Internet Calendar Scheduling Protocol (iSchedule)
draft-desruisseaux-ischedule-00

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 on May 19, 2010.

Copyright Notice

Copyright (c) 2009 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents
Table of Contents

1. Introduction .............................................. 4
   1.1. Motivations ........................................... 4
   1.2. Related Memos ........................................ 5
   1.3. Notational Conventions ............................... 5
2. iSchedule Model ........................................... 6
3. iSchedule Intermediaries .................................. 6
4. iSchedule Receiver Discovery .............................. 7
   4.1. iSchedule SRV Service Types .......................... 7
   4.2. iSchedule Receiver Request-URI ...................... 8
   4.3. Resolving Calendar User Addresses .................... 8
   4.4. Using the SRV Record Result ........................... 9
5. iSchedule Receiver Capabilities ........................... 9
   5.1. Example: Querying iSchedule Receiver Capabilities ... 9
6. Scheduling ................................................. 11
   6.1. POST Method .......................................... 11
   6.1.1. Schedule Response ................................ 11
   6.1.2. Status Codes for use with the POST method ....... 12
   6.1.3. Example: Simple meeting invitation ................ 13
   6.1.4. Example: Search for Busy Time Information ......... 15
   6.1.5. Example: Simple Task Assignment ................... 17
7. HTTP Headers ................................................. 18
   7.1. iSchedule-Version General Header .................... 18
   7.2. iSchedule-Via General Header ......................... 18
   7.3. Originator Request Header ............................. 19
   7.4. Recipient Request Header .............................. 19
8. XML Element Definitions .................................... 19
   8.1. schedule-response XML Element ....................... 19
   8.1.1. response XML Element ............................... 20
   8.1.1.1. recipient XML Element ............................ 20
   8.1.1.2. request-status XML Element ...................... 20
   8.1.1.3. calendar-data XML Element ....................... 21
   8.1.1.4. error XML Element ............................... 21
   8.1.1.5. responsedescription XML Element ................ 21
   8.2. query-result XML Element .............................. 22
   8.2.1. capability-set XML Element ......................... 22
   8.2.1.1. supported-version-set XML Element ............... 23
   8.2.1.2. supported-scheduling-message-set XML Element ... 23
   8.2.1.3. supported-calendar-data XML Element ............ 24
1. Introduction

This binding document provides the transport specific information necessary to convey iCalendar Transport-independent Interoperability Protocol (iTIP) [I-D.ietf-calsify-2446bis] 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 allows 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.

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 than with users of their own system. This section clarifies the motivations for a binding from the iCalendar Transport-independent Interoperability Protocol (iTIP) [I-D.ietf-calsify-2446bis] 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 to verify 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 [I-D.ietf-calsify-2446bis] specifies an interoperability protocol for scheduling between different implementations.

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 used by this document to describe protocol elements is described in Section 2.1 of [RFC2616]. Because this augmented BNF uses the basic production rules provided in Section 2.2 of [RFC2616], those rules apply to this document as well.

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:

```
+----------+   +----------+             +----------+   +----------+
| Calendar |   | Calendar |  iSchedule  | Calendar |   | Calendar |
| User     |-->| Service  |------------>| Service  |-->| Store    |
| Agent    |   |          |             |          |   |          |
+----------+   +----------+             +----------+   +----------+
```

When an iSchedule Sender has a scheduling message to transmit, it determines the iSchedule Receivers 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 service types for use with SRV records:

- 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
Example: service record for server with transport layer security

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

4.2. iSchedule Receiver Request-URI

This specification registers a well-known URI for the iSchedule service, namely, "ischedule" (see Section 10.3.1). iSchedule Receivers MUST support requests targeted at this well-known URI. iSchedule Sender MUST handle HTTP redirects on the 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 map a mailto URI to an SRV record name to query, the "domain" portion of the URI is extracted and appended to the service identifier "_ischedule._tcp." or "_ischedules._tcp." to form the record name.

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.

5.1. Example: Querying iSchedule Receiver Capabilities

GET /.well-known/ischedule?query=capabilities HTTP/1.1
Host: cal.example.com
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</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>
      <calendar-data content-type="text/calendar" version="2.0"/>
    </supported-calendar-data>
    <supported-attachment-values>
      <inline-attachment/>
      <external-attachment/>
    </supported-attachment-values>
    <supported-recipient-uri-scheme-set>
      <scheme>mailto</scheme>
    </supported-recipient-uri-scheme-set>
    <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.

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): 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 a single calendar user
address of the currently authenticated user;

(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.

