

Workgroup: JMAP  
Internet-Draft: draft-ietf-jmap-tasks-03  
Published: 7 March 2022  
Intended Status: Standards Track  
Expires: 8 September 2022  
Authors: J.M. Baum, Ed.    H.J. Happel, Ed.  
          audriga            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 8 September 2022.

## **Copyright Notice**

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

## Table of Contents

- [1. Introduction](#)
  - [1.1. Notational Conventions](#)
  - [1.2. Terminology](#)
  - [1.3. Data Model Overview](#)
  - [1.4. Addition to the Capabilities Object](#)
    - [1.4.1. urn:ietf:params:jmap:tasks](#)
- [2. Principals](#)
  - [2.1. Principal Capability urn:ietf:params:jmap:tasks](#)
- [3. TaskLists](#)
  - [3.1. TaskList/get](#)
  - [3.2. TaskList/changes](#)
  - [3.3. TaskList/set](#)
- [4. Tasks](#)
  - [4.1. Additional JSCalendar properties](#)
    - [4.1.1. mayInviteSelf](#)
    - [4.1.2. mayInviteOthers](#)
    - [4.1.3. hideAttendees](#)
    - [4.1.4. relatedTo](#)
    - [4.1.5. progress](#)
    - [4.1.6. estimatedWork](#)
    - [4.1.7. impact](#)
  - [4.2. Participants](#)
  - [4.3. Properties similar in JMAP for Calendar](#)
  - [4.4. Task/get](#)
  - [4.5. Task/changes](#)
  - [4.6. Task/set](#)
  - [4.7. Task/copy](#)
  - [4.8. Task/query](#)
  - [4.9. Task/queryChanges](#)
- [5. Task Notifications](#)
  - [5.1. Object Properties](#)
  - [5.2. TaskNotification/get](#)
  - [5.3. TaskNotification/changes](#)
  - [5.4. TaskNotification/set](#)
  - [5.5. TaskNotification/query](#)
    - [5.5.1. Filtering](#)
    - [5.5.2. Sorting](#)
  - [5.6. TaskNotification/queryChanges](#)
- [6. Security Considerations](#)
- [7. IANA Considerations](#)
  - [7.1. JMAP Capability Registration for "tasks"](#)
  - [7.2. JSCalendar Property Registrations](#)
- [8. Normative References](#)
- [9. Informative References](#)
- [Authors' Addresses](#)

## 1. Introduction

JMAP ([\[RFC8620\]](#) - (U+2013) 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. 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.

### 1.3. 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 [\[I-D.ietf-calext-jscalendar\]](#) 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.

### 1.4. Addition to the Capabilities Object

The capabilities object is returned as part of the JMAP Session object; see [\[RFC8620\]](#), Section 2. This document defines one additional capability URI.

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

This represents support for 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' (U+2019)s accountCapabilities property is an object that MUST contain the following information on server capabilities and permissions for that account:

**\*shareesActAs:** String This MUST be one of:

- self - sharees act as themselves when using tasks in this account.
- secretary- sharees act as the principal to which this account belongs.

**\*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.

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

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

**\*mayCreateTaskList:** Boolean If true, the user may create a task list in this account.

## 2. Principals

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 tasks.

### 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.

**\*account:** Account|null The JMAP Account object corresponding to the accountId, null if none.

**\*sendTo:** String[String]|null If this principal may be added as a participant to a task, this is the map of methods for adding it, in the same format as Participant#sendTo in JSTask (see [[I-D.ietf-calext-jscalendar](#)], Section 4.4.5).

## 3. TaskLists

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

A **TaskList** object has the following 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.

**\*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  $0 \leq \text{sortOrder} < 2^{31}$ .

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.

**\*defaultAlertsWithTime:** Id[Alert]|null (default: null) A map of alert ids to Alert objects (see [[I-D.ietf-calext-jscalendar](#)], 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.

**\*defaultAlertsWithoutTime:** Id[Alert]|null (default: null) A map of alert ids to Alert objects (see [[I-D.ietf-calext-jscalendar](#)], 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.

**\*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.

**\*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 user requesting the object does not have the mayAdmin right, or 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 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. If the

shareesActAs account capability is "self", these properties MUST all be stored per-user, and changes do not affect any other user of the task list. If shareesActAs is "secretary", the values are shared between all users.

- keywords
- color
- useDefaultAlerts
- alerts

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.

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

### 3.1. 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.

### 3.2. TaskList/changes

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

### 3.3. TaskList/set

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

**\*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 "role" and "shareWith" properties may only be set by users that have the mayAdmin right. The value is shared across all users, although users without the mayAdmin right cannot see the value.

