Workgroup: JMAP
Internet-Draft: draft-ietf-jmap-tasks-06
Published: 10 March 2023
Intended Status: Standards Track
Expires: 11 September 2023
Authors: J.M. Baum, Ed.
audriga
H.J. Happel, Ed.
audriga

# JMAP for Tasks

## Abstract

This document specifies a data model for synchronizing task data with a server using JMAP.

## Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

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."

This Internet-Draft will expire on 11 September 2023.

**Table of Contents**

## 1. Introduction

JMAP ([RFC8620] – JSON Meta Application Protocol) is a generic protocol for synchronizing data, such as mail, calendars or contacts, between a client and a server. It is optimized for mobile and web environments, and aims to provide a consistent interface to different data types.

JMAP for Calendars ([I-D.ietf-jmap-calendars]) defines a data model for synchronizing calendar data between a client and a server using JMAP. The data model is designed to allow a server to provide consistent access to the same data via CalDAV [RFC4791] as well as JMAP.

While CalDAV defines access to tasks, JMAP for Calendars does not. This specification fills this gap and defines a data model for synchronizing task data between a client and a server using JMAP. It is built upon JMAP for Calendars and reuses most of its definitions. For better readability, this document only outlines differences between this specification and JMAP for Calendars. If not stated otherwise, the same specifics that apply to Calendar, CalendarEvent and CalendarEventNotification objects as defined in the aforementioned specification also apply to similar data types introduced in this specification.

### 1.1. Notational Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

Type signatures, examples, and property descriptions in this document follow the conventions established in Section 1.1 of [RFC8620]. Data types defined in the core specification are also used in this document.

### 1.2. The LocalDate Data Type

Where LocalDate is given as a type, it means a string in the same format as Date (see [RFC8620], Section 1.4), but with the time-offset omitted from the end. For example, 2014-10-30T14:12:00. The interpretation in absolute time depends upon the time zone for the task, which may not be a fixed offset (for example when daylight saving time occurs).

### 1.3. The Duration Data Type

Where Duration is given as a type, it means a length of time represented by a subset of the ISO 8601 duration format, as defined in [RFC8984], Section 1.4.6.

### 1.4. Terminology

The same terminology is used in this document as in the core JMAP specification, see [RFC8620], Section 1.6.

The terms ParticipantIdentity, TaskList, Task and TaskNotification are used to refer to the data types defined in this document and instances of those data types.

### 1.5. Data Model Overview

Similar to JMAP for Calendar, an Account (see [RFC8620], Section 1.6.2) contains zero or more TaskList objects, which is a named collection of Tasks belonging to a Principal (see [I-D.ietf-jmap-sharing] Section XXX). Task lists can also provide defaults, such as alerts and a color, to apply to tasks in the calendar. Clients commonly let users toggle visibility of tasks belonging to a particular task list on/off.

A Task is a representation of a single task or recurring series of Tasks in JSTask [RFC8984] format. Recurrence rules and alerts as

defined in JMAP for Calendars (see [I-D.ietf-jmap-calendars] Section XXX) apply.

Just like the CalendarEventNotification objects (see [I-D.ietf-jmap-calendars] Section XXX), TaskNotification objects keep track of the history of changes made to a task by other users. Similarly, the ShareNotification type (see [I-D.ietf-jmap-sharing] Section XXX) notifies the user when their access to another user's task list is granted or revoked.

Use cases for task systems vary. Only a few systems will require implementation of all available features defined within this specification. For this reason, this document describes several extensions to the core task properties and objects through which support for a certain feature MUST be advertised via capabilities. In addition to the core features advertised via urn:ietf:params:jmap:tasks support for recurrences, assignees, alerts, localizations as well as custom time zones can be advertised. As defined in [RFC8620], servers SHOULD throw an "urn:ietf:params:jmap:error:unknownCapability" error when a client uses a capability that the server does not understand. However, a Task object might just contain data that the server does not understand. In this case, the server SHOULD save it and ignore its existence.

### 1.6. Addition to the Capabilities Object

The capabilities object is returned as part of the JMAP Session object; see [RFC8620], Section 2. This document defines six additional capability URIs.

#### 1.6.1. urn:ietf:params:jmap:tasks

This represents support for the core properties and objects of the TaskList, Task and TaskNotification data types and associated API methods. The value of this property in the JMAP Session capabilities property is an empty object.

