CALCONNECT.ORG
The CALENDARING & SCHEDULING CONSORTIUM

Title: **CalDAV SCHEDULING REQUIREMENTS**

Version: **1.1**

Date: **11 July 2007**

Contributed: **TC CALDAV**

Editors: **Tony Becker, Marware**
Cyrus Daboo, Apple
Bernard Desruisseaux, Oracle

Status: **Published**

This document was created by the CalDAV Technical Committee of the Calendaring and Scheduling Consortium. It presents a list of features in the form of requirements for the scheduling extensions to CalDAV [RFC 4791], that is, the extensions to the Web Distributed Authoring and Versioning (WebDAV) [RFC 2518] protocol to specify a standard way of exchanging and processing scheduling messages based on the iCalendar Transport-Independent Interoperability Protocol (iTIP) [RFC 2446].

STATEMENT OF INTELLECTUAL PROPERTY RIGHTS

This document and the information it contains is the work product of The Calendaring and Scheduling Consortium ("Consortium"), and as such, the Consortium claims all rights to any intellectual property contained herein.

STATEMENT OF APPROPRIATE USAGE

Standards Setting Organizations and others who find this document is of use in their work are hereby granted the right to copy, redistribute, incorporate into their own documents, make derivative works from, and otherwise make further use of the document and the material it contains at no cost and without seeking prior permission from the Consortium, subject to properly attributing the source if unmodified to the Consortium and notifying the Consortium of its use according to the guidelines below:

1. If the document is excerpted or used in its entirety in another document, the text must remain unchanged and a complete citation must be supplied referencing the full title, version, date, and appropriate section/subsection/ paragraph identification from the original document.

2. A normative or informative reference to this document may be used in place of excerpting or incorporating the entire original document. Such references should include the full title, version, date, and appropriate section/subsection/paragraph identification from the Consortium document being referenced.

3. In either case, the user referencing or excerpting a Consortium document is requested to notify the Consortium of the referencing specification and to provide the Consortium with an appropriate link or other way of reviewing the specification.

DISCLAIMER OF WARRANTY

THIS DOCUMENT AND THE INFORMATION IT CONTAINS IS PROVIDED ON AN "AS IS" BASIS, WITHOUT ANY WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, FROM THE CONSORTIUM, ITS CONTRIBUTORS, AND THE ORGANIZATIONS ITS CONTRIBUTORS REPRESENT OR ARE SPONSORED BY (IF ANY), INCLUDING, WITHOUT LIMITATION, ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR ANY PARTICULAR PURPOSE, AND NON-INFRINGEMENT.

1. Introduction

This document presents a list of features in the form of requirements for the scheduling extensions to CalDAV [RFC 4791], that is, the extensions to the Web Distributed Authoring and Versioning (WebDAV) [RFC 2518] protocol to specify a standard way of exchanging and processing scheduling messages based on the iCalendar Transport-Independent Interoperability Protocol (iTIP) [RFC 2446].

2. Terminology

This document makes use of the following terms:

