Network Working Group
Internet-Draft
Updates: 8620 (if approved)
Intended status: Standards Track
Expires: 19 September 2026

B. Gondwana
Fastmail

18 March 2026

JMAP Object History
draft-gondwana-jmap-object-history-00

Abstract

The JMAP base protocol (RFC8620) provides methods for synchronizing the current state of data objects between client and server. This extension adds the ability to retrieve historical versions of objects, including objects that have been destroyed, by extending the standard Foo/get method.

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 19 September 2026.

Table of Contents

1. Introduction

JMAP ([JMAP-CORE] — JSON Meta Application Protocol) provides a standard set of methods for each data type: Foo/get, Foo/changes, Foo/set, Foo/query, and Foo/queryChanges. These methods operate on the current state of objects. The Foo/changes method tells a client which objects have changed, but does not provide the previous values.

In many applications it is useful to access historical versions of objects. For example, a user may want to see previous versions of a contact record ([JMAP-CONTACTS]), recover a deleted email ([JMAP-MAIL]), or audit changes to a shared mailbox.

This extension adds history support to the standard Foo/get method. When the urn:ietf:params:jmap:object-history capability is included in the request's using array, Foo/get gains additional parameters that allow the client to request previous versions of objects, including objects that have been destroyed.

1.1. Conventions Used in This Document

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.

2. Addition to the Capabilities Object

The capabilities object is returned as part of the JMAP Session object; see [JMAP-CORE], Section 2.

This document defines an additional capability URI.

2.1. urn:ietf:params:jmap:object-history

The capability urn:ietf:params:jmap:object-history being present in the "accountCapabilities" property of an account represents support for retrieving historical object versions in that account. Servers that include the capability in one or more "accountCapabilities" properties MUST also include the property in the "capabilities" property.

The value of this property in the JMAP session "capabilities" property MUST be 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 for that account:

* maxHistoryDuration: "UnsignedInt|null"

  The maximum number of seconds after replacement that the server retains old versions, or null if the server does not impose a time-based limit. A version whose replaced timestamp is more than this many seconds ago may have been discarded.

3. Extensions to Foo/get

When this capability is included in the request's using array, the standard Foo/get method ([JMAP-CORE], Section 5.1) accepts the following additional request arguments:

* includeReplaced: "Boolean" (default: false)

If true, the server returns previous versions of the requested objects in addition to the current version. Each previous version appears as a separate entry in the list array, with the same id as the current version. Every entry includes an objectHistory property (see below). Entries with the same id MUST be ordered by objectHistory.version (ascending). Entries for different objects MAY be interleaved.

* includeDestroyed: "Boolean" (default: false)

  If true, the server also returns objects that have been destroyed. If includeReplaced is false, only the final version before destruction is returned. If includeReplaced is true, all retained historical versions of destroyed objects are returned. For destroyed objects, every entry has a non-null objectHistory.replaced value, since every version has been superseded (the final version was replaced by destruction).

* historyAfter: "UTCDate|null" (default: null)

  If supplied, only versions with an objectHistory.replaced value strictly after this date are returned, plus the current (live) version which has a null replaced value. If null, no time-based filter is applied.

* historyLimit: "UnsignedInt|null" (default: null)

  If supplied, the maximum total number of entries to return in the list array. The server MUST return the most recent entries (those with the highest version numbers), not the oldest. If the limit is reached for any requested id, the response includes a hasMoreHistory property set to true. A client that needs to know which specific objects have more history available must fetch each id individually.

3.1. The objectHistory property

When includeReplaced or includeDestroyed is true, each object in the list array includes an additional property:

* objectHistory: "ObjectHistoryInfo"

  An ObjectHistoryInfo object with the following properties:

  - version: "UnsignedInt" A per-object monotonically increasing number that provides ordering among the versions of a single object within a response. Version numbers MAY have gaps (e.g., a server may use internal sequence numbers such as modseq

values). The current (live) version of an object always has the highest version number. Servers SHOULD return consistent version numbers across requests, but clients MUST NOT rely on this — version numbers exist primarily to provide ordering within a single response.

- replaced: "UTCDate|null" The date and time at which this version was superseded, either by a subsequent update or by destruction. If null, this is the current live version of the object. For destroyed objects, every version has a non-null replaced value — the final version's replaced is the time of destruction.

When includeReplaced and includeDestroyed are both false (the default), the objectHistory property is not included in the response, and the behaviour is identical to standard Foo/get.

The server is not required to create a new version for every individual change to an object. For example, rapid successive updates may be collapsed into a single version. The server MAY also prune old versions in any order, so there can be gaps in the version history available to the client. Clients MUST NOT assume that the version history is complete or contiguous.

If the server does not retain history for a particular data type, it MUST still return the current version with an objectHistory property when requested. In this case, the server MAY return a version of 1 for every object and a replaced of null, and ignore the includeReplaced flag (since there are no previous versions to return). The includeDestroyed flag may still be honoured if the server tracks destroyed object ids even without full history.

3.2. Additional response properties

When includeReplaced or includeDestroyed is true, the response includes:

* hasMoreHistory: "Boolean"

  If true, there are older entries available for at least one requested id. The client can make a wider request (e.g., a larger historyLimit or individual id requests) to retrieve more versions. If false, the server has returned all retained history for the requested ids. The server MAY return false even if more history exists (e.g., if it cannot efficiently determine whether older versions remain). If the server returns true, the client MUST NOT assume the history will still be available in a subsequent request, as the server may purge history at any time.

