Blyott | REMOTE INSIGHTS

Blyott Platform

(API Overview Documentation)

Document Version 8.1 (March 2021)

Contents

AUTHENTICATION AND AUTHORIZATION

Authentication is performed via Login API call (see Login details) through which access credentials (provided by the Blyott) are exchanged for an Access Token and Refresh Token. Provided Access Token needs to be present in all subsequent HTTP API calls as a HTTP header named token. Access Token obtained is a JWT token which when decoded has a role property indicating either User or Administrator role alongside its expiration time.

Login (POST /login)

Credentials are exchanged for an Access Token by doing a Post request against /login endpoint. Response will contain the AccessToken and RefreshToken. AccessToken should then be provided in all subsequent API calls as a HTTP header "token". Further curl examples indicate the presence of a token with {token}.

Curl Example:

```
curl 'https://api.blyott.com/login' |
-H 'authority: api.blyott.com' |
-H 'accept: application/json, text/plain, */*' |
-H 'content-type: application/json' |
-H 'origin: https://portal.blyott.com' |
-H 'sec-fetch-site: same-site' |
-H 'sec-fetch-mode: cors' |
-H 'sec-fetch-dest: empty' |
-H 'referer: https://portal.blyott.com/login' |
-H 'accept-language: en-US,en;q=0.9' |
--data-binary '{"username":"username","password":"password"}' |
```

Refresh token (POST /getCredentialsWithRefreshToken)

Refresh token is used to get a new access token.

Curl Example:

```
curl 'https://api.blyott.com/getCredentialsWithRefreshToken' |
  -H 'authority: api.blyott.com' |
  -H 'accept: application/json, text/plain, */*' |
  -H 'token: {token}' |
  -H 'content-type: application/json' |
  -H 'origin: https://portal.blyott.com' |
  -H 'sec-fetch-site: same-site' |
  -H 'sec-fetch-mode: cors' |
  -H 'sec-fetch-dest: empty' |
  -H 'referer: https://portal.blyott.com/' |
  -H 'accept-language: en-US,en;q=0.9' |
  --data-binary '{"username":"","refreshToken":"{refreshToken}"}' |
```

Forgot Password (POST user/{userName}/forgotPassword)

If the user exists, reset password code with a link will be sent to the user's email.

Curl Example:

```
curl 'https://api.blyott.com/user/test/forgotPassword' |
  -H 'authority: api.blyott.com' |
  -H 'accept: application/json, text/plain, */*' |
  -H 'content-type: application/json' |
  -H 'origin: https://portal.blyott.com' |
  -H 'sec-fetch-site: same-site' |
  -H 'sec-fetch-mode: cors' |
  -H 'sec-fetch-dest: empty' |
  -H 'referer: https://portal.blyott.com/' |
  -H 'accept-language: en-US,en;q=0.9' |
  --data-binary '{}' |
```

Confirm Forgot Password (POST /user/confirmForgotPassword)

When the user requests the password reset, a new password needs to be confirmed with the confirmation code which is sent to the user's email. Confirmation is done by issuing a POST request to the platform against the */user/confirmForgotPassword* endpoint with of JSON payload consisting of following parameters:

1. Username - username of user which requests new password.
2. NewPassword - new password.
3. Code - code received by the email after user called Forgot Password.

Curl Example:

```
curl 'https://api.blyott.com/user/confirmForgotPassword' \
  -H 'authority: api.blyott.com' \
  -H 'accept: application/json, text/plain, */*' \
  -H 'content-type: application/json' \
  -H 'origin: https://portal.blyott.com' \
  -H 'sec-fetch-site: same-site' \
  -H 'sec-fetch-mode: cors' \
  -H 'sec-fetch-dest: empty' \
  -H 'referer: https://portal.blyott.com/' \
  -H 'accept-language: en-US,en;q=0.9' \
  --data-binary '{"Username":"test","NewPassword":"newPassword","Code":"111111"}' \
```

TAG SPECIFIC OPERATIONS:

Create Tag (POST /tag)

Tag is created by issuing a POST request to the platform against the */tag* endpoint with a body of JSON payload consisting of at least the following parameters:

1. TagId - BLE MAC address of the Tag
2. NFCId - Tag's NFC identifier, usually the same as BLE MAC
3. TagType:
    a. 1 - Mobile Tags assigned to assets expected to move.
    b. 2 - Fixed Tags assigned to assets not expected to move.
    c. 3 - Machine Learning tag
4. FixedLocation - only applicable if TagType is 2, then the value should be the set to the internal Id of the Location (see Locations) otherwise null.
5. TagHardwareId - Tag's Manufacturer and model information, should be set to Id of the HardwareModel (see Hardware)
6. AssetId - Asset to which the Tag is linked to, can be null if the Tag is unassigned. Id should be of an unassigned Asset (see Unassigned Assets)
7. Dynamic properties - any custom dynamic property added through Layout Builder.

Curl example:

```
curl 'https://api.blyott.com/tag' \
  -H 'authority: api.blyott.com' \
  -H 'accept: application/json, text/plain, */*' \
  -H 'token: {token}' \
  -H 'content-type: application/json' \
  -H 'origin: https://portal.blyott.com' \
  -H 'sec-fetch-site: same-site' \
  -H 'sec-fetch-mode: cors' \
  -H 'sec-fetch-dest: empty' \
  -H 'referer: https://portal.blyott.com/admin-panel/tags/new' \
  -H 'accept-language: en-US,en;q=0.9' \
  --data-binary
'{"TagId":"123456ABCDEF","NFCId":"123456ABCDEF","TagType":2,"FixedLocationId":285,"Tag
HardwareId":1,"AssetId":null,"CustomFields":[{"Id":22,"Value":"123"}]}' \
```

Get Tag Details (GET /tag/{TagID})

Tag details can be easily retrieved by issuing a GET request against the */tag/{TagId}* endpoint where the TagId is Tag's MAC address identifier.