The value of this property in an account's accountCapabilities property is an object that MUST contain the following information on server capabilities and permissions for that account:

- **minDateTime**: LocalDate The earliest date-time the server is willing to accept for any date stored in a Task.
- **maxDateTime**: LocalDate The latest date-time the server is willing to accept for any date stored in a Task.
- **mayCreateTaskList**: Boolean If true, the user may create a task list in this account.

#### 1.6.2. urn:ietf:params:jmap:tasks:recurrences

This represents support for the recurrence properties and objects of the TaskList, Task and TaskNotification data types and associated API methods. The value of this property in the JMAP Session capabilities property is an empty object.

The value of this property in an account's accountCapabilities property is an object that MUST contain the following information on server capabilities and permissions for that account:

* **maxExpandedQueryDuration**: Duration The maximum duration the user may query over when asking the server to expand recurrences.

#### 1.6.3. urn:ietf:params:jmap:tasks:assignees

This represents support for the assignee properties and objects of the TaskList, Task and TaskNotification data types and associated API methods. The value of this property in the JMAP Session capabilities property is an empty object.

The value of this property in an account's accountCapabilities property is an object that MUST contain the following information on server capabilities and permissions for that account:

* **maxParticipantsPerTask**: UnsignedInt|null The maximum number of participants a single task may have, or null for no limit.

#### 1.6.4. urn:ietf:params:jmap:tasks:alerts

This represents support for the alerts properties and objects of the TaskList, Task and TaskNotification data types and associated API methods. The value of this property in the JMAP Session capabilities property and the account's accountCapabilities property is an empty object.

#### 1.6.5. urn:ietf:params:jmap:tasks:multilingual

This represents support for the multilingual properties and objects of the TaskList, Task and TaskNotification data types and associated API methods. The value of this property in the JMAP Session capabilities property and the account's accountCapabilities property is an empty object.

#### 1.6.6. urn:ietf:params:jmap:tasks:customtimezones

This represents support for the custom time zone properties and objects of the TaskList, Task and TaskNotification data types and associated API methods. The value of this property in the JMAP

Session capabilities property and the account's accountCapabilities property is an empty object.

## 2. Principals and Sharing

For systems that also support JMAP Sharing [I-D.ietf-jmap-sharing], the tasks capability is used to indicate that this principal may be used with task management.

### 2.1. Principal Capability urn:ietf:params:jmap:tasks

A "urn:ietf:params:jmap:tasks" property is added to the Principal "capabilities" object, the value of which is an object with the following properties:

* **accountId**: Id|null Id of Account with the urn:ietf:params:jmap:tasks capability that contains the task data for this principal, or null if none (e.g. the Principal is a group just used for permissions management), or the user does not have access to any data in the account. The corresponding Account object can be found in the principal's "accounts" property, as per [RFC XXX].
* **mayShareWith**: Boolean May the user add this principal as a task sharee (by adding them to the shareWith property of a task list, see Section XXX)?
* **sendTo**: String[String]|null If this principal may be added as a participant to a task, this is the Participant#sendTo property to add (see [RFC8984], Section 4.4.5) for scheduling messages to reach it.

## 3. TaskLists

A TaskList is a named collection of tasks. All tasks are associated with exactly one TaskList.

A **TaskList** object has the following core properties:

* **id**: Id (immutable; server-set) The id of the task list.

* **role**: String|null (default: null) Denotes the task list has a special purpose. This MUST be one of the following:

  - inbox: This is the principal's default task list;
  - trash: This task list holds messages the user has discarded;

* **name**: String The user-visible name of the task list. This may be any UTF-8 string of at least 1 character in length and maximum 255 octets in size.

* **description**: String|null (default: null) An optional longer-form description of the task list, to provide context in shared environments where users need more than just the name.

* **color**: String|null (default: null) A color to be used when displaying tasks associated with the task list.

  If not null, the value MUST be a case-insensitive color name taken from the set of names defined in Section 4.3 of CSS Color Module Level 3 COLORS, or an RGB value in hexadecimal notation, as defined in Section 4.2.1 of CSS Color Module Level 3.

  The color SHOULD have sufficient contrast to be used as text on a white background.

* **keywordColors** String[String]|null (default:null) A map of keywords to the colors used when displaying the keywords associated to a task. The same considerations, as for color above, apply.

* **categoryColors** String[String]|null (default:null) A map of categories to the colors used when displaying the categories associated to a task. The same considerations, as for color above, apply.

