CalConnect TC

Internet Calendar Scheduling Protocol (iSchedule)

Working Draft Standard

Warning for drafts

This document is not a CalConnect Standard. It is distributed for review and comment, and is subject to change without notice and may not be referred to as a Standard. Recipients of this draft are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

Recipients of this draft are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

The Calendaring and Scheduling Consortium, Inc. 2017

The Calendaring and Scheduling Consortium, Inc.

4390 Chaffin Lane
McKinleyville
California 95519
United States of America

copyright@calconnect.org
www.calconnect.org

Contents

Abstract iv

Introduction v
Motivations v
Related Memos v

1. Scope 1

2. Normative references 1

3. Terms and definitions 2

4. iSchedule Model 2

5. iSchedule Overview 3
5.1. iSchedule Sender Actions 3
5.2. iSchedule Receiver Actions 3

6. iSchedule Receiver Discovery 4
6.1. iSchedule SRV Service Type 4
6.2. iSchedule Service TXT Records 5
6.3. iSchedule Receiver Request-URI 5

7. iSchedule Receiver Capabilities 5
7.1. Example: Querying iSchedule Receiver Capabilities 5

8. Scheduling 6
8.1. POST Method 7
8.2. Schedule Response 8
8.3. Failed Schedule Response 8

9. HTTP Headers 10
9.1. iSchedule-Version General Header 10
9.2. iSchedule-Capabilities Response Header 10
9.3. iSchedule-Message-ID Request Header 10
9.4. Originator Request Header 10
9.5. Recipient Request Header 11

10. XML Element Definitions 11
10.1. schedule-response XML Element 11
10.2. query-result XML Element 13

11. Security Considerations 18
11.1. Privacy 18
11.2. Authentication 18
11.3. DNS Considerations 18
11.4. Attachment Considerations 18

12. IANA Considerations 18
12.1. Namespace Registration 18
12.2. HTTP Headers Registration 19
12.3. Well-Known URI Registration 20

13. Acknowledgments 20

Appendix A (normative) Example Scheduling Transactions 21
A.1. Example: Simple Meeting Invitation 21
A.2. Example: Search for Busy Time Information 22
A.3. Example: Failed Request 23

Bibliography 25

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.

Introduction

This binding document provides the transport specific information necessary to convey iCalendar Transport-independent Interoperability Protocol (iTIP) IETF RFC 5546 messages over the Hypertext Transfer Protocol (HTTP) IETF RFC 7230.

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.

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

Motivations

The iCalendar Message-Based Interoperability Protocol (iMIP) IETF RFC 6047, 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) IETF RFC 5546 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.

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 specifies a core specification of objects, data types, properties and property parameters;
— iTIP 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.

Internet Calendar Scheduling Protocol (iSchedule)

1. Scope

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.

2. Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

IETF RFC 2119, S. BRADNER. *Key words for use in RFCs to Indicate Requirement Levels*. 1997. RFC Publisher. https://www.rfc-editor.org/info/rfc2119.

IETF RFC 2782, A. GULBRANDSEN, P. VIXIE and L. ESIBOV. *A DNS RR for specifying the location of services (DNS SRV)*. 2000. RFC Publisher. https://www.rfc-editor.org/info/rfc2782.

IETF RFC 2818, E. RESCORLA. *HTTP Over TLS*. 2000. RFC Publisher. https://www.rfc-editor.org/info/rfc2818.

IETF RFC 3688, M. MEALLING. *The IETF XML Registry*. 2004. RFC Publisher. https://www.rfc-editor.org/info/rfc3688.

IETF RFC 3986, T. BERNERS-LEE, R. FIELDING and L. MASINTER. *Uniform Resource Identifier (URI): Generic Syntax*. 2005. RFC Publisher. https://www.rfc-editor.org/info/rfc3986.

IETF RFC 4033, R. ARENDS, R. AUSTEIN, M. LARSON, D. MASSEY and S. ROSE. *DNS Security Introduction and Requirements*. 2005. RFC Publisher. https://www.rfc-editor.org/info/rfc4033.

IETF RFC 5234, P. OVERELL. *Augmented BNF for Syntax Specifications: ABNF*. 2008. RFC Publisher. https://www.rfc-editor.org/info/rfc5234.