Curl example:

```
curl 'https://api.blyott.com/tag/0CF3EEB88D9B' |
  -H 'authority: api.blyott.com' |
  -H 'accept: application/json, text/plain, */*' |
  -H 'token: {token}' |
  -H 'origin: https://portal.blyott.com' |
  -H 'sec-fetch-site: same-site' |
  -H 'sec-fetch-mode: cors' |
  -H 'sec-fetch-dest: empty' |
  -H 'referer: https://portal.blyott.com/admin-panel/tags' |
  -H 'accept-language: en-US,en;q=0.9' |
```

Edit Tag (PUT /tag)

Tag properties can be edited by issuing a PUT request against the */tag* endpoint with the body consisting of with a body of JSON payload consisting of at least the following parameters:

1. TagId - BLE MAC address of the Tag
2. NFCId - Tag's NFC identifier, usually the same as BLE MAC
3. TagType:
    a. 1 - Mobile Tags assigned to assets expected to move.
    b. 2 - Fixed Tags assigned to assets not expected to move.
4. FixedLocation - only applicable if TagType is 2, then the value should be the set to the internal Id of the Location (see Locations) otherwise null.
5. TagHardwareId - Tag's Manufacturer and model information, should be set to Id of the HardwareModel (see Hardware).
6. AssetId - Asset to which the Tag is linked to, can be null if the Tag is unassigned. Id should be of an unassigned Asset (see Unassigned Assets).
7. Dynamic properties - any custom dynamic property added through Layout Builder.

Curl example:

```
curl 'https://api.blyott.com/tag' \
  -X 'PUT' \
  -H 'authority: api.blyott.com' \
  -H 'accept: application/json, text/plain, */*' \
  -H 'token: {token}' \
  -H 'content-type: application/json' \
  -H 'origin: https://portal.blyott.com' \
  -H 'sec-fetch-site: same-site' \
  -H 'sec-fetch-mode: cors' \
  -H 'sec-fetch-dest: empty' \
  -H 'referer: https://portal.blyott.com/admin-panel/tags/123456ABCDEF/edit' \
  -H 'accept-language: en-US,en;q=0.9' \
  --data-binary
'{"TagId":"123456ABCDEF","NFCId":"123456ABCDEF","TagType":2,"FixedLocationId":285,"Tag
HardwareId":1,"AssetId":null,"CustomFields":[{"Id":22,"Value":"123"}]}' \
```

Delete Tag (DELETE /tag/{TagID})

Tag can be easily removed from the system by issuing a DELETE request against the */tag/{TagId}* endpoint where the TagId is Tag's MAC address identifier.

Curl example:

```
curl 'https://api.blyott.com/tag/123456ABCDEF' |
  -X 'DELETE' |
  -H 'authority: api.blyott.com' |
  -H 'accept: application/json, text/plain, */*' |
  -H 'token: {token}' |
  -H 'origin: https://portal.blyott.com' |
  -H 'sec-fetch-site: same-site' |
  -H 'sec-fetch-mode: cors' |
  -H 'sec-fetch-dest: empty' |
  -H 'referer: https://portal.blyott.com/admin-panel/tags/123456ABCDEF/edit' |
  -H 'accept-language: en-US,en;q=0.9' |
```

Search Tags (POST /tagsAurora)

To search for specific Tag a post request to the */tagsAurora* endpoint should be issued with following Filtering properties in the JSON body payload:

1. Page - Since results are paginated by 20 items, the page of the required result set should be provided. If field not provided all results will be returned
2. PageSize - If there are more than 20 results, specify this filter value.
    a. if not specified, number of items per page is 20.
    b. if number is less than 20, it is interpreted as 20.
    c. if number is greater than 1000, it is interpreted as 1000.
3. GlobalSearch - Searches the content in almost all grid columns.
4. SortBy - By which Tag property should the results be sorted.
5. SortOrder - Sorting direction can be:
    a. Asc - Ascending order
    b. Desc - Descending order
6. Filters - Arrays of applied filter, contains the list of following elements:
    a. FilterBy - Property by which to filter on.
    b. FilterContent - Array of values to search the value requested property against based on the contains type of logic.

Possible Filter properties:

7. TagId - filter by Tag MAC Address.
8. NFCId - filter by Tag NFC Identifier.
9. TagType - filter by Tag Type.
10. FixedLocationName - filter by Location Name Asset should be located.
11. HardwareModel - filter by Tag's Hardware Model value.
12. AssetName - filter by Assigned Asset's Name.
13. Activity - filter by Activity (see Activity).
14. Dynamic properties - filter by any custom dynamic property.

Take note that alongside the standard Tag properties, this endpoint will return Rssi and LastSeen properties as well.

Curl Example:

```
curl 'https://api.blyott.com/tagsAurora' |
  -H 'authority: api.blyott.com' |
  -H 'accept: application/json, text/plain, */*' |
  -H 'token: {token}' |
  -H 'content-type: application/json' |
```

```
  -H 'origin: https://portal.blyott.com' \
  -H 'sec-fetch-site: same-site' \
  -H 'sec-fetch-mode: cors' \
  -H 'sec-fetch-dest: empty' \

  -H 'referer: https://portal.blyott.com/admin-panel/tags' \
  -H 'accept-language: en-US,en;q=0.9' \
  --data-binary
'{"Page":1,"GlobalSearch":"0CF","SortBy":"TagId","SortOrder":"Asc","Filters":[{"FilterBy":"TagId","
FilterContent":["MAC"]},{"FilterBy":"NFCId","FilterContent":["NFC"]},{"FilterBy":"TagType","FilterC
ontent":[2]},{"FilterBy":"FixedLocationName","FilterContent":["FixedLoc"]},{"FilterBy":"Hardware
Model","FilterContent":["Hard"]},{"FilterBy":"AssetName","FilterContent":["assetAssigend"]},{"Filt
erBy":"Activity","FilterContent":["1"]}]}' \
```

List Unassigned Tags (GET /allUnassignedTags)

List of all unassigned Tag MAC addresses i.e., Tags not linked to an Asset can be retrieved by issuing a GET request against the */allUnassignedTags* endpoint.

