Synchronization API

Reference Guide

November 2018

Contents

What is BEMS Synchronization APIs?

BEMS Synchronization APIs provide a service between BlackBerry Dynamics apps, such as BlackBerry Access, BlackBerry Work, BlackBerry Dynamics Launcher, and third-party applications and BEMS to receive notifications when specific Uniform Resource Identifiers (URIs) such as bookmarks are not synchronized between the user's devices. For example, if a user has two devices and updates a bookmark on device A, device B receives a notification indicating that the bookmark changed and the timestamp of the change. Device B then retrieves the updates to make sure that the bookmarks are synchronized across both devices for the user.

The Synchronization APIs require that each application registers for specific URIs that the application wants to receive notifications for and that the user logs in to the app a minimum of every 30 days to continue receiving notifications.

The Synchronization API consists of the following components:

- Device registration API: This API registers the URIs for notification. For example, bookmarks.
- JSONStore API: This API stores and modifies the data related to a URI.
- GEMSUpdate notification message: This message is sent from BEMS to the BlackBerry Dynamics app when URI data changed.

Device registration API

The deviceregistration API provides device registration for all notifications. A BlackBerry Dynamics app may register a device push token (for eaxample, GCM, APNS, GNP) for notification of specific URIs, such as bookmarks, to synchronize between devices. The Synchronization API requires that each BlackBerry Dynamics app registers for specific URIs that the application wants to receive notifications for and that the user logs in to the app a minimum of every 30 days to continue receiving notifications. If a user does not log in every 30 days, BEMS removes the push channel device registration for the user's devices.

You must specify the BEMS endpoint in the API. The endpoint specifies where the object address is located.

```
Endpoint: <BEMS>/api/deviceregistration
```

Where *BEMS* is the FQDN of BEMS.

The format for the HTTP Request to register for device notifications in BEMS is:

```
POST <BEMS>/api/deviceregistration
<request-header>
<request-body>
```

The following is a sample request header:

```
Content-Type: application/json
X-Good-GD-AuthToken: <GDAuthToken>
```

Device registration properties

The following table describes the request body properties you can include in the deviceregistration API when you create the push channel device registration.

Property	Type	Value	Description
registrationId	String		Specifies the unique ID for the client application. **Note:** BEMS doesn't validate the registrationID.
account	String		Specifies the user's email address.
pushToken	String		Specifies the APNS or GNP token for push destination.
bundleId	String		Specifies the mobile app ID.
deviceType	String	• ios • android	Specifies the device type.
settings	Subtype	• "notificationNetwork":"android_gcm"	Specifies the optional settings for Android Firebase Cloud Messaging (FCM)* only.

Property	Type	Value	Description
gnpToken	String		Specifies the GNP token.
clientType	String	• BlackBerry Access • BlackBerry Work • BlackBerry Dynamics Launcher • ISV	Specifies the the application type.
URI	String[]	• bookmarks	Specifies the list of URI information that the application requires change notifications for.

* Google now uses the new service Firebase, replacing the Google Cloud Messaging (GCM) API site and project spaces.

Code example for device registration request

The following code example sends a device registration request to BEMS:

```
POST https://<BEMS FQDN>:8443/api/deviceregistration
Content-Type:application/json
X-Good-GD-AuthToken: "<AuthToken>"
{
"account": <user1>@<FQDN of BEMS instance> ,
"deviceType": "android",
"registrationId": "<registrationId>",
"bundleId": "com.good.goodaccess.enterprise",
"pushToken": "<pushToken>",
"clientType":"BlackBerry  Access",
"gnpToken":<gnpToken>",
"URI”:[”bookmarks”],
"settings":{ "notificationNetwork":"android_gcm"}
}
```

If the request is successful, BEMS returns the following response:

```
HTTP/1.1 200 OK
```

JSONStore API

BEMS supports generic data storage in JSON format of client application data for a URI. This section describes the API to create, update, fetch, and delete these client data. The following subsections uses the "bookmarks" client data as an example of this API usage.

You must specify the BEMS endpoint in the API. The endpoint specifies where the object address is located.

```
Endpoint: <BEMS>/<serviceName>
```

- Where <*BEMS*> is the FQDN of BEMS.
- Where <*serviceName*> is the name of the service or URI to synchronize.