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 "timeZone", "defaultAlertsWithoutTime" and "defaultAlertsWithTime" properties are stored per-user if the task list account's "shareesActAs" capability is "self", and may be set by any user who is subscribed to the task list. Otherwise, these

properties are shared, and may only be set by users that have the mayAdmin right.

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

- \*name
- \*color
- \*sortOrder

These properties are initially inherited from the owner's copy of the task list, but if set by a sharee that user gets 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 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, or recurring series of tasks. It is a JSTask object, as defined in [[I-D.ietf-calext-jscalendar](#)], with the following additional properties:

- \***id**: Id 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 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 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  $0 \leq \text{sortOrder} < 2^{31}$ .

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.

\***source**: String|null (default: null) The source of the task like "Web app", "Mobile client" or "Email".

#### **4.1. Additional JSCalendar properties**

This document defines four new JSCalendar properties and extends one properties with new values.

##### **4.1.1. 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.

#### **4.1.2. 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.

#### **4.1.3. 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.

#### **4.1.4. relatedTo**

Type: Id[String]|null (default: null)

A map of task ids to relations. Relation SHOULD be one of:

- \*blockedBy: Blocked by task with id.
- \*clonedBy: Task with id was cloned from this issue.
- \*duplicatedBy: Task with id is a duplicate of this issue.
- \*causedBy: Task with id was the cause for this task.
- \*relatesTo: Task with id is related.
- \*childOf: Task with id is parent.

#### **4.1.5. progress**

The progress property is extended by the following values:

deferred - Indicates the task has been deferred or is waiting on someone

resolved - Indicates the task has been resolved

needs-feedback - Indicates the task is waiting on feedback

confirmed - Indicates the task has been confirmed

Mapping the status correctly to the present values of the Task can be challenging. In the most simple case tasks can have two states - done or not done. On the other hand, tasks can have various domain-specific statuses. Here are some recommendations for mapping more common values that SHOULD be used:

needs-action - not-done (most simple case), not-started, new, ... (U+2026)

in-process - in-progress, active, assigned, ... (U+2026)

completed - done (most simple case), closed, verified, ... (U+2026)

#### **4.1.6. estimatedWork**

Type: `UnsignedInt|null` (default: `null`)

This specifies the estimated amount of work the task takes to complete. The number is an abstract value without any actual unit, similar to Agile/Scrum complexity points.

#### **4.1.7. impact**

Type: `String|null` (default: `null`)

This specifies the impact or severity of the task. Some examples are: minor, trivial, major or block.

### **4.2. Participants**

The Participant object, as defined in [[I-D.ietf-calex-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

### **4.3. Properties similar in JMAP for Calendar**

Attachments, per-user properties, recurrences and updates to recurrences are described in [[I-D.ietf-jmap-calendars](#)], Section XXX.

### **4.4. 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.

#### 4.5. Task/changes

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

#### 4.6. 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.

#### 4.7. Task/copy

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

#### 4.8. 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.

#### 4.9. Task/queryChanges

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

### 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.

- name**: String The name of the person who made the change.
- email**: String The email of the person who made the change, or null if no email is available.
- principalId**: String|null The id of the principal corresponding to the person who made the change, if any. This will be null if the change was due to receiving an iTIP message.

\***comment**: String|null Comment sent along with the change by the user that made it. (e.g. COMMENT property in an iTIP message).

\***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.

To reduce data, if the change only affects a single instance of a recurring task, the server MAY set the task and taskPatch properties for the 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 "/changes" 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 for Calendars [[I-D.ietf-jmap-calendars](#)] apply to this specification.

## 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. JSCalendar Property Registrations

All IANA registrations for JSTask are described in JMAP for Calendars [[I-D.ietf-jmap-calendars](#)].

## 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-08, 23 February 2022, <<https://datatracker.ietf.org/doc/html/draft-ietf-jmap-calendars-08>>.

[I-D.ietf-jmap-sharing] Jenkins, N., "JMAP Sharing", Work in Progress, Internet-Draft, draft-ietf-jmap-sharing-01, 6 February 2022, <<https://datatracker.ietf.org/doc/html/draft-ietf-jmap-sharing-01>>.

[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>>.

[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>>.

## 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  
Durlacher Allee 47  
76131 Karlsruhe  
Germany

Email: [joris@audriga.com](mailto:joris@audriga.com)  
URI: <https://www.audriga.com>

Hans-Joerg (editor)  
audriga

Durlacher Allee 47  
76131 Karlsruhe  
Germany

Email: [hans-joerg@audriga.com](mailto:hans-joerg@audriga.com)

URI: <https://www.audriga.com>