Curl Example:

```
curl 'https://api.blyott.com/allUnassignedTags' |
  -H 'authority: api.blyott.com' |
  -H 'accept: application/json, text/plain, */*' |
  -H 'token: {token}' |
  -H 'origin: https://portal.blyott.com' |
  -H 'sec-fetch-site: same-site' |
  -H 'sec-fetch-mode: cors' |
  -H 'sec-fetch-dest: empty' |
  -H 'referer: https://portal.blyott.com/admin-panel/assets/new' |
  -H 'accept-language: en-US,en;q=0.9' |
```

ASSET SPECIFIC OPERATIONS:

Create Asset (POST /asset)

Asset is created by issuing a POST request to the platform against the */asset* endpoint with a body of JSON payload consisting of at least the following parameters:

1. AssetName - Name of the Asset.
2. AssetCode - Asset's unique serial identifier.
3. TagId - Tag to which the Asset is linked to, can be null if the Asset is unassigned. Id should be of an unassigned Asset (see Unassigned Tags).
4. AllUsers - If true, Asset is visible to all users, i.e., to users with any access level assigned.
5. AccessLevels - If AllUsers parameter is false, this one needs to be specified and it should exist in AccessLevel list (see Access Levels/Permissions).
6. Dynamic properties - any custom dynamic property added through Layout Builder.

Curl Example:

```
curl 'https://api.blyott.com/asset' |
  -H 'authority: api.blyott.com' |
  -H 'accept: application/json, text/plain, */*' |
  -H 'token: {token}' |
  -H 'content-type: application/json' |
  -H 'origin: https://portal.blyott.com' |
  -H 'sec-fetch-site: same-site' |
  -H 'sec-fetch-mode: cors' |
  -H 'sec-fetch-dest: empty' |
  -H 'referer: https://portal.blyott.com/admin-panel/assets/new' |
  -H 'accept-language: en-US,en;q=0.9' |
  --data-binary '{"AssetName":"Example asset name",
"AssetCode":"00001","TagId":null,"AllUsers":false,"AccessLevels":[29,8],"CustomFields":[{"Id":1
4,"Value":"123"},{"Id":15,"Value":"456"}]}' |
```

Get Asset Details (GET /assetDetails/{AssetID})

Asset details can be easily retrieved by issuing a GET request against the */assetDetails/{AssetID}* endpoint where the AssetID is AssetCode.

Curl Example:

```
curl 'https://api.blyott.com/assetDetails/test300' |
  -H 'authority: api.blyott.com' |
  -H 'accept: application/json, text/plain, */*' |
  -H 'token: {token}' |
  -H 'origin: https://portal.blyott.com' |
  -H 'sec-fetch-site: same-site' |
  -H 'sec-fetch-mode: cors' |
  -H 'sec-fetch-dest: empty' |
  -H 'referer: https://portal.blyott.com/admin-panel/assets' |
  -H 'accept-language: en-US,en;q=0.9' |
```

Edit Asset (PUT /asset)

Asset properties can be edited by issuing a PUT request against the */asset* endpoint with the body consisting of with a body of JSON payload consisting of at least the following parameters:

1. AssetId - Unique internal SHP Asset identifier.
2. AssetName - Name of the Asset.
3. AssetCode - Asset's unique serial identifier.
4. TagId - Tag to which the Asset is linked to, can be null if the Asset is unassigned. Id should be of an unassigned Asset (see Unassigned Tags).
5. AllUsers - If true, Asset is visible to all users, i.e., to users with any access level assigned.
6. AccessLevels - If AllUsers parameter is false, this one needs to be specified and it should exist in AccessLevel list (see Access Levels/Permissions).
7. Dynamic properties - any custom dynamic property added through Layout Builder.

Curl Example:

```
curl 'https://api.blyott.com/asset' \
  -X 'PUT' \
  -H 'authority: api.blyott.com' \
  -H 'accept: application/json, text/plain, */*' \
  -H 'token: {token}' \
  -H 'content-type: application/json' \
  -H 'origin: https://portal.blyott.com' \
  -H 'sec-fetch-site: same-site' \
  -H 'sec-fetch-mode: cors' \
  -H 'sec-fetch-dest: empty' \
  -H 'referer: https://portal.blyott.com/admin-panel/assets/test300/edit' \
  -H 'accept-language: en-US,en;q=0.9' \
  --data-binary
'{"AssetId":4368,"AssetName":"asset300","AssetCode":"test300","TagId":"test300","AllUsers":t
rue,"AccessLevels":null,"CustomFields":[{"Id":14,"Value":"note"},{"Id":15,"Value":"desc"}]}' \
```

Delete Asset (DELETE /asset/{AssetId})

Asset can be easily removed from the system by issuing a DELETE request against the */asset/{AssetId}* endpoint where the AssetId is Asset's Blyott internal identifier.