* **sortOrder**: UnsignedInt (default: 0) Defines the sort order of task lists when presented in the client's UI, so it is consistent between devices. The number MUST be an integer in the range $0 \leq sortOrder < 2^{31}$.

  A task list with a lower order should be displayed before a list with a higher order in any list of task lists in the client's UI. Task lists with equal order SHOULD be sorted in alphabetical order by name. The sorting should take into account locale-specific character order convention.

* **isSubscribed**: Boolean Has the user indicated they wish to see this task list in their client? This SHOULD default to false for task lists in shared accounts the user has access to, and true for any new task list created by the user themselves.

  If false, the task list should only be displayed when the user explicitly requests it or to offer it for the user to subscribe to.

* **timeZone**: String|null (default: null) The time zone to use for tasks without a time zone when the server needs to resolve them into absolute time, e.g., for alerts or availability calculation. The value MUST be a time zone id from the IANA Time Zone Database TZDB. If null, the timeZone of the account's associated Principal

will be used. Clients SHOULD use this as the default for new tasks in this task list, if set.

* **workflowStatuses**: String[] (default: [completed, failed, in-process, needs-action, cancelled, pending]) Defines the allowed values for workflowStatus. The default values are based on the values defined within [RFC8984], Section 5.2.5 and pending. pending indicates the task has been created and accepted, but it currently is on-hold.

  As naming and workflows differ between systems, mapping the status correctly to the present values of the Task can be challenging. In the most simple case, a task system may support merely two states - done and not-done. On the other hand, statuses and their semantic meaning can differ between systems or task lists (e.g. projects). In case of uncertainty, here are some recommendations for mapping commonly observed values that can help during implementation:

  - completed: done (most simple case), closed, verified, ...
  - in-process: in-progress, active, assigned, ...
  - needs-action: not-done (most simple case), not-started, new, ...
  - pending: waiting, deferred, on-hold, paused, ...

* **shareWith**: Id[TaskRights]|null (default: null) A map of Principal id to rights for principals this task list is shared with. The principal to which this task list belongs MUST NOT be in this set. This is null if the task list is not shared with anyone. May be modified only if the user has the mayAdmin right. The account id for the principals may be found in the urn:ietf:params:jmap:principals:owner capability of the Account to which the task list belongs.

* **myRights**: TaskRights (server-set) The set of access rights the user has in relation to this TaskList.

  - The user may fetch the task if they have the mayReadItems right on any task list the task is in.
  - The user may remove a task from a task list (by modifying the task's "taskListId" property) if the user has the appropriate permission for that task list.
  - The user may make other changes to the task if they have the right to do so in *all* task list to which the task belongs.

A **TaskRights** object has the following properties:

* **mayReadItems**: Boolean The user may fetch the tasks in this task list.

*__mayWriteAll__: Boolean The user may create, modify or destroy all tasks in this task list, or move tasks to or from this task list. If this is true, the mayWriteOwn, mayUpdatePrivate and mayRSVP properties MUST all also be true.

*__mayWriteOwn__: Boolean The user may create, modify or destroy a task on this task list if either they are the owner of the task (see below) or the task has no owner. This means the user may also transfer ownership by updating a task so they are no longer an owner.

*__mayUpdatePrivate__: Boolean The user may modify the following properties on all tasks in the task list, even if they would not otherwise have permission to modify that task. These properties MUST all be stored per-user, and changes do not affect any other user of the task list.

The user may also modify the above on a per-occurrence basis for recurring tasks (updating the recurrenceOverrides property of the task to do so).

*__mayRSVP__: Boolean The user may modify the following properties of any Participant object that corresponds to one of the user's ParticipantIdentity objects in the account, even if they would not otherwise have permission to modify that task

  -participationStatus
  -participationComment
  -expectReply

If the task has its "mayInviteSelf" property set to true (see Section XXX), then the user may also add a new Participant to the task with a sendTo property that is the same as the sendTo property of one of the user's ParticipantIdentity objects in the account. The roles property of the participant MUST only contain "attendee".

If the task has its "mayInviteOthers" property set to true (see Section XXX) and there is an existing Participant in the task corresponding to one of the user's ParticipantIdentity objects in the account, then the user may also add new participants. The roles property of any new participant MUST only contain "attendee".

The user may also do all of the above on a per-occurrence basis for recurring tasks (updating the recurrenceOverrides property of the task to do so).

