JMAP for Migration and Data Portability

IETF 116

https://datatracker.ietf.org/doc/draft-baum-jmap-portability

Migration and Data Portability Spec Overview

Motivation:

- Move existing user data between systems over generic API
    - e.g., due to DMA Article 6
- Give API spec to legacy systems which have no appropriate API
- Combine with other solutions for migration and portability-related problems

Migration and Data Portability Spec Overview (2)

RFC 8620 observations:

+ feature-rich
+ generic
− complex
− unclear how to implement it partially

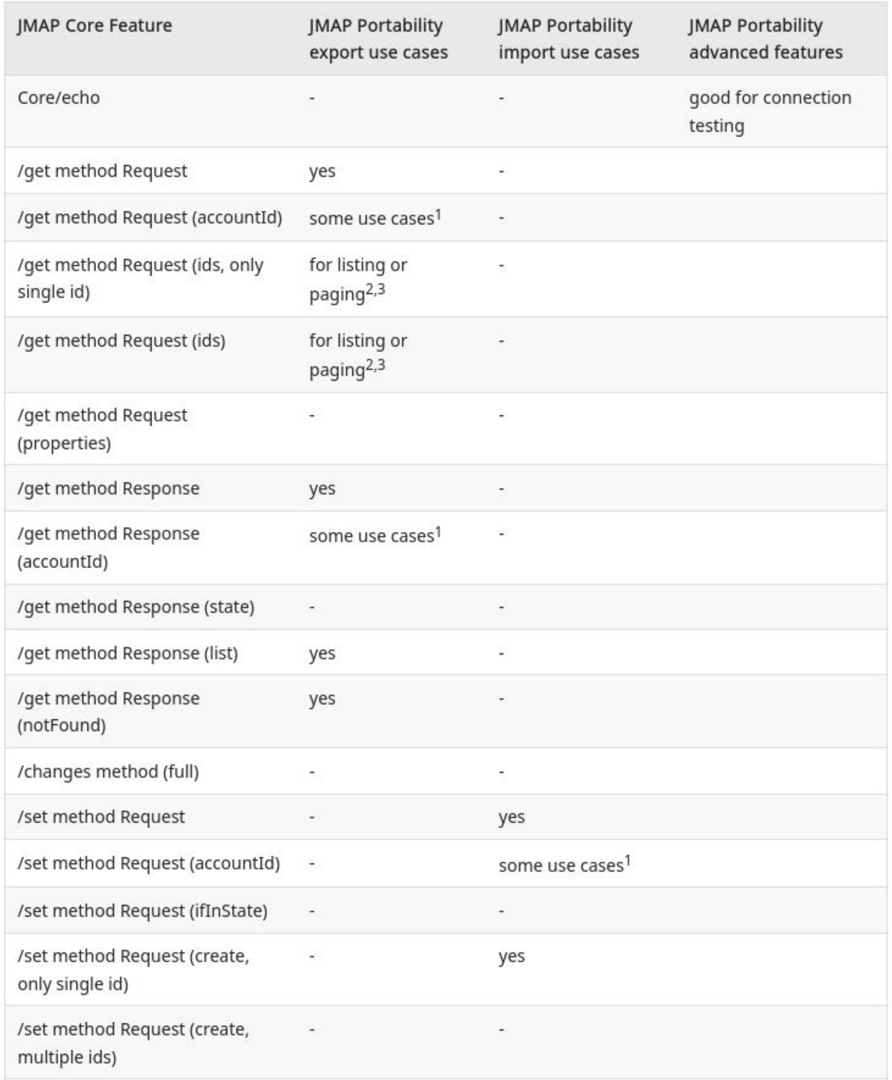
-> high entry barrier and high requirements; bad for adoption

Migration and Data Portability Spec Overview (3)

1. *“How to Quickstart JMAP”*: Guidance on bare minimum for one-time migration use for lower entry barrier
   - Session Resource with constant values for a lot of use cases
   - Focus on key objects, methods and properties for migration use-case
     - e.g., no /query for some use cases
     - no /copy or /changes methods
   - No batching, no Push, …
2. Introduce simplified request scheme
   ➜ Even lower requirements
3. Extensions for further migration-related problems?
   ➜ Improve Portability solutions even further

Focus on key objects, methods and properties

- Document how to implement RFC8620 in a minimal way
- Define additional steps necessary for common data portability use cases:
    - data export (optionally with listing/paging)
    - data import
    - attachment support
    - recommended some “advanced” features of RFC8620 (e.g., Core/echo)
- Provide developers with a simple overview what needs to be implemented for their use case
    - Overview table that could be used as a scope statement