Curl example:

```
curl 'https://api.blyott.com/asset/3762' |
  -X 'DELETE' |
  -H 'authority: api.blyott.com' |
  -H 'accept: application/json, text/plain, */*' |
  -H 'token: {token}'' |
  -H 'origin: https://portal.blyott.com' |
  -H 'sec-fetch-site: same-site' |
  -H 'sec-fetch-mode: cors' |
  -H 'sec-fetch-dest: empty' |
  -H 'referer: https://portal.blyott.com/admin-panel/assets' |
  -H 'accept-language: en-US,en;q=0.9' |
```

Search Assets (POST /assets)

To search for specific Asset a POST request to the */assets* endpoint should be issued with following Filtering properties in the JSON body payload:

1. Page - Since results are paginated by 20 items, the page of the required result set should be provided. If the field is not provided all results will be returned.
2. SortBy - By which Asset property should the results be sorted.
3. PageSize - If there are more than 20 results, specify this filter value.
    a. if not specified, number of items per page is 20.
    b. if number is less than 20, it is interpreted as 20.
    c. if number is greater than 1000, it is interpreted as 1000.
4. GlobalSearch - Searches the content in almost all grid columns.
5. SortOrder - Sorting direction can be:
    a. Asc - Ascending order.
    b. Desc - Descending order.
6. Filters - Arrays of applied filter, contains the list of following elements:
    a. FilterBy - Property by which to filter on.
    b. FilterContent - Array of values to search the value requested property against based on the contains type of logic.

Possible Filter properties:

1. AssetName - filter by Asset Name.
2. AssetCode - fiter by Asset Code.
3. AccessLevel - filter by Access Level (see Access Levels/Permissions).
4. TagId - filter by Mac address of assigned Tag.
5. Activity - filter by Activity (see Activity).
6. LocationName - filter by Location Name where Asset is seen (only if assigned to Tag).
7. LocationCode - filter by Location Code where Asset is seen (only if assigned to Tag).
8. FixedLocationName - filter by Location Name where Asset should be located.
9. Dynamic properties - filter by any custom dynamic property.

Take note that alongside the standard Asset properties, this endpoint will return Rssi and TimeLastSeen properties as well.

Curl Example:

```
curl 'https://api.blyott.com/assets' \
  -H 'authority: api.blyott.com' \
```

```
  -H 'accept: application/json, text/plain, */*' \
  -H 'token: {token}' \
  -H 'content-type: application/json' \
  -H 'origin: https://portal.blyott.com' \
  -H 'sec-fetch-site: same-site' \
  -H 'sec-fetch-mode: cors' \
  -H 'sec-fetch-dest: empty' \

  -H 'referer: https://portal.blyott.com/admin-panel/assets' \
  -H 'accept-language: en-US,en;q=0.9' \
  --data-binary
'{"Page":1,"GlobalSearch":"","SortBy":"AssetName","SortOrder":"Asc","Filters":[{"FilterBy":"Asset
Name","FilterContent":["name"]},{"FilterBy":"AssetCode","FilterContent":["code"]},{"FilterBy":"Ta
gId","FilterContent":["assigned"]},{"FilterBy":"Activity","FilterContent":["2"]},{"FilterBy":"LocationN
ame","FilterContent":["locname"]},{"FilterBy":"LocationCode","FilterContent":["locCode"]},{"Filter
By":"FixedLocationName","FilterContent":["FixedLoc"]}]}' \
```

List Unassigned Assets (GET /allUnassignedAssets)

List of all unassigned Assets i.e., Assets not linked to an Tag can be retrieved by issuing a GET request against the */allUnassignedAssets* endpoint.

Curl Example:

```
curl 'https://api.blyott.com/allUnassignedTags' |
  -H 'authority: api.blyott.com' |
  -H 'accept: application/json, text/plain, */*' |
  -H 'token: {token}' |
  -H 'origin: https://portal.blyott.com' |
  -H 'sec-fetch-site: same-site' |
  -H 'sec-fetch-mode: cors' |
  -H 'sec-fetch-dest: empty' |
  -H 'referer: https://portal.blyott.com/admin-panel/assets/new' |
  -H 'accept-language: en-US,en;q=0.9' |
```

Get Asset History (POST /getAssetHistory)

To retrieve specific Assets historical locations on a given time range a POST request to the */getAssetHistory* endpoint should be issued with following request properties in the JSON body payload:

1. AssetIds - Array of Assets to be searched for. Assets are represented by their internal Asset identifiers (See Assets).
2. From - Date and time representing the beginning of a search range.
3. To - Date and time representing the end of a search range.

For each of the Asset provided an array of HistoryRecord will be provided with the times representing:

1. LocationId - Location Asset was present for this record, represented by Location internal identifier (See Location).
2. StartTime - Date and time representing the start of Assets' occurrence on this Location.
3. EndTime - Date and time representing the end of Assets' occurrence on this Location.

Curl Example:

```
curl 'https://api.blyott.com/getAssetHistory' |
  -H 'authority: api.blyott.com' |
  -H 'accept: application/json, text/plain, */*' |
  -H 'token: {token}' |
  -H 'content-type: application/json' |
  -H 'origin: https://portal.blyott.com' |
  -H 'sec-fetch-site: same-site' |
  -H 'sec-fetch-mode: cors' |
  -H 'sec-fetch-dest: empty' |
  -H 'referer: https://portal.blyott.com/admin-panel/assets/new' |
  -H 'accept-language: en-US,en;q=0.9' |
  --data-binary '{
    "AssetIds": [1,2,3,4,5,6],
    "From":    "2021-01-1T00:00:00.000Z",
    "To":   "2021-01-2T00:00:00.000Z"}'  |
```

LOCATION SPECIFIC OPERATIONS:

Create Location (POST /location)

Location is created by issuing a POST request to the platform against the */location* endpoint with a body of JSON payload consisting of at least the following parameters:

1. LocationName - Name of the Location.
2. LocationCode - Unique Location identifier such as room number etc.
3. LocationType:
    a. 0 - Active Location meaning assets are utilized on this Location.
    b. 1 - Passive Location meaning Storage, assets are not utilized here.
4. Dynamic properties - any custom dynamic property added through Layout Builder.