6.1.3. Example: Simple meeting invitation

>> Request <<

POST /.well-known/ischedule HTTP/1.1
Host: cal.example.org
iSchedule-Version: 1.0
Originator: mailto:bernard@example.com
Recipient: mailto:cyrus@example.org
Content-Type: text/calendar
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=INDIVIDUAL;CN=Bernard Desruisseaux:mailto:bernard@example.com
ATTENDEE;PARTSTAT=NEEDS-ACTION;RSVP=TRUE;ROLE=REQ-PARTICIPANT;CUTYPE=INDIVIDUAL;CN=Cyrus Daboo:mailto:cyrus@example.org
END:VEVENT
END:VCALENDAR
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">
  <recipient>mailto:cyrus@example.org</recipient>
  <request-status>2.0;Success</request-status>
  <responsedescription>Delivered to recipient scheduling inbox</responsedescription>
</response>
</schedule-response>

In this 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.
6.1.4. Example: Search for Busy Time Information

>> Request <<

POST /.well-known/ischedule HTTP/1.1
Host: cal.example.org
iSchedule-Version: 1.0
Originator: mailto:bernard@example.com
Recipient: mailto:cyrus@example.org
Content-Type: text/calendar
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
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>

In this 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.
6.1.5. Example: Simple Task Assignment

>> Request <<

POST /.well-known/ischedule HTTP/1.1
Host: cal.example.org
iSchedule-Version: 1.0
Originator: mailto:bernard@example.com
Recipient: mailto:cyrus@example.org
Content-Type: text/calendar
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=REQ-PARTICIPANT;CN=Cyrus Daboo: mailto:cyrus@example.org
END:VEVENT
END:VCALENDAR
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 scheduling inbox</responsedescription>
  </response>
</schedule-response>

In this 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.

7. HTTP Headers

7.1. iSchedule-Version General Header

iSchedule-Version = "iSchedule-Version" ":" 1*DIGIT "." 1*DIGIT

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

TODO: Consider whether the iSchedule-Version should be returned on a per recipient basis in the response body.

7.2. iSchedule-Via General Header

iSchedule-Via = "iSchedule-Via" ":" 1#( received-protocol
                received-by [ comment ] )

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

TODO: Consider whether the iSchedule-Via should be returned on a per recipient basis in the response body.

7.3. Originator Request Header

    Originator = "Originator" ':' absoluteURI

; absoluteURI as defined in [RFC3986]

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 production is defined in RFC3986 [RFC3986].

7.4. Recipient Request Header

    Recipient = "Recipient" ':' 1#absoluteURI

; absoluteURI as defined in [RFC3986]

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 production is defined in RFC3986 [RFC3986].

8. XML Element Definitions

8.1. 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*)>
8.1.1. 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:

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

8.1.1.1. 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:

```xml
<!ELEMENT recipient (href)>
```

8.1.1.2. 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.
8.1.1.3. 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)>

8.1.1.4. 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>

8.1.1.5. responsedescription XML Element

Name: responsedescription

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

Purpose: Contains information about a status response
8.2. 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)>

8.2.1. 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, supported-attachment-values, supported-recipient-uri-scheme-set, max-content-length, min-date-time, max-date-time, max-instances, max-recipients, administrator) >
8.2.1.1. 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:

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

8.2.1.1.1. version XML Element

Name: version

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

Purpose: Specifies an iSchedule version.

Definition:

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

8.2.1.2. 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:

```xml
<!ELEMENT supported-scheduling-message-set (comp)+>
```
8.2.1.2.1.  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 -->

8.2.1.2.1.1.  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 -->

8.2.1.3.  supported-calendar-data XML Element

Name:  supported-calendar-data

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

Purpose:  Contains

Description:
Definition:

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

8.2.1.3.1. calendar-data XML Element

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

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

8.2.1.4. supported-attachment-values XML Element

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

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

8.2.1.4.1. 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>

8.2.1.4.2.  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>

8.2.1.5.  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+)>
8.2.1.5.1. scheme XML Element

Name: scheme

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

Purpose: Specifies a supported URI scheme.

Description:

Definition:

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

8.2.1.6. 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:

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

8.2.1.7. 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:
8.2.1.8. 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 -->

8.2.1.9. 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) -->

8.2.1.10. 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.

Definition:

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

8.2.1.11. 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 -->

9. 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.

9.1. Privacy

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

9.2. Authentication

Mutual authentication is done with TLS [RFC5246] using certificates exchanged between both the iSchedule Sender and the iSchedule Receiver.
9.3. Authorization

Once an iSchedule Sender has been successfully authenticated, the iSchedule Receiver MUST verify that the iSchedule Sender has the authority to send a scheduling message on behalf of the Originator.

TODO: Clarify how verification must be done. Should trusted iSchedule Sender be identified as such in DNS (e.g., by storing their certificate in the DNS (see [RFC4398])) Should the iSchedule Receiver contact the iSchedule Receiver of the domain of the iSchedule Sender to find out whether it should be trusted or not?

10. IANA Considerations

10.1. Namespace Registration

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

10.1.1. 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.

10.2. HTTP Headers Registration

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

10.2.1. 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
10.2.2. 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

10.2.3. 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

10.2.4. 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

10.3. Well-Known URI Registration

This specification registers a new well-known URI as per [I-D.nottingham-site-meta].
10.3.1. iSchedule Well-Known URI Registration

URI suffix: ischedule

Change controller: IETF.

Specification document(s): this specification

Related information: none

11. 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, 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.

12. References

12.1. Normative References


[RFC2616] Fielding, R., Gettys, J., Mogul, J.,


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


12.2. Informative References

Appendix A. Open Issues

1. Should we consider [I-D.faltstrom-uri] as another approach to discover iSchedule Receiver?

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/