3.3. Examples

3.3.1. Contact history

Requesting the history of a contact, including all versions:

```
[["ContactCard/get", {
  "accountId": "abc",
  "ids": ["contact1"],
  "properties": ["name", "emails"],
  "includeReplaced": true,
  "historyLimit": 10
}, "0"]]
```

The response includes three versions: the original creation, a name update, and the current version with an additional email address:

```
   [["ContactCard/get", {
     "accountId": "abc",
     "state": "abc123",
     "list": [
       {
         "id": "contact1",
         "name": {"full": "Robert Smith"},
         "emails": {
           "e1": {"address": "bob@example.com"}
         },
         "objectHistory": {
           "version": 1,
           "replaced": "2026-02-20T14:15:00Z"
         }
       },
       {
         "id": "contact1",
         "name": {"full": "Bob Smith"},
         "emails": {
           "e1": {"address": "bob@example.com"}
         },
         "objectHistory": {
           "version": 2,
           "replaced": "2026-03-01T09:00:00Z"
         }
       },
       {
         "id": "contact1",
         "name": {"full": "Bob Smith"},
         "emails": {
           "e1": {"address": "bob@example.com"},
           "e2": {"address": "bob@personal.example"}
         },
         "objectHistory": {
           "version": 3,
           "replaced": null
         }
       }
     ],
     "notFound": [],
     "hasMoreHistory": false
   }, "0"]]
```

3.3.2. Recovering destroyed emails

A client can use Email/changes to discover which emails have been destroyed, then use a backreference to fetch their final state. This is useful for displaying a "recently deleted" view:

```
[
  ["Email/changes", {
    "accountId": "abc",
    "sinceState": "state42"
  }, "0"],
  ["Email/get", {
    "accountId": "abc",
    "#ids": {
      "resultOf": "0",
      "name": "Email/changes",
      "path": "/destroyed"
    },
    "properties": ["subject", "from", "receivedAt"],
    "includeDestroyed": true
  }, "1"]
]
```

The Email/get response returns the final state of each destroyed email. The objectHistory.replaced value is the time of destruction:

```
[["Email/get", {
  "accountId": "abc",
  "state": "state43",
  "list": [
    {
      "id": "email42",
      "subject": "Meeting notes",
      "from": [{"name": "Alice",
        "email": "alice@example.com"}],
      "receivedAt": "2026-03-10T09:00:00Z",
      "objectHistory": {
        "version": 1,
        "replaced": "2026-03-15T16:30:00Z"
      }
    },
    {
      "id": "email57",
      "subject": "Project update",
      "from": [{"name": "Bob",
        "email": "bob@example.com"}],
      "receivedAt": "2026-03-12T14:00:00Z",
      "objectHistory": {
        "version": 1,
        "replaced": "2026-03-15T16:31:00Z"
      }
    }
  ],
  "notFound": [],
  "hasMoreHistory": false
}, "1"]]
```

Since includeReplaced was not set, only the final version of each destroyed email is returned. The non-null replaced value indicates when each email was destroyed.

4. IANA Considerations

4.1. JMAP Capability Registration for urn:ietf:params:jmap:object-history

IANA is requested to register the "Object History" Capability as follows:

Capability Name: urn:ietf:params:jmap:object-history

Intended use: common

Change Controller: IETF

Specification document: this document

Security and privacy considerations: this document, Security Considerations

4.2. Update to the JMAP Data Types Registry

This document adds a "Supports History" column to the "JMAP Data Types" registry defined in [JMAP-CORE], Section 9.4. The value is "Yes" or "No", indicating whether the data type supports the history extensions defined in this document.

The initial values for existing registrations are:

Type Name	Supports History
Core/Echo	No
Mailbox	Yes
Thread	No
Email	Yes
SearchSnippet	No
Identity	Yes
EmailSubmission	No
VacationResponse	Yes
ContactCard	Yes
AddressBook	Yes
Principal	No
Quota	No
FileNode	Yes

Table 1

The default value for this column is "No". New registrations that do not explicitly include a value are assumed not to support history.

5. Security Considerations

All security considerations from [JMAP-CORE] apply to this document.

5.1. Storage and Resource Consumption

Retaining historical versions of objects can consume significant server storage. Servers SHOULD impose reasonable limits on history retention, either by duration (advertised via maxHistoryDuration) or by total storage. Servers MAY silently discard history entries when storage limits are reached.

5.2. Access Control

Historical versions of an object MUST respect the same access controls as the current version. A user who does not have permission to read an object MUST NOT be able to read its history. If an object's access permissions have changed over time, the server MUST only return versions for which the requesting user would have had read access.

5.3. Information Disclosure

Object history may reveal information that the user or an administrator intended to remove. For example, a contact record that was updated to remove a phone number will still have that phone number visible in its history. Servers SHOULD provide administrators with the ability to purge history for specific objects when required for privacy or compliance reasons.

6. Changes

EDITOR: please remove this section before publication.

The source of this document exists on github at: https://github.com/brong/draft-gondwana-jmap-object-history/

draft-gondwana-jmap-object-history-00

* initial proposal

7. Acknowledgements

TODO

{backmatter}

8. References

8.1. Normative References

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

[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/rfc/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/rfc/rfc8174>.

8.2. Informative References

[JMAP-CONTACTS] Jenkins, N., Ed., "JSON Meta Application Protocol (JMAP) for Contacts", RFC 9610, DOI 10.17487/RFC9610, December 2024, <https://www.rfc-editor.org/rfc/rfc9610>.

[JMAP-MAIL] Jenkins, N. and C. Newman, "The JSON Meta Application Protocol (JMAP) for Mail", RFC 8621, DOI 10.17487/RFC8621, August 2019, <https://www.rfc-editor.org/rfc/rfc8621>.

Author’s Address

Bron Gondwana
Fastmail
Email: brong@fastmailteam.com