Request format

The following format is for the HTTP Request to create, update, fetch, and delete data items in BEMS:

```
<HTTP Method> jsonstore/<serviceName>/<option>
<request-header>
<request-body>
```

- Where <*HTTP Method*> is POST, GET, or DELETE.
- Where <*serviceName*> is the name of the URI. For example, bookmarks.
- Where <*option*> is one of the following actions:
  - createupdate: Use with POST to create or update one or more records.
  - read/<*id*>: Use with GET to fetch the record with the specified Id.
  - fetch: Use with POST to fetch one or multiple records.
  - delete/<*id*>: Use with DELETE to delete the record with the specified Id.
  - delete: Use with POST to delete multiple records.

The following is a sample request header:

```
Content-Type: application/json
X-Good-GD-AuthToken: <GDAuthToken>
X-Good-GEMS-Scope:<scope>
X-Good-GEMS-RegistrationId: <RegistrationId>
```

- Where <*scope*> specifies where an object can be stored and retrieved from. The scope is included in every request header. You can specify one for the following scopes:
  - User: The User scope allows for the object to be accessed, modified, and retrieved by any application on any device that is assigned to the user.
  - Application: The application scope allows for the object to be accessed, modified, and retrieved only by the specified application on any device assigned to the user.
  - Container: The container scope allows for the object to be accessed, modified, and retrieved only by the specified container for the user.
- Where <*RegistrationId*> specifies the unique ID for the client application in the request header. The registration ID is used as the originator in the GEMSUpdate message to other client applications. RegistrationId is only used when you create, update, and delete bookmarks.

HTTP response codes

Code	Description
200	Records are fetched, updated and deleted successfully.
201	At least one record was created successfully.
400	The request is bad.
404	None of the specified records are found.
406	The create, update or delete is missing the X-Good-GEMS-RegistrationID.

Creating and updating bookmarks

Create and update bookmark properties

The following table describes the request properties you can include in the JSONStore API when you create and update bookmarks.

Property	Type	Description
id	String	Specifies the unique ID of bookmark.
payload	Subtype	Specifies the JSON data to be stored or retrieved from the database.
lastModifiedTime	Timestamp	If creating a new bookmark, set lastModifiedTime to 0. If updating a bookmark, set last modified time to lastModifiedTime returned from a fetch or create of the bookmark.

Create and updated multiple bookmarks

You can store and update multiple bookmarks and view BEMS response to identify those bookmarks that failed to store or encountered conflicts and those bookmarks that successfully stored and updated. When a record, for this example bookmark, is updated, the request includes the bookmark id and timestamp that was changed. If the bookmark is a new bookmark, the request includes a timestamp of 0 (zero).

The following is an example of storing and updating multiple bookmarks.

```
POST jsonstore/bookmarks/createupdate
Content-Type: application/json
X-Good-GD-AuthToken: "<AuthToken>"
X-Good-GEMS-Scope:USER
X-Good-GEMS-RegistrationId: <RegistrationId>

[
  {
    "id": "204996f57b552ff58e43cebe03f6f58de",
    "payload": {
      "title": "example 1",
      "body": "http://example1.com"
```

```
    },
    "lastModifiedTime": 1484251451970
  },
  {
    "id": "3fba55721a8ff31c367baac5df03cbc38",
    "payload": {
      "title": "Example 2",
      "body": "http://example2.com"
    },
    "lastModifiedTime": 1484251451970
  },
  {
    "id": "39bc1dd4dd90aad1f4910d3682f349b07",
    "payload": {
     "title": "Example 3",
     "body": "http://example3.com"
    },
   "lastModifiedTime": 0
  },
  {
    "id": "3fbe1884dd9099d1f4810d47f3c495c18",
    "payload": {
      "title": "Example 4",
      "body": "http://example4.com",
      "newfield": "testing"
    },
   "lastModifiedTime": 0
}
]
```

BEMS returns response codes indicating whether bookmarks are newly created and updated successfully:

```
[
  {
    "id": "204996f57b552ff58e43cebe03f6f58de",
    "error": "ALREADY_EXISTS"
  },
  {
    "id": "3fba55721a8ff31c367baac5df03cbc38",
    "error": "NOT_FOUND"
  },
  {
    "id": "39bc1dd4dd90aad1f4910d3682f349b07",
    "lastModifiedTime": 1484673690049
  },
  {
    "id": "3fbe1884dd9099d1f4810d47f3c495c18",
    "lastModifiedTime": 1484673690049
  }
]
```

BEMS response codes

The following table describes the response codes that BEMS might return. These sample response codes are based on the sample code in Create and updated multiple bookmarks.