*__mayAdmin__: Boolean The user may modify sharing for this task list.

*   **mayDelete**: Boolean (server-set) The user may delete the task list itself. This property MUST be false if the account to which this task list belongs has the *isReadOnly* property set to true.

The user is an **owner** for a task if the Task object has a "participant" property, and one of the Participant objects both:

1. Has the "chair" role.
2. Corresponds to one of the user's ParticipantIdentity objects in the account (as per Section XXX).

A task has no owner if its participant property is null or omitted.

### 3.1. Alerts extension

A TaskList has the following alerts properties:

*   **defaultAlertsWithTime**: Id[Alert]|null A map of alert ids to Alert objects (see [RFC8984], Section 4.5.2) to apply for tasks where "showWithoutTime" is false and "useDefaultAlerts" is true. Ids MUST be unique across all default alerts in the account, including those in other task lists; a UUID is recommended.

    If omitted on creation, the default is server dependent. For example, servers may choose to always default to null, or may copy the alerts from the default task list.

*   **defaultAlertsWithoutTime**: Id[Alert]|null A map of alert ids to Alert objects (see [RFC8984], Section 4.5.2) to apply for tasks where "showWithoutTime" is true and "useDefaultAlerts" is true. Ids MUST be unique across all default alerts in the account, including those in other task lists; a UUID is recommended.

    If omitted on creation, the default is server dependent. For example, servers may choose to always default to null, or may copy the alerts from the default task list.

### 3.2. TaskList/get

This is a standard "/get" method as described in [RFC8620], Section 5.1. The *ids* argument may be null to fetch all at once.

### 3.3. TaskList/changes

This is a standard "/changes" method as described in [RFC8620], Section 5.2.

### 3.4. TaskList/set

This is the standard "/set" method as described in [RFC8620], Section 5.3 but with the following additional request argument:

- ***onDestroyRemoveTasks**: Boolean (default: false)

If false, any attempt to destroy a TaskList that still has Tasks in it will be rejected with a TaskListHasTask SetError. If true, any Tasks that were in the TaskList will be removed from it, and they will be destroyed.

The "shareWith" properties may only be set by users that have the mayAdmin right. When modifying the shareWith property, the user cannot give a right to a principal if the principal did not already have that right and the user making the change also does not have that right. Any attempt to do so must be rejected with a forbidden SetError.

Users can subscribe or unsubscribe to a task list by setting the "isSubscribed" property. The server MAY forbid users from subscribing to certain task lists even though they have permission to see them, rejecting the update with a forbidden SetError.

The following properties may be set by anyone who is subscribed to the task list, and are all stored per-user:

- *name
- *color
- *sortOrder
- *timeZone
- *defaultAlertsWithoutTime
- *defaultAlertsWithTime

The "name", "color" and "timeZone" properties are initially inherited from the owner's copy of the task list, but if set by a sharee then they get their own copy of the property; it does not change for any other principals. If the value of the property in the owner's task list changes after this, it does not overwrite the sharee's value.

The "sortOrder", "isVisible", "defaultAlertsWithTime", and "defaultAlertsWithoutTime" properties are initially the default value for each sharee; they are not inherited from the owner.

The following extra SetError types are defined:

For "destroy":

- ***taskListHasTask**: The Task List has at least one task assigned to it, and the "onDestroyRemoveTasks" argument was false.

## 4. Tasks

A **Task** object contains information about a task. It is a JSTask object, as defined in [RFC8984]. However, as use-cases of task systems vary, this Section defines relevant parts of the JSTask object to implement the core task capability as well as several extensions to it. Only the core capability MUST be implemented by any task system. Implementers can choose the extensions that fit their own use case. For example, the recurrence extension allows having a Task object represent a series of recurring Tasks.

The core JSTask objects are Task, Link, Location, Relation and VirtualLocation. The core properties are JSTask's Metadata Properties (Section 4.1), What and Where Properties (Section 4.2), Task Properties (Section 5.2) as well as priority (Section 4.4.1), privacy (Section 4.4.3), replyTo (Section 4.4.4) and timeZone (Section 4.7.1).

On top of the JSTask properties, a Task object has the following additional core properties:

* **id**: Id (immutable; server-set) The id of the Task. This property is immutable. The id uniquely identifies a JSTask with a particular "uid" and "recurrenceId" within a particular account.

