18.10. mSupply sync API

What is it?

There are multiple ways to configure an mSupply installation. Where the internet is good, or you are using a local area network, you can have real-time communication with an mSupply server. Where internet access is poor or non-existent, we have mSupply sync. You can have a local application that saves all the changes the user makes to a local database - this can be a mobile app, a single user application on a PC or a tablet, or a client-server application on a local LAN.

The machine on which the data is stored, which we call a *satellite*, then periodically communicates with a central mSupply server, which we call the *primary* - this may be in a central location in the country, or it may be a cloud-based server. The REST API described here is used to send and receive data from the primary server to keep both satellite and primary sites up to date with data additions, modifications and deletions. See Remote Synchronisation for more details on how sync works.

Basics

You might want to read about our mSupply Mobile API, as the basics are the same. That is:

1. The SYNC API is a subset of the mSupply REST API.
2. All calls to the sync API use a pseudo-resource of “sync”.
3. The second URI segment is the API version. Allowable values:

Value	Release Date
v1	25 Jan 2016
v2	01 Mar 2017

Our Android mobile app uses v2 exclusively to communicate with the primary mSupply server, whereas mSupply itself uses v1 for synchronisation between different instances of mSupply. For example, you call “https://example.com/sync/v1/somethinghere”.

Server requirements

You will need to be querying an mSupply standalone or server that:

1. Has a fixed IP address.
2. Ideally has a domain name and a valid SSL certificate - we rely on SSL for stopping your communications and password(s) being acquired. You should use it.
3. Is reachable from the machine you are using these APIs on.

Authentication

We use basic HTTP authentication:

1. v1 of the API only supported a simple *username:password*, with each satellite sharing the same credentials for a single special sync user on the primary server.
2. v2 expects the password part to be SHA256 encoded i.e. *sitename:SHA256_encoded_password*, and each satellite can have its own individual site name and password credentials on the primary server.
3. v1 now also supports SHA256-encoded passwords and individual satellite site credentials.

Available calls (v1)

Note that each site has a unique integer ID, and that the ID for the primary server (running a web server) is always set to 1.

HTTP GET

```
https://example.com/sync/v1/queued_records/count?from_site=4&to_site=1
```

will retrieve the number of sync records in the outgoing queue for the sync satellite site with ID=4 (the requesting site) on the sync site with ID=1.

```
https://example.com/sync/v1/queued_records/?from_site=4&to_site=1&limit=10
```

will retrieve the first 10 sync records in the outgoing queue for the sync satellite site with ID=4 (the requesting site) on the sync site with ID=1. If *limit* is missing, it will only retrieve a single record from the head of the queue.

```
https://example.com/sync/v1/queued_records/?from_site=4&to_site=1&sync_id=GS
HJDJKSKS
```

will retrieve a single sync record from the outgoing queue for the sync satellite site with ID=4 (the requesting site) on the sync site with ID=1, where the sync record ID=GSHJDJKSKS (UUID).

```
https://example.com/sync/v1/initial_dump/?from_site=4&to_site=1&type=full
```

will request the sync site with ID=1 to regenerate all sync records for the sync satellite site with ID=4 (the requesting site). This is used to (re)initialise a new satellite. If *type* is missing, or has any other value, then only the subset of sync records necessary for a mobile store will be generated.

HTTP POST

```
https://example.com/sync/v1/acknowledged_records/?from_site=4&to_site=1
```

will inform the server that the sync records whose IDs (UUID) are specified (as JSON) in the HTTP BODY have been consumed from the outgoing queue for the sync satellite site with ID=4 (the requesting site) on the sync site with ID=1.

```
https://example.com/sync/v1/queued_records/?from_site=4&to_site=1
```

will apply the sync record data specified (as JSON) in the HTTP BODY from the sync satellite site with ID=4 (the requesting site) to the sync site with ID=1. Multiple sync records can be sent in one packet.