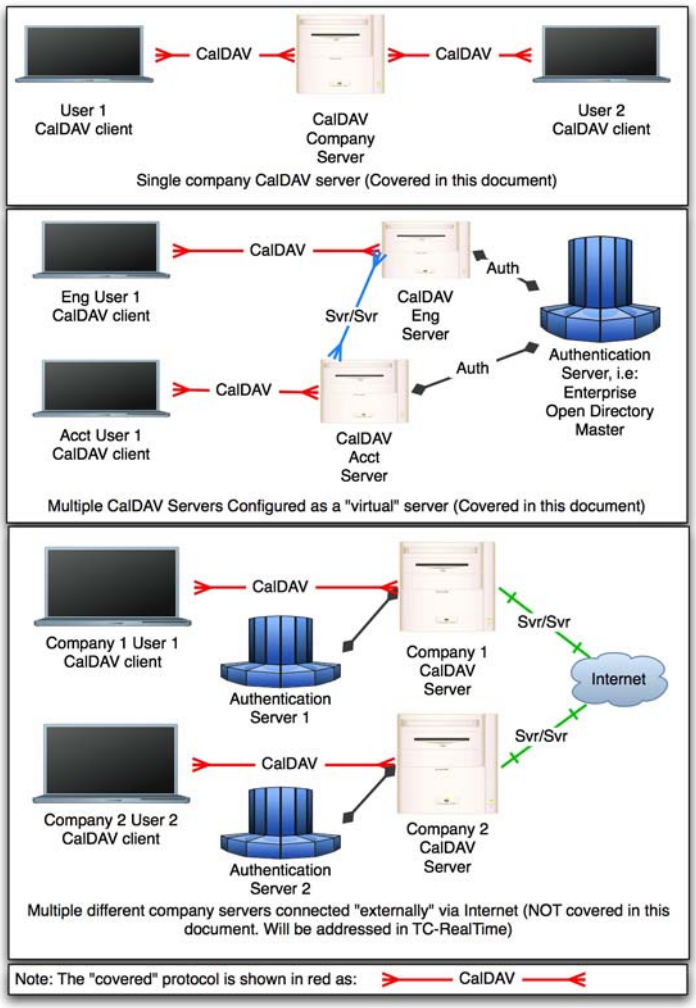
- **Free Busy request**: A VFREEBUSY object included in an HTTP POST request, targeting one or more calendar users for their Free/Busy information, given a time range.
- **Scheduling message**: An iCalendar object that represents one of the following:
    - an event publication, such as a meeting or class
    - an event invitation, to a meeting
    - an event modification, such as a meeting time change
    - an event cancellation
    - a to-do assignment, such as a task to perform or deadline to meet
    - a to-do modification, such as a deadline update
    - a to-do cancellation
    - a journal publication, such as a document addition to the meeting
    - a journal cancellation.
    - or any other iTIP related scheduling action.
- Local user:
    - a user who’s calendar (inbox) resides on the same (local) server as the client is connected to.
- Remote user:
    - a user who’s calendar (inbox) resides on a remote server, but still within the calendar domain for addresses and principals. This server would most likely be clustered or federated, so the user is equivalent to a “local” user.
- Internal user:
    - either a local or remote user.
- External user:
    - a user who’s calendar (inbox) resides on a truly remote server, outside the calendar domain.

- Calendar domain:
    - the domain in which calendar addresses and principals are consistently defined and can be used in the protocol without external lookups or mechanisms – i.e.: the domain of the Internal (local and remote) users.
- Authentication domain:
    - the domain that an authentication principal identifier is valid (server, department, corporate, etc)
- Authentication principal:
    - an identifier, returned from the authentication server, that is used as an authentication token in the protocol requests for authorization/permission validation(s).
- Calendar addresses:
    - calendar user addresses are typically specified as mailto URI, but other types of URI are allowed.
- Group addresses:
    - A collection of user addresses (internal and/or external). A group address may itself be internal or external. Note that there is weak definition/support for “groups” in existing Internet standards (members and access rights) and so, we may not be able to require/deliver the level of “group” support we would like.

3. Scope

The scope of this document is the state-less protocol between a CalDAV client and server. We will address boundary protocol exchanges, where appropriate for clarity. CalDAV server to server protocol is out of scope. This document assumes the following:

- That all client to server protocol users are within a single authentication domain, either on a single server, or part of a calendar domain, well known and/or under local control. i.e.: user and server “lookups” are configured correctly and will resolve, in all cases, and that resolution to a principal identifier, as used by the CalDAV protocol is also correctly configured and working. By “remote” user or server, we mean not the one that your client is connected to, but still within the authentication domain and/or accessible by a known URI, using the same principal identifier. Scheduling with external users and servers, i.e.: ones outside of your authentication domain and control, is not covered in this document. This will be addressed by separate requirements defined by the Realtime Technical Committee.



4. Free Busy Requirements

4.1. Free Busy Access

4.1.1. Calendar user addresses must be used to identify the users for which free busy information is being requested.

Free busy queries should be targeted at calendar users addresses, not at specific calendars owned by those users. It should not be required to use specific calendar names to obtain free busy information. For security concerns, the calendar name(s) should NOT be returned in the response.

4.1.2. It must be possible for a user to query the free busy information of any (internal or external) user with a single request targeted at their CalDAV server.

A free busy query should allow a user to specify any calendar user address (URI). The server should differentiate an error response where the calendar user is internal but access is denied (privilege error), and where the calendar user is external and the server doesn't know how to get that user's calendar (unknown user error) – so that the client may try alternative methods to get the external user's calendar if it is able. The protocol should support error codes for the following cases:

-For unknown external users for which access to free busy information is not available -> User unknown error

-For unknown internal users for which free busy information is not available -> User Not Found error

-For users for which access to free busy information is not granted to the requester -> Privilege error

4.1.3. It must be possible for a user to query the free busy information of one or more users with a single request targeted at their CalDAV server.

A user should be able to get the free busy information of multiple users in a single request to the server.