Curl Example:

```
curl 'https://api.blyott.com/location' \
  -H 'authority: api.blyott.com' \
  -H 'accept: application/json, text/plain, */*' \
  -H 'token: {token}' \
  -H 'content-type: application/json' \
  -H 'origin: https://portal.blyott.com' \
  -H 'sec-fetch-site: same-site' \
  -H 'sec-fetch-mode: cors' \
  -H 'sec-fetch-dest: empty' \
  -H 'referer: https://portal.blyott.com/admin-panel/locations/new' \
  -H 'accept-language: en-US,en;q=0.9' \
  --data-binary
'{"LocationName":"Test","LocationCode":"Test","LocationType":0,"CustomFields":[{"Id":1,"Value
":""}]}' \
```

Get Location Details (GET /location/{LocationID})

Location details can be easily retrieved by issuing a GET request against the */location/{LocationID}* endpoint where the LocationID is Location's Blyott internal identifier.

Curl Example:

```
curl 'https://api.blyott.com/location/3186' \
  -H 'authority: api.blyott.com' \
  -H 'accept: application/json, text/plain, */*' \
  -H 'token: {token}'\
  -H 'origin: https://portal.blyott.com' \
  -H 'sec-fetch-site: same-site' \
  -H 'sec-fetch-mode: cors' \
  -H 'sec-fetch-dest: empty' \
  -H 'referer: https://portal.blyott.com/admin-panel/locations' \
  -H 'accept-language: en-US,en;q=0.9' \
```

Edit Location (PUT /location)

Location properties can be edited by issuing a PUT request against the */location* endpoint with the body consisting of with a body of JSON payload consisting of at least the following parameters:

1. LocationId - Unique internal Blyott Location identifier.
2. LocationName - Name of the Location.
3. LocationCode - Unique Location identifier such as room number etc.
4. LocationType:
    a. 0 - Active Location meaning assets are utilized on this Location.
    b. 1 - Inactive location meaning Storage, assets are not utilized here.
5. Dynamic properties - any custom dynamic property added through Layout Builder.

Curl Example:

```
curl 'https://api.blyott.com/location' \
  -X 'PUT' \
  -H 'authority: api.blyott.com' \
  -H 'accept: application/json, text/plain, */*' \
  -H 'token: {token}' \
  -H 'content-type: application/json' \
  -H 'origin: https://portal.blyott.com' \
  -H 'sec-fetch-site: same-site' \
  -H 'sec-fetch-mode: cors' \
  -H 'sec-fetch-dest: empty' \
  -H 'referer: https://portal.blyott.com/admin-panel/locations/3186/edit' \
  -H 'accept-language: en-US,en;q=0.9' \
  --data-binary '{"LocationId":"3186","LocationName":"Ground floor - North2","LocationCode":"TQN2","LocationType":0,"CustomFields":[{"Id":1,"Value":"Room 12"}]}' \
  |
```

Delete Location (DELETE /location/{LocationId})

Location can be easily removed from the system by issuing a DELETE request against the */location/{LocationId}* endpoint where the LocationId is Location's Blyott internal identifier.

Curl Example:

```
curl 'https://api.blyott.com/location/27484' |
  -X 'DELETE' |
  -H 'authority: api.blyott.com' |
  -H 'accept: application/json, text/plain, */*' |
  -H 'token: {token}' |
  -H 'origin: https://portal.blyott.com' |
  -H 'sec-fetch-site: same-site' |
  -H 'sec-fetch-mode: cors' |
  -H 'sec-fetch-dest: empty' |
  -H 'referer: https://portal.blyott.com/admin-panel/locations' |
  -H 'accept-language: en-US,en;q=0.9' |
```

Search Locations (POST /locationsAurora)

To search for specific Location a post request to the */locationsAurora* endpoint should be issued with following Filtering properties in the JSON body payload:

1. Page - Since results are paginated by 20 items, the page of the required result set should be provided. If the field is not provided all results will be returned.
2. SortBy - By which Asset property should the results be sorted.
3. PageSize - If there are more than 20 results, specify this filter value.
    a. if not specified, number of items per page is 20.
    b. if number is less than 20, it is interpreted as 20.
    c. if number is greater than 1000, it is interpreted as 1000.
4. GlobalSearch - Searches the content in almost all grid columns.
5. SortOrder - Sorting direction can be:
    a. Asc - Ascending order
    b. Desc - Descending order
6. Filters - Arrays of applied filter, contains the list of following elements:
    a. FilterBy - Property by which to filter on.
    b. FilterContent - Array of values to search the value requested property against based on the contains type of logic.

Possible Filter properties:

1. LocationName - Name of the Location.
2. LocationCode - Unique Location identifier such as room number etc.
3. LocationType:
    a. 0 - Active Location meaning assets are utilized on this Location.
    b. 1 - Inactive location meaning Storage, assets are not utilized here.
4. Dynamic properties - filter by any custom dynamic property.

Curl Example:

```
curl 'https://api.blyott.com/locationsAurora' \
  -H 'authority: api.blyott.com' \
  -H 'accept: application/json, text/plain, */*' \
  -H 'token: {token}' \
  -H 'content-type: application/json' \
  -H 'origin: https://portal.blyott.com' \
  -H 'sec-fetch-site: same-site' \
  -H 'sec-fetch-mode: cors' \
  -H 'sec-fetch-dest: empty' \
  -H 'referer: https://portal.blyott.com/' \
```

```
  -H 'accept-language: en-US,en;q=0.9' \
  --data-binary
'{"Page":1,"PageSize":50,"GlobalSearch":"","SortBy":"LocationName","SortOrder":"Asc","Filters"

:[{"FilterBy":"LocationName","FilterContent":["name"]},{"FilterBy":"LocationCode","FilterContent
":["code"]},{"FilterBy":"LocationType","FilterContent":["0"]}]}' \
```

List all Locations (GET /listAllLocations)

Issuing a GET request to */listAllLocations* will retrieve a list of all existing Location values and their associated entries that can be used to create/edit a Tag or a Locator.