JMAP Core Feature	JMAP Portability export use cases	JMAP Portability import use cases	JMAP Portability advanced features
Core/echo	-	-	good for connection testing
/get method Request	yes	-	
/get method Request (accountId)	some use cases[1]	-	
/get method Request (ids, only single id)	for listing or paging[2,3]	-	
/get method Request (ids)	for listing or paging[2,3]	-	
/get method Request (properties)	-	-	
/get method Response	yes	-	
/get method Response (accountId)	some use cases[1]	-	
/get method Response (state)	-	-	
/get method Response (list)	yes	-	
/get method Response (notFound)	yes	-	
/changes method (full)	-	-	
/set method Request	-	yes	
/set method Request (accountId)	-	some use cases[1]	
/set method Request (ifInState)	-	-	
/set method Request (create, only single id)	-	yes	
/set method Request (create, multiple ids)	-	-	

Issue: JMAP Portability as an alternative to RFC8620?

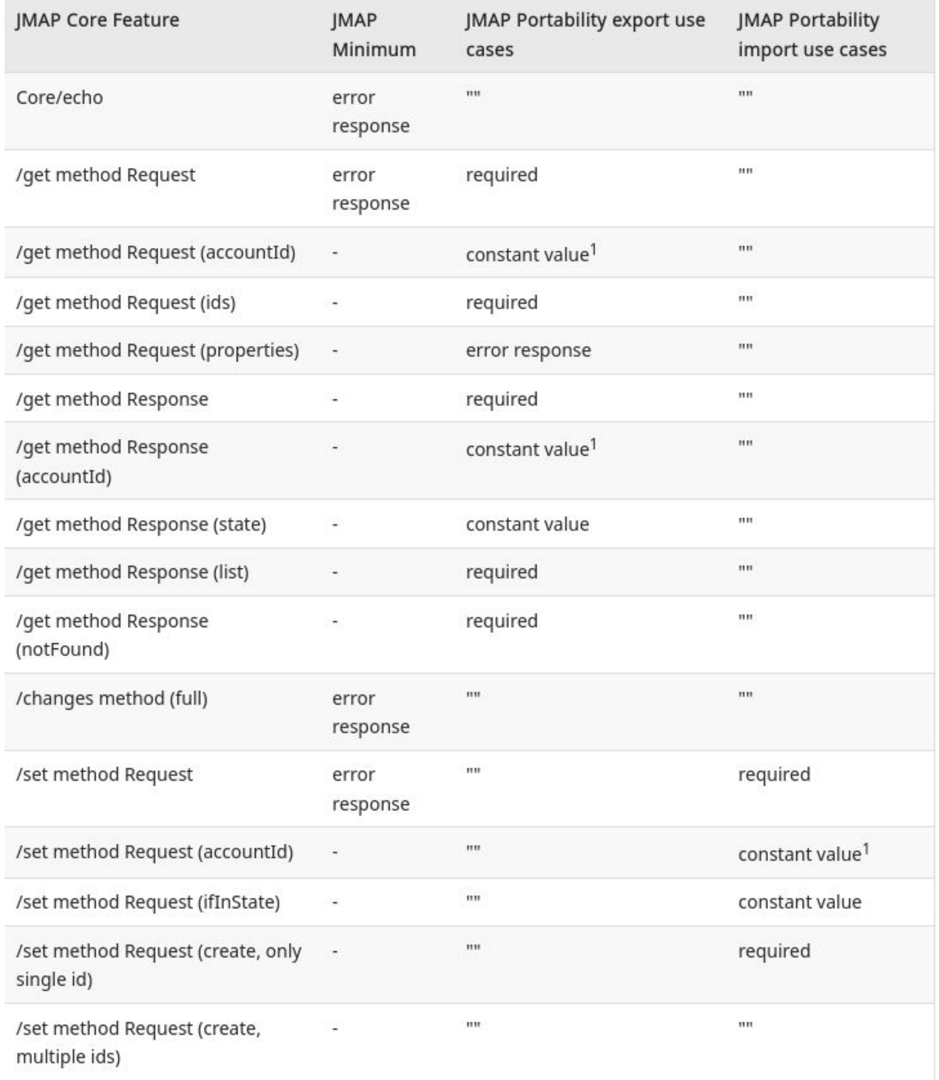
Main issue from mailing list: Merely omitting certain features of RFC8620 is forbidden.