* **taskListId**: Id The TaskList id this task belongs to. A task MUST belong to exactly one TaskList at all times (until it is destroyed).

* **isDraft**: Boolean If true, this task is to be considered a draft. The server will not send any push notifications for alerts. This may only be set to true upon creation. Once set to false, the value cannot be updated to true. This property MUST NOT appear in "recurrenceOverrides".

* **utcStart**: UTCDate For simple clients that do not or cannot implement time zone support. Clients should only use this if also asking the server to expand recurrences, as you cannot accurately expand a recurrence without the original time zone.

  This property is calculated at fetch time by the server. Time zones are political, and they can and do change at any time. Fetching exactly the same property again may return different results if the time zone data has been updated on the server. Time zone data changes are not considered "updates" to the task.

  If set, the server will convert the UTC date to the task's current time zone using its current time zone data and store the local time.

This is not included by default and must be requested explicitly.

Floating tasks (tasks without a time zone) will be interpreted as per the time zone given as a Task/get argument.

Note that it is not possible to accurately calculate the expansion of recurrence rules or recurrence overrides with the utcStart property rather than the local start time. Even simple recurrences such as "repeat weekly" may cross a daylight-savings boundary and end up at a different UTC time. Clients that wish to use "utcStart" are RECOMMENDED to request the server to expand recurrences (see Section XXX).

* **utcDue**: UTCDate The server calculates the end time in UTC from the start/timeZone/duration properties of the task. This is not included by default and must be requested explicitly. Like utcStart, this is calculated at fetch time if requested and may change due to time zone data changes. Floating tasks will be interpreted as per the time zone given as a Task/get argument.

* **sortOrder**: UnsignedInt (default: 0) Defines the sort order of a task when presented in the client's UI, so it is consistent between devices. The number MUST be an integer in the range $0 <= sortOrder < 2^{31}$.

  A task with a lower order should be displayed before a task with a higher order in any list of tasks in the client's UI. Tasks with equal order SHOULD be sorted in alphabetical order by name. The sorting should take into account locale-specific character order convention.

* **workflowStatus**: String|null (default: null) Specifies the status of the task. The allowed values are defined within workflowStatuses. If set, progress MUST null.

### 4.1. Extensions to JSCalendar data types

This document extends one JSCalendar data type with new values.

#### 4.1.1. Relation

The keys for relation of the Relation object are extended by the following values:

* depends-on: This task depends on the referenced task in some manner. For example, a task may be blocked waiting on the other, referenced, task.
* clone: The referenced task was cloned from this task.
* duplicate: The referenced task is a duplicate of this task.
* cause: The referenced task was the cause for this task.

### 4.2. Additional JSCalendar properties

This document defines two new core JSCalendar properties for the JSTask object.

#### 4.2.1. estimatedWork

Type: UnignedInt|null (default: null)

This specifies the estimated amount of work the task takes to complete. In Agile software development or Scrum, it is known as complexity or story points. The number has no actual unit, but a larger number means more work.

#### 4.2.2. impact

Type: String|null (default: null)

This specifies the impact or severity of the task, but does not say anything about the actual prioritization. Some examples are: minor, trivial, major or block. Usually, the priority of a task is based upon its impact and urgency.

#### 4.2.3. checklists

Type: Id[Checklist]

A map of Checklist IDs to Checklist objects, containing checklist items. A **Checklist** object has the following properties:

- ***@type**: String Specifies the type of this object. This MUST be Checklist.
- ***title**: String (optional) Title of the list.
- ***checkItems**: CheckItem[] (optional) The items of the check list.

A **CheckItem** object has the following properties:

- ***@type**: String Specifies the type of this object. This MUST be CheckItem.
- ***title**: String Title of the item.
- ***sortOrder**: UnsignedInt (default: 0) Defines the sort order of CheckItem when presented in the client's UI. The number MUST be an integer in the range 0 <= sortOrder < 2^31. An item with a lower order should be displayed before an item with a higher order. Items with equal order SHOULD be sorted in alphabetical order by name. The sorting should take into account locale-specific character order convention.
- ***updated**: UTCDateTime (optional) The date and time when this item was updated
- ***isComplete**: Boolean

*__assigneee__: Person (optional) The person that this item is assigned to. The Person object has the following properties of which either principalId or uri MUST be defined:

  -__@type__: String (mandatory) Specifies the type of this object. This MUST be Person.
  -__name__: String (optional) The name of the person.
  -__uri__: String (optional) A URI value that identifies the person. This SHOULD be the scheduleId of the participant that this item was assigned to.
  -__principalId__: String (optional) The id of the Principal corresponding to the person, if any.