4.1.4. The response to a free busy query must contain free busy information separated per queried calendar user.

Often times the organizer of an event is unable to schedule the event at a time where all the attendees are free. The organizer should have access to the individual free busy information to know which users his event will create a conflict with (e.g., a manager may decide to double book one of the attendees under his direct control, but may want to avoid double booking his own manager). The server must not aggregate free-busy information for different users, so that the client software will be able to present the free-busy information or error status on a per-user basis.

4.1.5. It must be possible to specify the calendar user address of a group in a free busy query. The group may be internal or external to the calendaring domain. Group members can be internal or external to the calendaring domain.

This may require additional human interaction to know what an external group is, and to be able to specify it in the request – or an understanding

that external groups have an email address too. A separate VFREEBUSY component should be returned per group member. The protocol should support group names as an element for the request AND error codes for the following cases:

- members for which free busy information is not available -> User Not Found error
- members for which access to free busy information is not granted to the requester -> Privilege error
- groups for which the membership info is only available to group members -> Privilege error

The protocol should support group names as an element for the request but suppress some error codes for privacy, to insure clients can't infer group members. See 4.1.2 for more response information.

4.1.6. It must be possible for a user to specify that only free busy periods that overlap a specified time range should be returned in a response to a free busy query.

Typically, users are only interested in the free busy information of other users for a limited period of time (e.g., this week only).

4.1.7. It must be possible for a user to perform a free busy query on behalf of another user.

The administrative assistant of a manager must be able to query the free busy information of users that have granted the manager the right to query their free busy information. See 4.3.2

4.1.8. The response to a free busy query must be returned synchronously to the client with the free busy information of the calendar users for which information was available.

Users want to get an immediate response to a free busy query to be able to schedule an event immediately with the same people whose free busy information was queried. Note: Given that the response to a free busy query must be synchronous, there is no purpose in keeping a copy of a free busy query on the CalDAV server.

4.1.9. The client should be able to specify a time-out value and the server should honor this value in any fan-outs to other servers.

Timeouts must be considered for the following cases:

-Network errors/timeouts client to server.

-Client to server timeout due to server busy (possible partial response).

-Client to server timeout due to server to server fanout, with fanout timeout (possible partial response)

4.1.10. For each calendar user for which free busy information was requested, a specific request status code must be returned (good and/or bad).

Different status codes could be used for the following conditions: (1) the information was correctly returned, (2) the calendar user address is invalid, (3) the calendar user address doesn't exist, (4) free busy information is not available synchronously for this calendar user - timeout, or (5) permission has been denied to access the free busy information of this calendar user, etc. iTIP status codes should be used.

4.2. Free Busy Management

4.2.1. It must be possible for a user to specify which calendars impact their free busy information. This calendar set can contain calendars that are owned or not owned by the user, and they could be internal or external to their Calendaring domain.

A user may own calendars that don't impact their availability and their availability may be impacted by calendars that they do not own. As such, a user should be able to specify any calendar on any server(s) which may impact their availability.

4.2.2. It must be possible for a user to locate and maintain the resource that specifies which calendar collections contribute to the free busy information of a specific user given their calendar user address.

Most users only need to locate the resource that specifies the calendar collections that contributes to their own free busy information, but administrative assistants may need to locate/edit/manage the resource that specifies the calendar collections that contributes to the free busy information of their managers. The protocol should support granting permissions to and allowing others to manage these resources on behalf of oneself.

4.3. Free Busy Access Control

These may be "system" or "solution" requirements, and not necessarily "protocol" requirements. Authentication is handled at the HTTP level and is outside the scope of the protocol. The protocol deals with an authorization "principal" which is then compared to various properties to determine privileges. The client will have to present the user with various options to support this, as described below.

4.3.1. It must be possible for a user to specify who is granted the right to query their free busy information.

Users should be able to specify which users are granted the right to query their free busy information. Users that are allowed to query free busy information will then be subject to the privilege granted to them at the calendar object resource level (i.e., CALDAV:read-free-busy privilege).

4.3.2. It must be possible for a user to specify who is granted the right to perform a free busy query on their behalf.

A manager should be able to grant their administrative assistant the right to query free busy information of other users on their behalf. When the administrative assistant is performing a free busy query on behalf of the manager, authorization verification should be done against the manager's identity (principal). i.e.: a request delegate. The system/solution should support property storage of grants/rights to other ACLs (as a delegate). See 4.1.7

4.3.3. It must be possible for a user to specify who is granted the right to grant other users the right to query his free busy information and/or perform a free busy query on their behalf.