Format of a record

- The format is JSON.
- If there is more than one record returned or to be sent, each record is an element of a JSON array.
- v1 uses internal field numbers and table numbers.
- Text ID fields are always UUIDs.
- Blobs, pictures etc. are BASE64 encoded.
- Example for v1:

```
[
  {
    "SyncID": "asdlkfj",
    "TableNumber": 3,
    "RecordID": "dkfaadj",
    "KeyFieldID": 5,
    "SyncType": "U",
    "StoreID": "knvslkj",
    "fields": [9, 11, 3],
    "values": ["sadflkj", "Trinity", "12"]
  }
]
```

Available calls (v2)

The v1 API calls above behave in exactly the same way for v2, except that the format of any sync records transferred is different. The extra v2-specific API calls below are used to retrieve the necessary parameters for our mobile app, which is designed to host a single store.

HTTP GET

```
https://example.com/sync/v2/queued_records/count?from_site=4&to_site=1
```

will retrieve the number of sync records in the outgoing queue for the sync satellite site with ID=4 (the requesting site) on the sync site with ID=1.

```
https://example.com/sync/v2/queued_records/?from_site=4&to_site=1&limit=10
```

will retrieve the first 10 sync records in the outgoing queue for the sync satellite site with ID=4 (the requesting site) on the sync site with ID=1. If *limit* is missing, it will only retrieve a single record from the head of the queue.

```
https://example.com/sync/v2/queued_records/?from_site=4&to_site=1&sync_id=GSHJDJKSKS
```

will retrieve a single sync record from the outgoing queue for the sync satellite site with ID=4 (the requesting site) on the sync site with ID=1, where the sync record ID=GSHJDJKSKS (UUID).

```
https://example.com/sync/v2/initial_dump/?from_site=4&to_site=1&type=full
```

will request the sync site with ID=1 to regenerate all sync records for the sync satellite site with ID=4 (the requesting site). This is used to (re)initialise a new satellite. If *type* is missing, or has any other value, then only the subset of sync records necessary for a mobile store will be generated.

```
https://example.com/sync/v2/site/?site_name=Satellite4
```

will retrieve the site ID (integer) and the corresponding store ID (UUID) and store-name ID (UUID) for the sync satellite site with name=Satellite4 on the primary sync site.

```
https://example.com/sync/v2/user/?store=CVCBNXNSHHSH
```

will retrieve the user ID (UUID) for the sync satellite site having store ID=CVCBNXNSHHSH (UUID, returned by the above REST call) on the primary sync site.

HTTP POST

```
https://example.com/sync/v2/acknowledged_records/?from_site=4&to_site=1
```

will inform the server that the sync records whose IDs (UUID) are specified (as JSON) in the HTTP BODY have been consumed from the outgoing queue for the sync satellite site with ID=4 (the requesting site) on the sync site with ID=1.

```
https://example.com/sync/v2/queued_records/?from_site=4&to_site=1
```

will apply the sync record data specified (as JSON) in the HTTP BODY from the sync satellite site with ID=4 (the requesting site) to the sync site with ID=1. Multiple sync records can be sent in one packet.

Format of a record

- The format is JSON.
- If there is more than one record returned or to be sent, each record is an element of a JSON array.
- v2 uses field names and table names as described in the Field descriptions chapter.
- Text ID fields are always UUIDs.
- Blobs, pictures etc. are BASE64 encoded.
- Example for v2:

```
[
  {
    "SyncID": "kjhkljg",
    "RecordType": "trans_line",
    "RecordID": "dsfhjd",
    "SyncType": "I",
    "StoreID": "klsvnsl",
    "Data": {"item_name": "hello", "quantity": "95555", "cost_price":
"65.5"}
  }
]
```

Available calls (v4)

HTTP POST

```
https://example.com/sync/v4/name_store_join
```

will tell the server to create a name_store_join record for the stoere and name records defined in the body.

Body

```
{
    "name_ID": "xxx",
    "store_ID": "xxx"
}
```

Response

If all is well, 200 OK and the payload:

```
{
  "name_ID": "6F0EA445B35F479AA1020BD2D24B9755",
  "store_ID": "8D967C2618BE4D78B3A6FAD6C1C8FF25",
  "inactive": false,
  "ID": "C6C0A32C1E9841FF93735144F0114290",
}
```

A 400 will be returned if either of the payload elements are missing or the IDs are not present.

A 401 will be returned if authentication fails.

*Previous: **19.10. mSupply legacy REST APIs*** | | *Next: **22. mSupply Mobile (Android)***