IETF RFC 5246, T. DIERKS and E. RESCORLA. *The Transport Layer Security (TLS) Protocol Version 1.2*. 2008. RFC Publisher. https://www.rfc-editor.org/info/rfc5246.

IETF RFC 5545, B. DESRUISSEAUX (ed.). *Internet Calendaring and Scheduling Core Object Specification (iCalendar)*. 2009. RFC Publisher. https://www.rfc-editor.org/info/rfc5545.

IETF RFC 5546, C. DABOO (ed.). *iCalendar Transport-Independent Interoperability Protocol (iTIP)*. 2009. RFC Publisher. https://www.rfc-editor.org/info/rfc5546.

IETF RFC 5785, M. NOTTINGHAM and E. HAMMER-LAHAV. *Defining Well-Known Uniform Resource Identifiers (URIs)*. 2010. RFC Publisher. https://www.rfc-editor.org/info/rfc5785.

IETF RFC 6763, S. CHESHIRE and M. KROCHMAL. *DNS-Based Service Discovery*. 2013. RFC Publisher. https://www.rfc-editor.org/info/rfc6763.

IETF RFC 7230, R. FIELDING and J. RESCHKE (eds.). *Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing*. 2014. RFC Publisher. https://www.rfc-editor.org/info/rfc7230.

IETF RFC 7232, R. FIELDING and J. RESCHKE (eds.). *Hypertext Transfer Protocol (HTTP/1.1): Conditional Requests*. 2014. RFC Publisher. https://www.rfc-editor.org/info/rfc7232.

IETF RFC 7234, R. FIELDING, M. NOTTINGHAM and J. RESCHKE (eds.). *Hypertext Transfer Protocol (HTTP/1.1): Caching*. 2014. RFC Publisher. https://www.rfc-editor.org/info/rfc7234.

IETF RFC 7235, R. FIELDING and J. RESCHKE (eds.). *Hypertext Transfer Protocol (HTTP/1.1): Authentication*. 2014. RFC Publisher. https://www.rfc-editor.org/info/rfc7235.

W3C REC-xml-20081126, EVE MALER, FRANÇOIS YERGEAU, JEAN PAOLI, MICHAEL SPERBERG-MCQUEEN and TIM BRAY (eds.). *Extensible Markup Language (XML) 1.0 (Fifth Edition)*. 2008. World Wide Web Consortium. https://www.w3.org/TR/2008/REC-xml-20081126/.

3. Terms and definitions

For the purposes of this document, the following terms and definitions apply.

This specification reuses much of the same terminology as iCalendar, iTIP, and HTTP. Additional terms used by this specification are:

3.1. Scheduling message
An iCalendar object conforming to the requirements of iTIP.

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

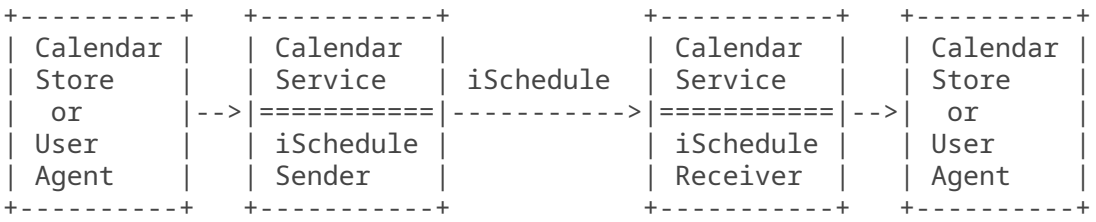
3.3. Recipient
A calendar user to whom a scheduling message is being sent.

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

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

4. 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 IETF RFC 4791, IETF RFC 6638 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.

5. 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.

5.1. iSchedule Sender Actions

A Calendar Service will generate an iTIP IETF RFC 5546 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 Clause 6. 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 Clause 7.

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 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 Clause 8, 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.

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.

5.2. iSchedule Receiver Actions

iSchedule Receivers MUST provide a capabilities document to Senders, as per Clause 7.

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 Clause 8.3.

6. 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 Clause 6.