A manager may want to grant their administrative assistant the right to manage their free busy access control. i.e.: an account management delegate. The system/solution should support storage of grants/rights to other ACLs (as an admin delegate).

4.4. Free Busy Requirements Left Out

This section describes issues that were considered by the Technical Committee as it was working on this document, but were not considered to be free busy scheduling requirements, or they were otherwise out of scope. However, the Technical Committee felt it was useful to include these here with an explanation of why they were left out.

4.4.1. It must be possible to specify a sub-address in a calendar user address (e.g., mailto:john+work@acme.com) to specify a specific calendar for which free busy information is being queried.

This requirement has been left out since it is already addressed by the CALDAV:free-busy-query report defined in CalDAV calendar-access.

4.4.2. It must be possible for a user to restrict the number of free time periods returned in a response to a free busy query.

This requirement was left out because iTIP doesn't provide a way to specify such a limit/restriction in a VFREEBUSY request.

While a server could take advantage of this limit to reduce its load when free busy information is requested for a single user, the same isn't true when free busy information is requested for multiple users.

4.4.3. It must be possible to get a separate VFREEBUSY component per queried calendar user or an aggregated VFREEBUSY for all the queried calendar users, or both in a response to a free busy request.

This requirement was left out because iTIP doesn't provide a way to specify such a limit/restriction in a VFREEBUSY request.

Aggregated VFREEBUSY could only be returned if all the individual VFREEBUSY had successfully been retrieved.

4.4.4. It must be possible for a user to specify that only free time periods (i.e., FBTYPE=FREE) should be returned in a response to a free busy query.

This requirement was left out because iTIP doesn't provide a way to specify such a limit/restriction in a VFREEBUSY request.

4.4.5. It must be possible for a user to specify that only free time periods (i.e., FBTYPE=FREE) with a minimum duration should be returned in a response to a free busy query.

This requirement was left out because iTIP doesn't provide a way to specify such a limit/restriction in a VFREEBUSY request.

4.4.6. It must be possible for a user to specify a list of recurrence instances (i.e., UID and RECURRENCE-ID) that should be ignored during the computation of free busy information.

This requirement was left out because iTIP doesn't provide a way to specify such a limit/restriction in a VFREEBUSY request.

In the process of rescheduling a specific recurrence instance, it would be useful to obtain the free busy information, of the attendees, that doesn't take into account this specific recurrence instance.

4.4.7. It should be possible to access free busy information easily from a simple HTTP client, i.e.: a browser.

Testers/Users may want to publish an HTTP URL to which their free busy information would be easily available to users with a simple HTTP browser client (e.g., `http://cal.example.com/freebusy/bernard.ifb`). Free busy information retrieved this way could be restricted to a limited time range (e.g., previous week to next two months). The protocol should not be so complex as to prevent simple, single requests from working. i.e.: no session state across multiple requests.

-May require HTTP Auth - username/password, as opposed to the "principal"

5. Scheduling Requirements

5.1. Generic

5.1.1. Calendar user addresses must be used to identify the users to whom the scheduling messages are being sent.

See 4.1.1.

5.2. Organizer

5.2.1. It must be possible for an organizer to send a scheduling message to one or more users that may or may not be listed as an attendee in the scheduling message with a single request.

The user will receive the scheduling message, but the user IS NOT listed as an attendee. This means that the message is FYI only, and the user is not expected to respond to the scheduling request. The recipient information must not be exposed to any other recipient or attendee for security/privacy issues.

If it is desired to communicate that this user was informed of the schedule request, they may be listed as a non-required attendee in the iCalendar data, which means everyone will know that user may have received the message.

5.2.2. It must be possible for an organizer to send a scheduling message to internal and external users with a single request targeted at their CalDAV server.

The implication here is that the local CalDAV server must be the directory lookup service and the forwarder of the request – not the requesting client.

-Server to server timeouts should produce service unavailable error, will retry, like SMTP.

-Server to server security (proxy), since the organizer could be remote.

Note: server to server protocol for external users is out of scope for this document

5.2.3. It must be possible for an organizer to send a scheduling message to multiple users without letting those users know about the other users that were also sent this message. The recipient list must be purged.

The organizer of an event may want to send a copy of a meeting invitation to the manager of one of the attendee to inform him. The organizer doesn't necessarily want the attendee to know that a copy of the meeting invitation was sent to his manager. This "manager" recipient is not an attendee, and thus, cannot respond to the message.

-NOTE: Every recipient of a scheduling message will get the list of attendees (required/not-required to attend), and thus, will know about all the attendees. They will not know about any of the recipients.

