in the "DKIM-Signature", iSchedule Receivers SHOULD accept values that are no more than 5 minutes in the future, to account for possible clock skew between iSchedule Sender and Receiver.

If the signature cannot be verified, the iSchedule Receiver MUST reject the iSchedule message outright.

If the signature is valid, then iSchedule Receivers have a guarantee that the iSchedule Sender has verified the authenticity of the Originator and determined they are authorized to send the enclosed iTIP message. This allows iSchedule Receivers to use the Originator calendar user address value for access control purposes.

7.5. Authorized Third-Party Signatures

The third-party signature authorization protocol defined in [RFC6541] MAY be used by iSchedule Senders and Receivers.

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="RFC6376"/>
```

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. It SHOULD be included in a response to any "OPTIONS *" HTTP request targeting the iSchedule Receiver, or any "OPTIONS" request on a resource supporting the iSchedule behaviors described in this specification (e.g., the .well-known resource or any resource that .well-known redirects to).

```
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-Capabilities Response Header

The "iSchedule-Capabilities" response header field MUST be specified by the iSchedule Receiver on all responses. iSchedule Senders SHOULD cache this value and use it to detect a change in the iSchedule Receiver capabilities that cause the iSchedule Sender to reload capabilities. The value of this header is maintained by the iSchedule Receiver as described in Section 9.2.1.1.

```
iSchedule-Capabilities = "iSchedule-Capabilities" ":" OWS 1*DIGIT
```

8.4. iSchedule-Message-ID Request Header

The "iSchedule-Message-ID" request header field SHOULD be specified by the iSchedule Sender on requests. This header provides a unique identifier that refers to the specific iSchedule request in which it is included. The uniqueness of this identifier is guaranteed by the iSchedule Sender that generates it. This identifier is intended to be machine readable and not necessarily meaningful to humans.

```
iSchedule-Message-ID   = "iSchedule-Message-ID" ":" OWS token
```

8.5. 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
```

8.6. 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

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?,
                    response-description?)>
```

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 search 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. response-description XML Element

Name: response-description

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

Purpose: Contains information about a status response

Description: See Section 6.1.1.

Definition:

```
<!ELEMENT response-description (#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 (capabilities)>
```

9.2.1. capabilities XML Element

Name: capabilities

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

Purpose: Contains iSchedule Receiver capabilities.

Description: The capabilities element contains capabilities of the iSchedule Receiver.

Definition:

```
<!ELEMENT capabilities (
    serial-number,
    versions,
    scheduling-messages,
    calendar-data-types,
    attachments,
    max-content-length,
    min-date-time,
    max-date-time,
    max-instances,
    max-recipients,
    administrator) >
```

9.2.1.1. serial-number XML Element

Name: serial-number

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

Purpose: Identifies the version of the capabilities information.

Description: This is a numeric value maintained by the iSchedule Receiver. The value is incremented by the iSchedule Receiver each time there has been a substantive change to the capabilities that would require an iSchedule Sender to reload the capabilities to adjust its behavior. The value of this element MUST be returned by the iSchedule Receiver in all HTTP requests via the "iSchedule-Capabilities" response header (Section 8.3). This allows iSchedule Senders to detect changes to the iSchedule Receiver's capabilities during the normal course of making requests, without the need to poll the iSchedule Receiver for such changes.

Definition:

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

9.2.1.2. versions XML Element

Name: versions

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. iSchedule Senders check this value to ensure they can send iSchedule messages with a matching version.

Definition:

```
<!ELEMENT versions (version)+>
```

9.2.1.2.1. version XML Element

Name: version

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

Purpose: Identifies an iSchedule protocol version.

Definition:

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

9.2.1.3. scheduling-messages XML Element

Name: scheduling-messages

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

Purpose: Identifies the type of supported scheduling messages.

Description: An iSchedule Receiver advertises which iCalendar component types it will accept for iTIP messages sent to it. In addition, for each component, it can specify the allowed iTIP "METHOD" property values.