The iSchedule Sender then uses the extracted domain name to issue a DNS SRV query for the iSchedule service 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 IETF RFC 2782. 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 IETF RFC 2782.

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 Clause 6.2), or in the absence of a matching TXT record, MUST use the .well-known URI (as per Clause 6.3).

=== 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.

6.1. 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 (IETF RFC 2818).

Example service record for iSchedule Receiver with transport layer security


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

6.2. 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 IETF RFC 6763, Section 6 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
```

6.3. iSchedule Receiver Request-URI

This specification registers a well-known URI IETF RFC 5785 for the iSchedule service, namely, "ischedule" (see Clause 12.3.1). iSchedule Receivers MUST support requests targeted at this well-known URI. iSchedule Senders MUST handle HTTP redirects on this well-known URI.

7. 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 (Clause 6). 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.
NOTE Informative rationale: The GET method was favored over the POST method to allow iSchedule Senders to query capabilities with "conditional GET" requests (see IETF RFC 7232).

iSchedule Receivers SHOULD use normal HTTP expiration mechanisms (as per IETF RFC 7234, Section 5.2) 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.

7.1. Example: Querying iSchedule Receiver Capabilities

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

>> Request <<

```
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>
    <rscales>
      <rscale>GREGORIAN</rscale>
      <rscale>CHINESE</rscale>
    </rscales>
    <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>
```

>> Response <<

8. Scheduling

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

8.1. 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" general header.

Every POST request SHOULD include the "iSchedule-Message-ID" 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" 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:

Table 1

Method	Originator Requirement
PUBLISH	MUST match ORGANIZER
REQUEST	MUST match ORGANIZER[a]
REPLY	MUST match ATTENDEE
ADD	MUST match ORGANIZER
CANCEL	MUST match ORGANIZER
REFRESH	MUST match ATTENDEE
COUNTER	MUST match ATTENDEE
DECLINECOUNTER	MUST match ORGANIZER

[a] 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" 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:

Table 2

Method	Recipient Requirement
PUBLISH	None[a]
REQUEST	MUST match ATTENDEE[a]
REPLY	MUST match ORGANIZER
ADD	MUST match ATTENDEE[a]
CANCEL	MUST match ATTENDEE[a]
REFRESH	MUST match ORGANIZER
COUNTER	MUST match ORGANIZER

Method	Recipient Requirement
DECLINECOUNTER	MUST match ATTENDEE

[a] 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 IETF RFC 5545. 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. If iCalendar data is returned in the response, within an IS:calendar-data XML element, then the media type of that data in the response MUST match the media type in the request.

8.2. 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.

8.3. 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: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.

9. 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 )
```

9.1. 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
```

9.2. 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 Clause 10.2.1.1.

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

9.3. 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
```

9.4. Originator Request Header

The "Originator" request header value is a URI which specifies the calendar user address of the originator of the scheduling message. Note that the absoluteURI rule is defined in IETF RFC 3986.

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



```
Originator-v = absoluteURI
```

9.5. Recipient Request Header

The “Recipient” request header value is a URI which specifies the calendar user address of the recipients to which the POST method should deliver the submitted scheduling message. Note that the absoluteURI rule is defined in IETF RFC 3986.

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

10. XML Element Definitions

10.1. 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 Clause 8.2.
Definition	`<!ELEMENT schedule-response (response*)>`

10.1.1. response XML Element

Name	response
Namespace	urn:ietf:params:xml:ns:ischedule
Purpose	Contains a single response for a POST method request.
Description	See Clause 8.2.

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

10.1.1.1. 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 Clause 8.2.

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

10.1.1.2. 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 Clause 8.2.

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

10.1.1.3. 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 Clause 8.2.

Definition
```
<!ELEMENT calendar-data (#PCDATA)>
<!ATTLIST calendar-data content-type CDATA "text/calendar"
                        version CDATA "2.0">
<!-- content-type value: a MIME media type -->
<!-- version value: a version string -->
```

10.1.1.4. 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 Clause 8.2.

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

10.1.1.5. response-description XML Element

Name response-description

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

Purpose Contains information about a status response

Description See Clause 8.2.