Curl Example:

```
curl 'https://api.blyott.com/listAllLocations' |
  -H 'authority: api.blyott.com' |
  -H 'accept: application/json, text/plain, */*' |
  -H 'token: {token}' |
  -H 'origin: https://portal.blyott.com' |
  -H 'sec-fetch-site: same-site' |
  -H 'sec-fetch-mode: cors' |
  -H 'sec-fetch-dest: empty' |
  -H 'referer: https://portal.blyott.com/' |
  -H 'accept-language: en-US,en;q=0.9' |
```

LOCATOR SPECIFIC OPERATIONS:

Create Locator (POST /locators)

Locator is created by issuing a POST request to the platform against the */locators* endpoint with a body of JSON payload consisting of at least the following parameters:

1. LocatorId - MAC address of the Locator device.
2. LocatorName - Name of the Locator device.
3. LocationId - Internal Blyott Location identifier (see Locations).
4. LocatorType - Type of Locator used, one of following:
    a. 0 - Mobile Locator, Locator that is expected to change location.
    b. 1 - WiFi Locator, Internal WiFi Locators used by on-premises clients.
    c. 2 - Fixed Locator, Locator provided by Blyott that is fixed to a Location.
5. LocatorSerialNumber - Internal serial number of the Locator device, optional and can be blank.
6. LocatorImsi - Imsi value of the Locator, optional if applicable needs to be unique.
7. LocatorImei - Imei value of Locator, optional.
8. LocatorHardwareId - Locator's Manufacturer and Model information, should be set to Id of the Locator HardwareModel (see Hardware).
9. Dynamic properties - any custom dynamic property added through Layout Builder.

Curl Example:

```
curl 'https://api.blyott.com/locator' \
  -H 'authority: api.blyott.com' \
  -H 'accept: application/json, text/plain, */*' \
  -H 'token: {token}' \
  -H 'content-type: application/json' \
  -H 'origin: https://portal.blyott.com' \
  -H 'sec-fetch-site: same-site' \
  -H 'sec-fetch-mode: cors' \
  -H 'sec-fetch-dest: empty' \
  -H 'referer: https://portal.blyott.com/admin-panel/locators/new' \
  -H 'accept-language: en-US,en;q=0.9' \
  --data-binary
'{"LocatorName":"Test","LocatorId":"Test","LocatorType":2,"LocationId":285,"LocatorHardware
Id":16,"LocatorSerialNumber":"Test","LocatorImsi":"Test","LocatorImei":"Test","CustomFields":[{
"Id":12,"Value":"1"}]}' \
```

Get Locator Details (GET /locator/{LocatorID})

Location details can be easily retrieved by issuing a GET request against the */location/{LocatorID}* endpoint where the LocatorID is Locator's Mac internal identifier.

Curl Example:

```
curl 'https://api.blyott.com/locator/A4CF12152728' \
  -H 'authority: api.blyott.com' \
  -H 'accept: application/json, text/plain, */*' \
  -H 'token: {token}'\
  -H 'origin: https://portal.blyott.com' \
  -H 'sec-fetch-site: same-site' \
  -H 'sec-fetch-mode: cors' \
  -H 'sec-fetch-dest: empty' \
  -H 'referer: https://portal.blyott.com/admin-panel/locator/A4CF12152728 \
  -H 'accept-language: en-US,en;q=0.9' \
```

Edit Locators (PUT /locator)

Locator is edited by issuing a PUT request to the platform against the */locators* endpoint with a body of JSON payload consisting of at least the following parameters:

1. LocatorId - MAC address of the Locator device.
2. LocatorName - Name of the Locator device.
3. LocationId - Internal Blyott Location identifier (see Locations).
4. LocatorType - Type of Locator used, one of following:
    a. 0 - Mobile Locator, Locator that is expected to change location.
    b. 1 - WiFi Locator, Internal WiFi locators used by on-premises clients.
    c. 2 - Fixed Locator, locator provided by Blyott that is fixed to a Location.
5. LocatorSerialNumber - Internal serial number of the Locator device, optional can be blank.
6. LocatorImsi - Imsi value of the Locator, optional if applicable needs to be unique.
7. LocatorImei - Imei value of Locator, optional.
8. LocatorHardwareId - Locator's Manufacturer and Model information, should be set to Id of the Locator HardwareModel (see Hardware).
9. Dynamic properties - any custom dynamic property added through Layout Builder.

Curl Example:

```
curl 'https://api.blyott.com/locator' |
  -X 'PUT' |
  -H 'authority: api.blyott.com' |
  -H 'accept: application/json, text/plain, */*' |
  -H 'token: {token}' |
  -H 'content-type: application/json' |
  -H 'origin: https://portal.blyott.com' |
  -H 'sec-fetch-site: same-site' |
  -H 'sec-fetch-mode: cors' |
  -H 'sec-fetch-dest: empty' |
  -H 'referer: https://portal.blyott.com/' |
  -H 'accept-language: en-US,en;q=0.9' |
  --data-binary '{"LocatorName":"Locator
3","LocatorId":"A4CF12152728","LocatorType":2,"LocationId":3187,"LocatorHardwareId":16,"Lo
catorSerialNumber":"sfsfsfsf","LocatorImsi":"sfsfsdhedagsdfgs","LocatorImei":null,"CustomFiel
ds":[{"Id":12,"Value":"sd"}]}' |
```

Delete Locator (DELETE /locator/{LocatorId})

Locator can be easily removed from the system by issuing a DELETE request against the */locator/{LocatorId}* endpoint where the LocatorId is Locator's Mac internal identifier.