Definition:

```
<!ELEMENT scheduling-messages (component)+>
```

9.2.1.3.1. component XML Element

Name: component

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

Purpose: Identifies a calendar component type.

Description: Used to specify a supported iCalendar component type for scheduling messages. If a IS:method child element is not present, then any iTIP "METHOD" property value can be used in iTIP messages sent to the iSchedule Receiver. If one or more IS:method elements are present, then those indicate the allowed set of iTIP "METHOD" property values.

Definition:

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

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

9.2.1.3.1.1. method XML Element

Name: method

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

Purpose: Identifies an iCalendar method type.

Description: See IS:component.

Definition:

```
<!ELEMENT method EMPTY>

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

9.2.1.4. calendar-data-types XML Element

Name: calendar-data-types

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

Purpose: Identifies what formats of iCalendar data are acceptable.

Definition:

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

9.2.1.4.1. calendar-data-type XML Element

Name: calendar-data-type

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

Purpose: Identifies a supported media type and version for iTIP messages.

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 -->
```

9.2.1.5. attachments XML Element

Name: attachments

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

Purpose: Identifies the attachment values supported.

Description: iSchedule Receivers might restrict what form of attachments are allowed in iTIP messages that are sent to it, for performance, or security reasons. In iCalendar data, attachments can either be specified using "inline" data in the form of a base64 encoded property value, or "external" data in the form of a URI property value. With this capability, an iSchedule Receiver can specify which of "inline" or "external" values it will accept in iTIP messages. See Section 10.4 for additional details.

Definition:

```
<!ELEMENT attachments (inline?, external?)>
```

9.2.1.5.1. inline XML Element

Name: inline

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

Purpose: Identifies "inline" attachments as a supported attachment value.

Definition:

```
<!ELEMENT inline EMPTY>
```

9.2.1.5.2. external XML Element

Name: external

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

Purpose: Identifies "external" attachments as a supported attachment value.

Definition:

```
<!ELEMENT external EMPTY>
```

9.2.1.6. max-content-length XML Element

Name: max-content-length

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

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

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

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

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: The maximum number of recurrence instances allowed in a scheduling message.

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: The maximum number of recipients allowed for a scheduling message.

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.

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 [RFC6376] 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) [RFC6376]. 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].

10.4. Attachment Considerations

iCalendar data can include "inline" attachment data in the form of a base64-encoded "ATTACH" property value. iSchedule Receivers MUST take care when allowing "inline" attachments in scheduling messages as such data might contain malicious content, and SHOULD use some form of content scanner on the attachment data to verify its safety (e.g., a content scanner used for email messages). In addition, "inline" attachment data is likely to be much larger than the actual calendar-related data in a scheduling message, and thus could adversely affect the performance of an iSchedule Receiver processing it. If an iSchedule Receiver allows "inline" attachment data, it MUST apply a limit on the size of acceptable scheduling messages to prevent possible denial-of-service attacks using large "inline" attachment data. In general, it is best for iSchedule Receivers to simply disable the ability for scheduling messages to contain "inline" attachment data, and instead rely solely on "external" attachments in the form of URI attachment values.

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-Capabilities Response Header Registration

Header field name: iSchedule-Capabilities

Applicable protocol: http

Status: standard

Author/Change controller: IETF

Specification document(s): this specification

Related information: none

11.2.4. iSchedule-Message-ID Request Header Registration

Header field name: iSchedule-Message-ID

Applicable protocol: http

Status: standard

Author/Change controller: IETF

Specification document(s): this specification

Related information: none

11.2.5. 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

11.2.6. 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 [RFC5785].

11.3.1. iSchedule Well-Known URI Registration

URI suffix: ischedule

Change controller: IETF.

Specification document(s): this specification

Related information: none

11.3.2. DKIM Well-Known URI Registration

URI suffix: domainkey

Change controller: IETF.