#### 4.2.4. comments

Type: Id[Comment] (optional).

Free-text comments associated with this task. A Comment object has the following properties:

*__@type__: String (mandatory). Specifies the type of this object. This MUST be Comment.
*__message__: String (mandatory). The free text value of this comment.
*__created__: UTCDateTime (optional). The date and time when this note was created.
*__updated__: UTCDateTime (optional). The date and time when this note was updated.
*__author__: Person (optional). The author of this comment. The Person object is defined in Section Section 4.2.3.

### 4.3. Properties similar in JMAP for Calendar

Attachments are described in [I-D.ietf-jmap-calendars], Section XXX.

### 4.4. Recurrences extension

For the recurrence extension, the JSCalendar objects NDay and RecurrenceRule as well as the Recurrence Properties (Section 4.3) need to be supported.

The Task will require the following further property:

*__baseTaskId__: Id|null (immutable; server-set) This is only defined if the *id* property is a synthetic id, generated by the server to represent a particular instance of a recurring Task (see Section XXX). This property gives the id of the "real" Task this was generated from.

#### 4.4.1. Properties similar in JMAP for Calendar

Recurrences and updates to recurrences are described in [I-D.ietf-jmap-calendars], Section XXX

### 4.5. Assignees extension

For the assignees extension, the JSCalendar object Participant as well as all Sharing and Scheduling Properties (Section 4.4) need to be supported.

The Task will require the following further property:

* **isOrigin**: Boolean (server-set) Is this the authoritative source for this task (i.e., does it control scheduling for this task; the task has not been added as a result of an invitation from another task management system)? This is true if, and only if:

  - the task's "replyTo" property is null; or
  - the account will receive messages sent to at least one of the methods specified in the "replyTo" property of the task.

Task objects MUST NOT have a "method" property as this is only used when representing iTIP [RFC5546] scheduling messages, not tasks in a data store.

#### 4.5.1. Per-user properties

In shared task lists, any top-level property registered in the IANA registry as "Is Per-User: yes" (see Section XXX) MUST be stored per-user. This includes:

* keywords
* color
* freeBusyStatus
* useDefaultAlerts
* alerts

The user may also modify these properties on a per-occurrence basis for recurring tasks; again, these MUST be stored per-user. Sharees initially receive the default value for each of these properties, not whatever value another user may have set.

When writing only per-user properties, the "updated" property MUST also be stored just for that user, if set. When fetching the "updated" property, the value to return is whichever is later of the per-user updated time or the updated time of the base task.

#### 4.5.2. Extensions to JSCalendar data types

The assignees extension extends one JSCalendar data type with new values.

##### 4.5.2.1. Participants

The Participant object, as defined in [I-D.ietf-calext-jscalendar] Section 4.4.6 is used to represent participants. This spec extends the keys for the roles property with the following value:

*assignee: the participant is expected to work on the task

#### 4.5.3. Additional JSCalendar properties

The assignees extension defines three new JSCalendar properties for the JSTask object.

##### 4.5.3.1. mayInviteSelf

Type: Boolean (default: false)

If true, any user that has access to the task may add themselves to it as a participant with the "attendee" role. This property MUST NOT be altered in the recurrenceOverrides; it may only be set on the master object.

This indicates the task will accept "party crasher" RSVPs via iTIP, subject to any other domain-specific restrictions, and users may add themselves to the task via JMAP as long as they have the mayRSVP permission for the task list.

##### 4.5.3.2. mayInviteOthers

Type: Boolean (default: false)

If true, any current participant with the "attendee" role may add new participants with the "attendee" role to the task. This property MUST NOT be altered in the recurrenceOverrides; it may only be set on the master object.

##### 4.5.3.3. hideAttendees

Type: Boolean (default: false)

If true, only the owners of the task may see the full set of participants. Other sharees of the task may only see the owners and themselves. This property MUST NOT be altered in the recurrenceOverrides; it may only be set on the master object.

### 4.6. Alerts extension

For the alerts extension, the JSCalendar objects Alert, AbsoluteTrigger and OffsetTrigger as well as all Alerts Properties (Section 4.4) need to be supported.

### 4.7. Multilingual extension

For the multilingual extension, the JSCalendar Multilingual Properties (Section 4.6) need to be supported.