ID	Response	Description
id:204996f57b552ff58e43cebe03f6f58de	Error: ALREADY EXISTS	This bookmark already exists in the database. The lastModifiedTime in the request is not equal to the lastModifiedTime in the database for the ID resulting in a conflict that must be resolved by the application.
id:3fba55721a8ff31c367baac5df03cbc3	Error:NOT_FOUND	The bookmark is not found in the database. It might have been deleted on another of the user's devices.
id:39bc1dd4dd90aad1f4910d3682f349b07	"lastModifiedTime": 1484673690049	The lastModifiedTime of 0 in the request indicates that the bookmark is newly created on the device, BEMS updates the database and returns a timestamp in the response.
3fbe1884dd9099d1f4810d47f3c495c18	"lastModifiedTime": 1484673690049	The lastModifiedTime of 0 in the request indicates that the bookmark is newly created on the device and BEMS returns a timestamp. This request contains a new payload field called newfield. Payloads can change. BEMS updates the database with the new field.

Fetching bookmarks

The application can retrieve one or multiple bookmarks.

Fetch a bookmark

The following is sample code for retrieving a single bookmark. In this example, the application is retrieving the bookmark for ID 204996f57b552ff58e43cebe03f6f58de, Example 1 from Create and updated multiple bookmarks.

```
GET jsonstore/bookmarks/read/"204996f57b552ff58e43cebe03f6f58de"
Content-Type: application/json
X-Good-GD-AuthToken: "<Authtoken>"
X-Good-GEMS-Scope:USER
```

BEMS returns a single bookmark that matches the ID specified.

```
{
  "id": "204996f57b552ff58e43cebe03f6f58de",
  "lastModifiedTime": 1484673689436,
  "payload": "{\"title\":\"example 1\",\"body\":\"http://news.example1.com\"}"
}
```

Fetching multiple bookmark request properties

The following table describes the request properties you can include in the JSONStore API when you fetch multiple bookmarks.

Property	Type	Description
idOnly	Boolean	Specifies only to include the bookmark record id in the response.
lastModifiedTime	Timestamp	If fetching all bookmarks, set the lastModifiedTime to 0. Otherwise, set the lastModifiedTime to the preferred time.
maxRecords	Number	Specifes the maximum number of records BEMS returns in a response batch.
offset	Number	Specifies the starting point of a batch response.

Fetching multiple bookmark response properties

The following table describes the response properties that might appear in the JSONStore API response when you fetch multiple bookmarks.

Property	Type	Description
Offset	Number	Specifies the starting point of a batch response.
TotalCount	Number	Specifies the total number of bookmarks that match the fetch query.
MoreAvailable	Boolean	Indicates that more bookmarks are available than the fetch response returned.
NextPageOffset	Number	Specifies the starting point of the second batch of bookmarks that are returned.
Size	Number	Specifies the number of bookmarks returned in the response, up to the maxRecords size specified.
bookmarks	Subtype[]	Specifies the array of bookmark data returned from the fetch query. Data may include only the "id" if "idOnly" is "true" in the request. If "idOnly" is "false", data may include "id", "lastModifiedTime", and "payload".
id	String	Specifies the unique ID of bookmark.
lastModifiedTime	Timestamp	The lastModifiedTime returned from a fetch of the bookmark.
payload	Subtype	Specifies the JSON data retrieved from the database.

Fetch batches of bookmarks

You can retrieve batches of bookmarks for a user for maxRecords. In the following sample code, BEMS retrieves all records, excluding deleted records, since the last modified time. The response includes up to the 100 records, as specified by maxRecords. If there are more than 100 records, the response includes a MoreAvailable=true. The

client application then sends additional requests using the NextOffset value to retrieve records in batches until MoreAvailable=false.

```
POST jsonstore/bookmarks/fetch
Content-Type: application/json
X-Good-GD-AuthToken: "<authtoken>"
X-Good-GEMS-Scope:USER

{
  "idOnly": "true",
  "lastModifiedTime": 1484593063122,
  "maxRecords" : 100,
  "offset" : 0
}
```

BEMS returns the following batches:

Values	Sample code

MoreAvailable=true	Call to BEMS to fetch records.
	`POST jsonstore/bookmarks/fetch`
	`Content-Type: application/json`
	`X-Good-GD-AuthToken: "<authtoken>"`
	`X-Good-GEMS-Scope:USER`
	`{`
	`  "idOnly": "true",`
	`  "lastModifiedTime": 1484593063122,`
	`  "maxRecords" : 100,`
	`  "offset" : 0`
	`}`
	BEMS response returns first batch of 100 records.
	`{`
	`  "Offset": 0,`
	`  "TotalCount": 110,`
	`  "MoreAvailable": true,`
	`  "NextPageOffset": 100,`
	`  "Size": 100,`
	`  "bookmarks":[`
	`  {`
	`  "id": "204996f57b552ff58e43cebe03f6f58de"`
	`  },`
	`  {`
	`  "id": "39cd2ee5ee01bbe2f4021e4793f350c18"`
	`  },`
	`.`
	`.`
	`.`
	`.`
	`  {`
	`  "id": "3fba55721a8ff31c367baac5df03cbc38"`
	`  }`
	`  ]`
	`}`

Values	Sample code

MoreAvailable=false	Call to BEMS to fetch additional records.
	`POST jsonstore/bookmarks/fetch`
	`Content-Type: application/json`
	`X-Good-GD-AuthToken: "<authtoken>"`
	`X-Good-GEMS-Scope:USER`
	`{`
	`"idOnly": "true",`
	`"lastModifiedTime": 1484593063122,`
	`"maxRecords" : 100,`
	`"offset" : 100`
	`}`
	BEMS response returns second batch of 10 records.
	`{`
	`"Offset": 100,`
	`"TotalCount": 110,`
	`"MoreAvailable": false,`
	`"NextPageOffset": null,`
	`"Size": 10,`
	`"bookmarks":[`
	`{`
	`"id": "204996f57b552ff58e43cebe03f6f58de"`
	`},`
	`{`
	`"id": "39cd2ee5ee01bbe2f4021e4793f350c18"`
	`},`
	`.`
	`.`
	`.`
	`.`
	`{`
	`"id": "3fba55721a8ff31c367baac5df03cbc38"`
	`}`
	`]`
	`}`

Deleting bookmarks

Delete a single bookmark record

The application sends a Delete request to BEMS to delete the record that is specified by the ID. This marks the bookmark to be deleted and clears the payload data from the database. Deleted records are permanently deleted after 30 days.

```
DELETE jsonstore/bookmarks/delete/<record id>
X-Good-GD-AuthToken: "<authtoken>""
X-Good-GEMS-Scope:USER
X-Good-GEMS-RegistrationId: <RegistrationId>
```

Delete multiple bookmark records

The following sample code deletes multiple records for a user. If an item exists, it is marked for deletion and timestamped for the deletion in the response from BEMS. If the item is already deleted or does not exist, the response returns a NOT_FOUND error. Deleted items are synchronized to the user's other devices. The following sample code deletes the Example 1 and Example 3 bookmarks.

```
POST jsonstore/bookmarks/delete
Content-Type: application/json
X-Good-GD-AuthToken: "<Authtoken>"
X-Good-GEMS-Scope:USER
X-Good-GEMS-RegistrationId: <RegistrationId>

 {
  "id": "204996f57b552ff58e43cebe03f6f58de"
 },
 {
  "id": "39bc1dd4dd90aad1f4910d3682f349b07"
 }
]
```

BEMS returns the following response:

```
[
  {
    "id": "204996f57b552ff58e43cebe03f6f58de",
    "lastModifiedTime": 1486674836738
  },
  {
    "id": "39bc1dd4dd90aad1f4910d3682f349b07",
    "error": "NOT_FOUND"
  }
]
```

For more information about response codes, see BEMS response codes

GEMSUpdate notification

After BEMS processes a JSONStore API to create, update, or delete records, it sends a GEMSUpdate message to each BlackBerry Dynamics app registered for the URI item that did not send the original synchronization message. If multiple records are updated with the jsonstore API, only one GEMSUpdate message is sent. The BlackBerry Dynamics app then calls the JSONStore API to fetch the data using the "updated" time as the lastModifiedTime to synchronize the URI record (for example, bookmarks). The GEMSUpdate message includes the following properties:

Properties	Description
server	The server property identifies the BEMS instance that is sending the notification.
updated	The updated property specifies the server relative time that the update occurred. For example UTC ms since 1970/01/01.
from	The from property specifies the user's device which initiated the update. This value is based on the registrationID provided in the header of the JSONStore API to create, update, or delete a URI related item. For example, if the registrationId is "<*user1*>@USER1androidc123456789", the update is specified as originating from USER1androidc123456789.
item	The item property specifies the affected URI. In this example, the updated URI is bookmarks.