Specification document(s): this specification

Related information: none

11.4. DKIM Parameters Registration

11.4.1. DKIM-Signature Query Method Registry

This specification registers two new query mechanisms that can be used in DKIM-Signature fields. The following values should be added to the DKIM-Signature Query Method Registry established in Section 7.3 of [RFC6376]:

Type	Option	Reference	Status
http	well-known	(this document Section 7.3.2)	active
private-exchange		(this document Section 7.3.3)	active

11.4.2. DKIM Service Type Registration

This specification registers a new DKIM service type to specify that a given public key MUST only be used to verify messages of iSchedule services. The following value should be added to the DKIM Service Type Registry established in Section 7.8 of [RFC6376]:

Type	Reference	Status
ischedule	(this document Section 7.3.1)	active

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, Ken Murchison, Simon Pilette, Arnaud Quillaud, Simon Vaillancourt, and Wilfredo Sanchez Vega.

The authors would also like to thank CalConnect, 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

[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.

[RFC2617] Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A., and L. Stewart, "HTTP Authentication: Basic and Digest Access Authentication", RFC 2617, 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.

[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.

[RFC5785] Nottingham, M. and E. Hammer-Lahav, "Defining Well-Known Uniform Resource Identifiers (URIs)", RFC 5785, April 2010.

[RFC6125] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", RFC 6125, March 2011.

[RFC6376] Crocker, D., Hansen, T., and M. Kucherawy, "DomainKeys Identified Mail (DKIM) Signatures", RFC 6376, September 2011.

[RFC6763] Cheshire, S. and M. Krochmal, "DNS-Based Service Discovery", RFC 6763, February 2013.

[W3C.REC-xml-20081126] Yergeau, F., Paoli, J., Sperberg-McQueen, C., Maler, E., 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

[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. 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.

[RFC6047] Melnikov, A., "iCalendar Message-Based Interoperability Protocol (iMIP)", RFC 6047, December 2010.

[RFC6541] Kucherawy, M., "DomainKeys Identified Mail (DKIM) Authorized Third-Party Signatures", RFC 6541, February 2012.

[RFC6638] Daboo, C. and B. Desruisseaux, "Scheduling Extensions to CalDAV", RFC 6638, June 2012.

Appendix 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=ischedule-relaxed/simple; q=dns/txt:http/well-known;
    t=1268069852; x=1283918400;
    h=Originator:Recipient:Content-Type:
    iSchedule-Version:iSchedule-Message-ID;
    bh=XXXxxxXXXxxxXXXxxxXXXxxxXXXxxxXXXxxx;
    b=XXXxxxXXXxxxXXXxxxXXXxxxXXXxxxXXXxxxXXXxxxXXXxxxXXXxxx
   Host: cal.example.org
   iSchedule-Version: 1.0
   iSchedule-Message-ID: 798F00BB-5B45-4634-B083-0D0CD3A2BB39
   Originator: mailto:bernard@example.com
   Recipient: mailto:cyrus@example.org
   Cache-Control: no-cache, no-transform
   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
Cache-Control: no-cache, no-transform
iSchedule-Version: 1.0
iSchedule-Capabilities: 123

<?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>
    <response-description>Delivered to recipient<
    /response-description>
  </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 users mailto:cyrus@example.org and mailto:mike@example.org, over the time range specified by the scheduling message sent in the request. The response includes VFREEBUSY components with the busy time for one calendar user, and an error for the other calendar user.

```
   >> Request <<

   POST /.well-known/ischedule HTTP/1.1
   DKIM-Signature: a=rsa-sha256; d=example.com; s=jupiter;
    c=ischedule-relaxed/simple; q=dns/txt:http/well-known;
    t=1268069852; x=1283918400;
    h=Originator:Recipient:Content-Type:
    iSchedule-Version:iSchedule-Message-ID;
    bh=XXXxxxXXXxxxXXXxxxXXXxxxXXXxxxXXXxxx;
    b=XXXxxxXXXxxxXXXxxxXXXxxxXXXxxxXXXxxxXXXxxxXXXxxxXXXxxx
   Host: cal.example.org
   iSchedule-Version: 1.0
   iSchedule-Message-ID: A98ADF24-9490-4F01-81C8-FE924F86A9FD
   Originator: mailto:bernard@example.com
   Recipient: mailto:cyrus@example.org
   Recipient: mailto:mike@example.org
   Cache-Control: no-cache, no-transform
   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
   ATTENDEE;CN=Mike Douglass:mailto:mike@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
   Cache-Control: no-cache, no-transform
   iSchedule-Version: 1.0
   iSchedule-Capabilities: 123

   <?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>
     <response>
       <recipient>mailto:mike@example.org</recipient>
       <request-status>5.3;No scheduling support for user<
       /request-status>
       <response-description>Unknown calendar user<
       /response-description>
     </response>
   </schedule-response>
```