### 4.8. Custom Time Zones extension

For the custom time zones extension, the JSCalendar objects TimeZone and TimeZoneRule as well as all Time Zone Properties (Section 4.7) need to be supported.

### 4.9. Task/get

This is the "CalendarEvent/get" method as described in [I-D.ietf-jmap-calendars], Section XXX.

TODO redefine this here. Similar to "TaskList/get" we only need to replace a few definitions. Copy+Paste most of the stuff.

### 4.10. Task/changes

This is a standard "/changes" method as described in [RFC8620], Section 5.2.

### 4.11. Task/set

This is the "CalendarEvent/set" method as described in [I-D.ietf-jmap-calendars], Section XXX.

TODO copy+paste most stuff from "CalendarEvent/set". It should be fine to just reference patching.

### 4.12. Task/copy

This is a standard "/copy" method as described in [RFC8620], Section 5.4.

### 4.13. Task/query

This is the "CalendarEvent/query" method as described in [I-D.ietf-jmap-calendars], Section XXX.

TODO copy+paste most stuff from "CalendarEvent/query". Mainly filtering should be different.

### 4.14. Task/queryChanges

This is a standard "/queryChanges" method as described in [RFC8620], Section 5.6. s

## 5. Task Notifications

The TaskNotification data type records changes made by external entities to tasks in task lists the user is subscribed to. Notifications are stored in the same Account as the Task that was changed.

This is the same specification as the CalendarEventNotification object from [I-D.ietf-jmap-calendars], Section XXX. Only the object properties differ slightly and are therefore fully described in this document.

### 5.1. Object Properties

The **TaskNotification** object has the following properties:

- ***id**: String The id of the TaskNotification.
- ***created**: UTCDate The time this notification was created.
- ***changedBy**: Person Who made the change. The Person object is defined in the Section Section 4.2.3.
- ***comment**: String|null Comment sent along with the change by the user that made it. (e.g. COMMENT property in an iTIP message), if any.
- ***type**: String This MUST be one of
  - -created
  - -updated
  - -destroyed
- ***TaskId**: String The id of the Task that this notification is about.
- ***isDraft**: Boolean (created/updated only) Is this task a draft?
- ***task**: JSTask The data before the change (if updated or destroyed), or the data after creation (if created).
- ***taskPatch**: PatchObject (updated only) A patch encoding the change between the data in the task property, and the data after the update.

If the change only affects a single instance of a recurring task, the server MAY set the task and taskPatch properties for the just that instance; the taskId MUST still be for the master task.

### 5.2. TaskNotification/get

This is a standard "/get" method as described in [RFC8620], Section 5.1.

### 5.3. TaskNotification/changes

This is a standard "/changes" method as described in [RFC8620], Section 5.2.

### 5.4. TaskNotification/set

This is a standard "/set" method as described in [RFC8620], Section 5.3.

Only destroy is supported; any attempt to create/update MUST be rejected with a forbidden SetError.

### 5.5. TaskNotification/query

This is a standard "/query" method as described in [RFC8620], Section 5.5.

#### 5.5.1. Filtering

A **FilterCondition** object has the following properties:

- ***after**: UTCDate|null The creation date must be on or after this date to match the condition.
- ***before**: UTCDate|null The creation date must be before this date to match the condition.
- ***type**: String The type property must be the same to match the condition.
- ***taskIds**: Id[]|null A list of task ids. The taskId property of the notification must be in this list to match the condition.

#### 5.5.2. Sorting

The "created" property MUST be supported for sorting.

### 5.6. TaskNotification/queryChanges

This is a standard "/queryChanges" method as described in [RFC8620], Section 5.6.

## 6. Security Considerations

All security considerations of JMAP [RFC8620] and JSCalendar [RFC8984] apply to this specification. Additional considerations specific to the data types and functionality introduced by this document are described in the following subsections.