New approach:

- Use *constant values* or *error responses* instead of simply omitting parts of RFC 8620

Examples:

- state/sessionState = “”, downloadUrl = “”, accountId = “self”
- Core/echo -> reply with serverFail error
- /get -> reply with requestTooLarge error (maxObjectsInGet was 0)
- /set -> reply with accountReadOnly error (accountReadOnly was true)



JMAP Core Feature	JMAP Minimum	JMAP Portability export use cases	JMAP Portability import use cases
Core/echo	error response	""	""
/get method Request	error response	required	""
/get method Request (accountId)	-	constant value[1]	""
/get method Request (ids)	-	required	""
/get method Request (properties)	-	error response	""
/get method Response	-	required	""
/get method Response (accountId)	-	constant value[1]	""
/get method Response (state)	-	constant value	""
/get method Response (list)	-	required	""
/get method Response (notFound)	-	required	""
/changes method (full)	error response	""	""
/set method Request	error response	""	required
/set method Request (accountId)	-	""	constant value[1]
/set method Request (ifInState)	-	""	constant value
/set method Request (create, only single id)	-	""	required
/set method Request (create, multiple ids)	-	""	""

Issue: JMAP Portability as an alternative to RFC8620? (2)

Constant values or error responses are not perfect:

- Only serverFail (“An *unexpected* or unknown error”) seems to fit for Core/echo, /query and /copy.
- Similarly, reply with “*invalid*Arguments” when certain properties are used (e.g., /query’s limit property)
- downloadUrl == “” when no attachments are supported. However, it “MUST contain variables”.

urn:ietf:params:jmap:core-essential-portability vs. *urn:ietf:params:jmap:core* :

- RFC 8620 might require some features that a lot of use cases do not. Is it flexible enough?
- Do we mind the higher complexity that comes with strictly following RFC8620?
- Discussion on the mailing list was in favour of *urn:ietf:params:jmap:core*

Session Resource

Sometimes a simple JSON with constant values is enough:

- a user login is tied to a single JMAP account
- access to shared data is not required
- capabilities, restrictions (e.g. maxMailboxesPerEmail) and URL properties (e.g., downloadUrl) are the same for every user

Then:

- accountId = “self”
- username and state are empty string

Session Resource (2)

```
"capabilities": {
   "urn:ietf:params:jmap:core": {
     "maxSizeUpload": 0,
     "maxConcurrentUpload": 0,
     "maxSizeRequest": <maxSizeRequest>,
     "maxConcurrentRequests": <maxConcurrentRequests>,
     "maxCallsInRequest": 1,
     "maxObjectsInGet": 0,
     "maxObjectsInSet": 0,
     "collationAlgorithms": []
   },
   "urn:ietf:params:jmap:<other-capability>": {},
   ...
 },
 "accounts": {
   "self": {
     "name": "",
     "isPersonal": true,
     "isReadOnly": true,
     "accountCapabilities": {
       "urn:ietf:params:jmap:<other-capability>": {
         "<key>": <value>,
         ...
       },
       ...
     }
   }
 },
```

```
"primaryAccounts": {
  "urn:ietf:params:jmap:<other-capability>": "self"
},
"username": "",
"apiUrl": "<apiUrl>",
"downloadUrl": "",
"uploadUrl": "",
"eventSourceUrl": "",
"state": ""
```

Simplified request scheme

- Request properties are inside the URI
- No need to implement processing JSON payload in Request
- WIP: Essential profile needs to mature first

```
{
  ...
  "capabilities": {
        ...,
        "urn:ietf:params:jmap:core-simple": {}
  },
  "apiUrlSimple": "https://jmap.me/api
        /?accountId=<account-id>&methodCall=<methodCall>&ids=<ids>"
}
```

Does introducing a new feature fit in the informational spec?

Extension: JMAP Debug

- Supply log messages along-side the usual data exchange instead of sending through a different channel
- Example use case: a JMAP API server running on a third-party infrastructure

```
"logs" : [
       {
       "file" : "Logger.php",
       "level" : "info",
       "line" : 32,
       "message" : "Array Logger has been successfully initialized",
       "timestamp" : "2022-01-18T10:26:56+01:00"
       },
       {
       "file" : "ErrorHandler.php",
       "level" : "warning",
       "line" : 52,
       "message" : "fopen(bridge.php):
       failed to open stream: No such file or directory",
       "timestamp" : "2022-01-18T10:26:56+01:00"
       },
       ...
],
"methodResponses" : [
       [
       "Core/echo",
       ...
```

Does it fit in the spec?

Extension: JMAP Backend Info

- Some server software does not properly follow RFC8620
- Supporting such servers requires identifying them by some means
- Typically hard-coded URI (error-prone)
- JMAP Backend Info provides clients with less error prone way

```
"capabilities": {
      "urn:ietf:params:jmap:core:backendinfo": {
        "backend": "OpenXPort/Horde v1.0.0",
        "product": "Horde Webmailer v1.0.0",
        "environment": "PHP v5.5",
        "capabilityInfo": {
          "urn:ietf:params:jmap:sieve": {
          "backend": "Cyrus timsieved",
          "product": "Horde Ingo v1.0.0",
          "fileType": "SIEVE/HORDE"
        }
      }
  },
  …
},
```

Does it fit in the spec?