A.3. Example: Failed Request

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. For some reason the

verification of the request fails as is indicated by the error response.

>> Request <<

```
POST /.well-known/ischedule HTTP/1.1
DKIM-Signature: a=rsa-sha256; d=example.com; s=jupiter;
 c=ischedule-relaxed/simple; q=dns/txt:http/well-known;
 t=1268069852; x=1283918400;
 h=Originator:Recipient:Content-Type:
 iSchedule-Version:iSchedule-Message-ID;
 bh=XXXxxxXXXxxxXXXxxxXXXxxxXXXxxxXXXxxx;
 b=XXXxxxXXXxxxXXXxxxXXXxxxXXXxxxXXXxxxXXXxxxXXXxxxXXXxxx
Host: cal.example.org
iSchedule-Version: 1.0
Originator: mailto:bernard@example.com
Recipient: mailto:cyrus@example.org
Cache-Control: no-cache, no-transform
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/1.1 403 FORBIDDEN
Date: Thu, 02 Sep 2004 16:53:32 GMT
Content-Type: application/xml; charset=utf-8
Content-Length: xxxx
iSchedule-Version: 1.0
iSchedule-Capabilities: 123

<?xml version="1.0" encoding="utf-8" ?>
<error xmlns="urn:ietf:params:xml:ns:ischedule">
  <verification-failed />
  <response-description>Unable to verify request<
  /response-description>
</error>
```

Appendix B. Change Log (to be removed by RFC Editor prior to publication)

B.1. Changes in -05

a. Fixed missing Recipient header in example.
b. Added statements about what happens when a signature is or is not valid.
c. Removed _ischedule SRV record type as we only support HTTPS.
d. Removed text about adding an extra Recipient header as we no longer need that.

B.2. Changes in -04

a. Added some addition error codes to match MUST requirements.
b. Free busy example now shows a failed calendar user response.
c. Fixed capabilities response example to add method elements for VTODO and VFREEBUSY.
d. More details added to XML element definitions.

B.3. Changes in -03

a. Removed http= tag from DKIM header.
b. Updated lists of must and must not sign headers.
c. Stated that Recipient list must match ATTENDEE list for VFREEBUSY requests.
d. Recommend 5 minute skew for t=.
e. Added serial-number to capabilities and iSchedule-Capabilities response header.

f. Added "ischedule-relaxed" header canonicalization.
g. Fixed examples.

B.4. Changes in -02

a. Major structural changes as well as addition of new sections, including an Overview.
b. Changed capabilities XML schema.
c. XML error elements are now named for the actual error as opposed to WebDAV style pre-conditions.
d. Removed intermediary support and iSchedule-Via header.
e. Added TXT path= lookup to accompany SRV lookup.
f. Added http/well-known public key lookup mechanism.
g. Added iSchedule-Message-ID header.
h. Provided suggested values for t= and x= to cope with clock skew and processing time issues.
i. Indicated that iSchedule-Version header can be returned in OPTIONS responses.
j. Clarified that Attendee list for VFREEBUSY has to be the same as the Recipient list.

B.5. 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.

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/