The following is a code example of the GEMSUpdate message received by the user's device:

```
{
   "GEMSUpdate" :
   {
      "server" : "<The computer hosting  BEMS>.<domain>:8181",
      "updated" : 98499696242,
      "from": "USER1androidc123456789",
      "item" : "bookmarks"
   }
}
```

Legal notice

WERE FORESEEN OR UNFORESEEN, AND EVEN IF BLACKBERRY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW IN YOUR JURISDICTION, BLACKBERRY SHALL HAVE NO OTHER OBLIGATION, DUTY, OR LIABILITY WHATSOEVER IN CONTRACT, TORT, OR OTHERWISE TO YOU INCLUDING ANY LIABILITY FOR NEGLIGENCE OR STRICT LIABILITY.

THE LIMITATIONS, EXCLUSIONS, AND DISCLAIMERS HEREIN SHALL APPLY: (A) IRRESPECTIVE OF THE NATURE OF THE CAUSE OF ACTION, DEMAND, OR ACTION BY YOU INCLUDING BUT NOT LIMITED TO BREACH OF CONTRACT, NEGLIGENCE, TORT, STRICT LIABILITY OR ANY OTHER LEGAL THEORY AND SHALL SURVIVE A FUNDAMENTAL BREACH OR BREACHES OR THE FAILURE OF THE ESSENTIAL PURPOSE OF THIS AGREEMENT OR OF ANY REMEDY CONTAINED HEREIN; AND (B) TO BLACKBERRY AND ITS AFFILIATED COMPANIES, THEIR SUCCESSORS, ASSIGNS, AGENTS, SUPPLIERS (INCLUDING AIRTIME SERVICE PROVIDERS), AUTHORIZED BLACKBERRY DISTRIBUTORS (ALSO INCLUDING AIRTIME SERVICE PROVIDERS) AND THEIR RESPECTIVE DIRECTORS, EMPLOYEES, AND INDEPENDENT CONTRACTORS.

IN ADDITION TO THE LIMITATIONS AND EXCLUSIONS SET OUT ABOVE, IN NO EVENT SHALL ANY DIRECTOR, EMPLOYEE, AGENT, DISTRIBUTOR, SUPPLIER, INDEPENDENT CONTRACTOR OF BLACKBERRY OR ANY AFFILIATES OF BLACKBERRY HAVE ANY LIABILITY ARISING FROM OR RELATED TO THE DOCUMENTATION.

Prior to subscribing for, installing, or using any Third Party Products and Services, it is your responsibility to ensure that your airtime service provider has agreed to support all of their features. Some airtime service providers might not offer Internet browsing functionality with a subscription to the BlackBerry® Internet Service. Check with your service provider for availability, roaming arrangements, service plans and features. Installation or use of Third Party Products and Services with BlackBerry's products and services may require one or more patent, trademark, copyright, or other licenses in order to avoid infringement or violation of third party rights. You are solely responsible for determining whether to use Third Party Products and Services and if any third party licenses are required to do so. If required you are responsible for acquiring them. You should not install or use Third Party Products and Services until all necessary licenses have been acquired. Any Third Party Products and Services that are provided with BlackBerry's products and services are provided as a convenience to you and are provided "AS IS" with no express or implied conditions, endorsements, guarantees, representations, or warranties of any kind by BlackBerry and BlackBerry assumes no liability whatsoever, in relation thereto. Your use of Third Party Products and Services shall be governed by and subject to you agreeing to the terms of separate licenses and other agreements applicable thereto with third parties, except to the extent expressly covered by a license or other agreement with BlackBerry.

The terms of use of any BlackBerry product or service are set out in a separate license or other agreement with BlackBerry applicable thereto. NOTHING IN THIS DOCUMENTATION IS INTENDED TO SUPERSEDE ANY EXPRESS WRITTEN AGREEMENTS OR WARRANTIES PROVIDED BY BLACKBERRY FOR PORTIONS OF ANY BLACKBERRY PRODUCT OR SERVICE OTHER THAN THIS DOCUMENTATION.

BlackBerry Enterprise Software incorporates certain third-party software. The license and copyright information associated with this software is available at http://worldwide.blackberry.com/legal/thirdpartysoftware.jsp.

BlackBerry Limited
2200 University Avenue East
Waterloo, Ontario
Canada N2K 0A7

BlackBerry UK Limited
Ground Floor, The Pearce Building, West Street,
Maidenhead, Berkshire SL6 1RL
United Kingdom

Published in Canada