5.2.4. It must be possible to specify the calendar user address of a group when sending a scheduling message. The group may be internal or external to the calendar domain. Group members can be internal or external.

Noting that Internet group support is weak, at best, the protocol must not prevent a group request, where all member permissions/grants are correct, from allowing a scheduling message to be posted. The minimum functionality of a "group" being a convenient way of maintaining a collection of existing users must be supported.

Group permissions/grants do not override individual member permissions/grants. You need both to successfully receive information.

Groups may be inclusive (all members participate) or exclusive (only one help desk person must respond).

See 4.1.2/4.1.5 for response status/errors.

5.2.5. It must be possible for the organizer to properly handle, on a per recurrence instance basis, attendee scheduling replies received out of order or received more than once (duplicate scheduling replies).

The organizer may receive attendee replies to a scheduling request out of order. The organizer should have a way to know whether they should ignore a reply from an attendee given that a more recent reply was already received from that attendee.

5.2.6. It must be possible for an organizer to determine, on a per recurrence instance basis, if a scheduling reply from a

given attendee is making reference to the last scheduling request sent to that given attendee.

The organizer may receive a reply, from an attendee, that makes reference to a scheduling request that preceded the last scheduling request sent to that attendee.

5.3. Attendee/Recipient

5.3.1. It must be possible for an attendee to receive a scheduling message sent by an internal or external organizer.

This is currently an issue due to the existing requirement of explicitly granting access to each submitter.

5.3.2. It must be possible for the attendee to properly handle, on a per recurrence instance basis, organizer scheduling messages received out of order or received more than once (duplicate scheduling messages).

Attendees may receive requests from an organizer out of order or multiple times. The attendee should have a way to detect out of order or duplicate requests and ignore them. This requires maintaining enough state information on the server and/or client to detect any problems.

5.3.3. It must be possible for an attendee to send a scheduling reply in response to a scheduling message received from a internal or external organizer.

If an attendee receives a scheduling request, they should be able to respond to it, if a response is required.

5.3.4. It must be possible for an attendee to respond more than once to a scheduling message received from a internal or external organizer, with a different response (accept/decline).

Attendees need to be allowed to “change their minds” about a reply they have previously sent to an organizer. So they must be able to send additional replies to the same scheduling request, each indicating a change in status. These replies need to be appropriately tracked by the organizer to ensure proper sequencing.

5.3.5. It must be possible for an attendee/recipient to know the originator of a scheduling message.

The originator might be a different user than the organizer (e.g., a calendar user forwarding a scheduling message, a calendar user sending a scheduling message on behalf of the organizer).

5.3.6. It must be possible for an attendee to receive and respond to scheduling messages where the original organizer has been replaced by a new one.

Security issue: *We must notify the client and/or log on the server that the organizer has changed, to insure no one is masquerading*.

5.4. Scheduling Access Control

See the previous section for delegation requirements.

5.4.1. It must be possible for a user to specify who he will accept a schedule request from.

Users should be able to specify which users are granted the right to schedule their time – i.e.: their managers and/or administrative assistant.

5.4.2. It must be possible for a user to specify who is granted the right to accept schedule requests on their behalf.

A manager should be able to grant their administrative assistant the right to submit/accept schedule requests.

5.4.3. It must be possible for a user to specify who he will accept schedule replies from.

The protocol is stateless – the original invite should/may not be stored on the server. New attendees may have been added and the organizer may have changed from the original message. The user must specify who he/she will receive schedule replies from.

5.5. Left out requirements

This section describes issues that were considered by the Technical Committee as it was working on this document, but were not considered to be protocol scheduling requirements, or they were otherwise out of scope. However, the Technical Committee felt it was useful to include these here with an explanation of why they were left out.

5.5.1. Ability to ask the server to strip the ATTENDEE and/or BCC list…

Handled by the requirement to BCC (blind copy) a recipient.

5.5.2. It must be possible for an attendee to forward a scheduling message to internal or external uninvited calendar user.

A user may wish to forward the request to their boss for permission, or their admin/delegate to manage. Security issue: A lot of potentially sensitive information is contained in the message. Difficult to differentiate (Re:/FYI:/or real scheduling message) – recommend to use email. Note: it's not a scheduling request (client to server) so out of band, so out of scope.

5.5.3. It must be possible for an attendee to "mark" on a per instance basis, whether the scheduling message has been (1) read/unread, (2) processed/unprocessed, and (3) responded (accept/decline)/non-responded (yet).

The client application needs to be able to display these states, and the server needs to be able to store these statuses– via message sequencing, etc. This requirement is out of scope for the scheduling protocol