Curl Example:

```
curl 'https://api.blyott.com/locator/S21' \
  -X 'DELETE' \
  -H 'authority: api.blyott.com' \
  -H 'accept: application/json, text/plain, */*' \
  -H 'token: {token}' \
  -H 'origin: https://portal.blyott.com' \
  -H 'sec-fetch-site: same-site' \
  -H 'sec-fetch-mode: cors' \
  -H 'sec-fetch-dest: empty' \
  -H 'referer: https://portal.blyott.com/' \
  -H 'accept-language: en-US,en;q=0.9' \
```

Search Locators (POST /locatorsAurora)

To search for specific Locator a POST request to the */locatorsAurora* endpoint should be issued with following Filtering properties in the JSON body payload:

1. Page - Since results are paginated by 20 items, the page of the required result set should be provided. If field is not provided all results will be returned.
2. SortBy - By which Locator property should the results be sorted.
3. PageSize - If there are more than 20 results, specify this filter value.
    a. if not specified, number of items per page is 20.
    b. if number is less than 20, it is interpreted as 20.
    c. if number is greater than 1000, it is interpreted as 1000.
4. GlobalSearch - Searches the content in almost all grid columns.
5. SortOrder - Sorting direction can be:
    a. Asc - Ascending order.
    b. Desc - Descending order.
6. Filters - Arrays of applied filter, contains the list of following elements:
    a. FilterBy - Property by which to filter on.
    b. FilterContent - Array of values to search the value requested property against based on the contains type of logic.

Possible Filter properties:

1. LocatorName - filter by name of the Locator.
2. LocationName - filter by name of the Location.
3. LocationCode - filter by unique Location identifier such as room number etc.
4. LocatorId - filter by MAC address of the Locator.
5. Activity - filter by Activity (see Activity).
6. LocatorSerialNumber - filter by Internal serial number of the Locator device.
7. LocatorImsi - filter by Locator Imsi values.
8. LocatorImei - filter by Locator Imei values.
9. LocatorType - filter by Locator type values.
10. LocatorHardwareModel - filter by Locator's Hardware Model value.
11. Dynamic properties - filter by any custom dynamic property.

Curl Example:

```
curl 'https://api.blyott.com/locatorsAurora' \
  -H 'authority: api.blyott.com' \
  -H 'accept: application/json, text/plain, */*' \
  -H 'token: {token}' \
  -H 'content-type: application/json' \
```

```
  -H 'origin: https://portal.blyott.com' \
  -H 'sec-fetch-site: same-site' \

  -H 'sec-fetch-mode: cors' \
  -H 'sec-fetch-dest: empty' \
  -H 'referer: https://portal.blyott.com/' \
  -H 'accept-language: en-US,en;q=0.9' \
  --data-binary
'{"Page":1,"PageSize":50,"GlobalSearch":"","SortBy":"LocatorName","SortOrder":"Asc","Filters":[
{"FilterBy":"LocatorName","FilterContent":["name"]},{"FilterBy":"LocationName","FilterContent":[
"locName"]},{"FilterBy":"LocationCode","FilterContent":["locCode"]},{"FilterBy":"LocatorId","Filter
Content":["MAC"]},{"FilterBy":"Activity","FilterContent":["1"]},{"FilterBy":"LocatorSerialNumber","Fi
lterContent":["Serial"]},{"FilterBy":"LocatorImsi","FilterContent":["imsi"]},{"FilterBy":"LocatorHard
wareModel","FilterContent":["model"]}]}' \
```

ACCESS LEVEL SPECIFIC OPERATIONS:

Create Access Level (POST /accessLevel)

Access Level is created by issuing a POST request to the platform against the */accessLevel* endpoint with a body of JSON payload consisting of at least the following parameters:

1. AccessLevelName - name of the Access Level.
2. AccessLevelDescription - additional description of the Access Level, optional.

Curl Example:

```
curl 'https://api.blyott.com/accessLevel' |
  -H 'authority: api.blyott.com' |
  -H 'accept: application/json, text/plain, */*' |
  -H 'token: {token}' |
  -H 'content-type: application/json' |
  -H 'origin: https://portal.blyott.com' |
  -H 'sec-fetch-site: same-site' |
  -H 'sec-fetch-mode: cors' |
  -H 'sec-fetch-dest: empty' |
  -H 'referer: https://portal.blyott.com/' |
  -H 'accept-language: en-US,en;q=0.9' |
  --data-binary '{"AccessLevelName":"Dermatology","AccessLevelDescription":"DERM"}' |
```

Get Access Level Details (GET /accessLevel/{accessLevelID})

Access Level details can be easily retrieved by issuing a GET request against the */accessLevel/{accessLevelID}* endpoint where the AccessLevelID is an internal identifier.

Curl Example:

```
curl 'https://api.blyott.com/accessLevel/36' |
  -H 'authority: api.blyott.com' |
  -H 'accept: application/json, text/plain, */*' |
  -H 'token: {token}' |
  -H 'origin: https://portal.blyott.com' |
  -H 'sec-fetch-site: same-site' |
  -H 'sec-fetch-mode: cors' |
  -H 'sec-fetch-dest: empty' |
  -H 'referer: https://portal.blyott.com/' |
  -H 'accept-language: en-US,en;q=0.9' |
```

Edit Access Level (POST /accessLevel/{AccessLevelID})

Access Level is edited by issuing a PUT request to the platform against the */accessLevel* endpoint with a body of JSON payload consisting of at least the following parameters:

1. AccessLevelId - unique internal Blyott Access Level identifier.
2. AccessLevelName - name of the Access Level.
3. AccessLevelDescription - additional description of the Access Level, optional.