Definition
```
<!ELEMENT response-description (#PCDATA)>
```

10.2. 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)>
```

10.2.1. 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,
    rscales,
    max-content-length,
    min-date-time,
    max-date-time,
    max-instances,
    max-recipients,
    administrator) >
```

10.2.1.1. 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. 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) -->
```

10.2.1.2. 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)+>
```

10.2.1.2.1. 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 -->
```

10.2.1.3. 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)+>
```

10.2.1.3.1. 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 -->
```

10.2.1.3.1.1. 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 -->`

10.2.1.4. 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)+>`

10.2.1.4.1. 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 -->`

10.2.1.5. 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 Clause 11.4 for additional details.

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

10.2.1.5.1. 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>`

10.2.1.5.2. 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>`

10.2.1.6. rscales XML Element

Name rscales

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

Purpose Identifies the “RSCALE” values supported.

Description iSchedule Receivers might support the iCalendar “RSCALE” element on the “RRULE” property. The iSchedule Receiver can advertise what “RSCALE” values are supported via the IS:rscales element.

Definition `<!ELEMENT rscales (rscale*)>`

10.2.1.6.1. rscale XML Element

Name rscale

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

Purpose Indicates a supported “RSCALE” value.

Definition `<!ELEMENT rscale (#PCDATA)>`

10.2.1.7. 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) -->
```

10.2.1.8. 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
-->
```

10.2.1.9. 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
-->
```

10.2.1.10. 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) -->
```

10.2.1.11. 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) -->
```

10.2.1.12. 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 -->`

11. 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.

11.1. Privacy

iSchedule Senders and iSchedule Receivers MUST use an HTTP connection protected with TLS IETF RFC 5246 as defined in IETF RFC 2818 for all transactions.

11.2. Authentication

11.3. DNS Considerations

DNS security issues are addressed by DNSSEC IETF RFC 4033.

11.4. 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.

12. IANA Considerations

12.1. Namespace Registration

This specification registers a new URN to identify a new XML namespace as per IETF RFC 3688.

12.1.1. 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.

12.2. HTTP Headers Registration

This specification registers new headers for use with HTTP as per IETF RFC 3864.

12.2.1. 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

12.2.2. 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

12.2.3. 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

12.2.4. Originator Request Header Registration

Header field name	Originator
Applicable protocol	http
Status	standard
Author/Change controller	IETF
Specification document(s)	this specification
Related information	none

12.2.5. Recipient Request Header Registration

Header field name	Recipient
Applicable protocol	http
Status	standard
Author/Change controller	IETF
Specification document(s)	this specification
Related information	none

12.3. Well-Known URI Registration

This specification registers a new well-known URI as per IETF RFC 5785.

12.3.1. iSchedule Well-Known URI Registration

URI suffix	ischedule
Change controller	IETF.
Specification document(s)	this specification
Related information	none

13. 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.

Appendix A
(normative)
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
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
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
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>
```


Bibliography

[1] IETF RFC 3864, G. KLYNE, M. NOTTINGHAM and J. MOGUL. *Registration Procedures for Message Header Fields*. 2004. RFC Publisher. https://www.rfc-editor.org/info/rfc3864.

[2] IETF RFC 4791, C. DABOO, B. DESRUISSEAUX and L. DUSSEAULT. *Calendaring Extensions to WebDAV (CalDAV)*. 2007. RFC Publisher. https://www.rfc-editor.org/info/rfc4791.

[3] IETF RFC 6047, A. MELNIKOV (ed.). *iCalendar Message-Based Interoperability Protocol (iMIP)*. 2010. RFC Publisher. https://www.rfc-editor.org/info/rfc6047.

[4] IETF RFC 6638, C. DABOO and B. DESRUISSEAUX. *Scheduling Extensions to CalDAV*. 2012. RFC Publisher. https://www.rfc-editor.org/info/rfc6638.

[5] IETF RFC 7529, C. DABOO and G. YAKUSHEV. *Non-Gregorian Recurrence Rules in the Internet Calendaring and Scheduling Core Object Specification (iCalendar)*. 2015. RFC Publisher. https://www.rfc-editor.org/info/rfc7529.