## 7. IANA Considerations

### 7.1. JMAP Capability Registration for "tasks"

IANA will register the "tasks" JMAP Capability as follows:

Capability Name: urn:ietf:params:jmap:tasks

Specification document: this document

Intended use: common

Change Controller: IETF

Security and privacy considerations: this document, Section XXX

### 7.2. JMAP Capability Registration for "tasks:recurrences"

IANA will register the "tasks:recurrences" JMAP Capability as follows:

Capability Name: urn:ietf:params:jmap:tasks:recurrences

Specification document: this document

Intended use: common

Change Controller: IETF

Security and privacy considerations: this document, Section XXX

### 7.3. JMAP Capability Registration for "tasks:assignees"

IANA will register the "tasks:recurrences" JMAP Capability as follows:

Capability Name: urn:ietf:params:jmap:tasks:assignees

Specification document: this document

Intended use: common

Change Controller: IETF

Security and privacy considerations: this document, Section XXX

### 7.4. JMAP Capability Registration for "tasks:alerts"

IANA will register the "tasks:alerts" JMAP Capability as follows:

Capability Name: urn:ietf:params:jmap:tasks:alerts

Specification document: this document

Intended use: common

Change Controller: IETF

Security and privacy considerations: this document, Section XXX

### 7.5. JMAP Capability Registration for "tasks:multilingual"

IANA will register the "tasks:multilingual" JMAP Capability as follows:

Capability Name: urn:ietf:params:jmap:tasks:multilingual

Specification document: this document

Intended use: common

Change Controller: IETF

Security and privacy considerations: this document, Section XXX

### 7.6. JMAP Capability Registration for "tasks:customtimezones"

IANA will register the "tasks:customtimezones" JMAP Capability as follows:

Capability Name: urn:ietf:params:jmap:tasks:customtimezones

Specification document: this document

Intended use: common

Change Controller: IETF

Security and privacy considerations: this document, Section XXX

### 7.7. JSCalendar Property Registrations

Some IANA registrations for JSTask are described in JMAP for Calendars [I-D.ietf-jmap-calendars]. TODO explicitly list them.

IANA will register the following additional properties in the JSCalendar Properties Registry.

#### 7.7.1. estimatedWork

Property Name: estimatedWork

Property Type: UnignedInt|null

Property Context: JSTask

Reference: This document, Section XXX.

Intended Use: Common

Is per-user: no

#### 7.7.2. impact

Property Name: impact

Property Type: String|null

Property Context: JSTask

Reference: This document, Section XXX.

Intended Use: Common

Is per-user: no

## 8. Normative References


**[I-D.ietf-calext-jscalendar]** Jenkins, N. and R. Stepanek, "JSCalendar: A JSON Representation of Calendar Data", Work in Progress, Internet-Draft, draft-ietf-calext-jscalendar-32, 15 October 2020, <https://datatracker.ietf.org/doc/html/draft-ietf-calext-jscalendar-32>.

**[I-D.ietf-jmap-calendars]** Jenkins, N. and M. Douglass, "JMAP for Calendars", Work in Progress, Internet-Draft, draft-ietf-jmap-calendars-10, 4 December 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-jmap-calendars-10>.

**[I-D.ietf-jmap-sharing]** Jenkins, N., "JMAP Sharing", Work in Progress, Internet-Draft, draft-ietf-jmap-sharing-02, 5 October 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-jmap-sharing-02>.

**[RFC2119]** Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

**[RFC5546]** Daboo, C., Ed., "iCalendar Transport-Independent Interoperability Protocol (iTIP)", RFC 5546, DOI 10.17487/

RFC5546, December 2009, <https://www.rfc-editor.org/info/rfc5546>.

**[RFC8174]** Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

**[RFC8620]** Jenkins, N. and C. Newman, "The JSON Meta Application Protocol (JMAP)", RFC 8620, DOI 10.17487/RFC8620, July 2019, <https://www.rfc-editor.org/info/rfc8620>.

**[RFC8984]** Jenkins, N. and R. Stepanek, "JSCalendar: A JSON Representation of Calendar Data", RFC 8984, DOI 10.17487/RFC8984, July 2021, <https://www.rfc-editor.org/info/rfc8984>.

## 9. Informative References

**[RFC4791]** Daboo, C., Desruisseaux, B., and L. Dusseault, "Calendaring Extensions to WebDAV (CalDAV)", RFC 4791, DOI 10.17487/RFC4791, March 2007, <https://www.rfc-editor.org/info/rfc4791>.

## Authors' Addresses

Joris Baum (editor)
audriga
Alter Schlachthof 57
76137 Karlsruhe
Germany

Email: joris@audriga.com
URI: https://www.audriga.com

Hans-Joerg (editor)
audriga
Alter Schlachthof 57
76137 Karlsruhe
Germany

Email: hans-joerg@audriga.com
URI: https://www.audriga.com