Curl Example:

```
curl 'https://api.blyott.com/accessLevel' |
  -X 'PUT' |
  -H 'authority: api.blyott.com' |
  -H 'accept: application/json, text/plain, */*' |
  -H 'token: {token}' |
  -H 'content-type: application/json' |
  -H 'origin: https://portal.blyott.com' |
  -H 'sec-fetch-site: same-site' |
  -H 'sec-fetch-mode: cors' |
  -H 'sec-fetch-dest: empty' |
  -H 'referer: https://portal.blyott.com/' |
  -H 'accept-language: en-US,en;q=0.9' |
  --data-binary
'{"AccessLevelId":"36","AccessLevelName":"Cardiology","AccessLevelDescription":"CARD"}' |
```

Delete Access Level (DELETE /accessLevel/{AccessLevelID})

Access Level can be easily removed from the system by issuing a DELETE request against the */accessLevel/{AccessLevelId}* endpoint where the Access Level ID is Blyott internal identifier.

Curl Example:

```
curl 'https://api.blyott.com/accessLevel/36' |
  -X 'DELETE' |
  -H 'authority: api.blyott.com' |
  -H 'accept: application/json, text/plain, */*' |
  -H 'token: {token}' |
  -H 'origin: https://portal.blyott.com' |
  -H 'sec-fetch-site: same-site' |
  -H 'sec-fetch-mode: cors' |
  -H 'sec-fetch-dest: empty' |
  -H 'referer: https://portal.blyott.com/' |
  -H 'accept-language: en-US,en;q=0.9' |
```

Search Access Levels (POST /accessLevels)

To search for specific Access Level a POST request to the */accessLevels* endpoint should be issued with following Filtering properties in the JSON body payload:

1. Page - Since results are paginated by 20 items, the page of the required result set should be provided. If field is not provided all results will be returned.
2. SortBy - By which Locator property should the results be sorted.
3. PageSize - If there are more than 20 results, specify this filter value.
    a. if not specified, number of items per page is 20.
    b. if number is less than 20, it is interpreted as 20.
    c. if number is greater than 1000, it is interpreted as 1000.
4. GlobalSearch - Searches the content in almost all grid columns.
5. SortOrder - Sorting direction can be:
    a. Asc - Ascending order.
    b. Desc - Descending order.
6. Filters - Arrays of applied filter, contains the list of following elements:
    a. FilterBy - Property by which to filter on.
    b. FilterContent - Array of values to search the value requested property against based on the contains type of logic.

Possible Filter properties:

1. AccessLevelName - filter by name of the Access Level.
2. AccessLevelDescription - filter by additional description of the Access Level.

Curl Example:

```
curl 'https://api.blyott.com/accessLevels' |
  -H 'authority: api.blyott.com' |
  -H 'accept: application/json, text/plain, */*' |
  -H 'token: {token}' |
  -H 'content-type: application/json' |
  -H 'origin: https://portal.blyott.com' |
  -H 'sec-fetch-site: same-site' |
  -H 'sec-fetch-mode: cors' |
  -H 'sec-fetch-dest: empty' |
  -H 'referer: https://portal.blyott.com/' |
  -H 'accept-language: en-US,en;q=0.9' |
  --data-binary
'{"Page":1,"PageSize":50,"GlobalSearch":"","SortBy":"AccessLevelName","SortOrder":"Asc","Filt
```

*ers":[{"FilterBy":"AccessLevelName","FilterName":"Access Levels","FilterContent":["DERM"]},{"FilterBy":"AccessLevelDescription","FilterName":"Description","FilterContent":["D"]}]}' *

List all Access Levels (GET /allAccessLevels)

Issuing a GET request to */allAccessLevels* will retrieve a list of all existing Access Level values and their associated entries that can be used to create/edit a User or an Asset.

Curl Example:

```
curl 'https://api.blyott.com/allAccessLevels' |
  -H 'authority: api.blyott.com' |
  -H 'accept: application/json, text/plain, */*' |
  -H 'token: {token}' |
  -H 'origin: https://portal.blyott.com' |
  -H 'sec-fetch-site: same-site' |
  -H 'sec-fetch-mode: cors' |
  -H 'sec-fetch-dest: empty' |
  -H 'referer: https://portal.blyott.com/' |
  -H 'accept-language: en-US,en;q=0.9' |
```

USER SPECIFIC OPERATIONS:

Create User (POST /user)

User is created by issuing a POST request to the platform against the */user* endpoint with a body of JSON payload consisting of the following parameters:

1. FirstName - user's first name.
2. Last name - user's last name.
3. username - uniquely identifies user, required for users to login.
4. email - user's email address.
5. role - defines access control, can be one of the following:
    a. Administrator
    b. User
6. totalAccess - if true, the user can access all current levels and the ones that will be added in the future.
7. accessLevels - if Total Access parameter is false (limited access), this one needs to be specified and it should exist in AccessLevel list (see Access Levels/Permissions).

Curl Example:

```
curl 'https://api.blyott.com/user' |
  -H 'authority: api.blyott.com' |
  -H 'accept: application/json, text/plain, */*' |
  -H 'token: {token}' |
  -H 'content-type: application/json' |
  -H 'origin: https://portal.blyott.com' |
  -H 'sec-fetch-site: same-site' |
  -H 'sec-fetch-mode: cors' |
  -H 'sec-fetch-dest: empty' |
  -H 'referer: https://portal.blyott.com/' |
  -H 'accept-language: en-US,en;q=0.9' |
  --data-binary
'{"FirstName":"Test","LastName":"Test","username":"test123","email":"test@test.com","role":"Ad
ministrator","totalAccess":true,"accessLevels":null}' |
```

Get User Details (GET /user/{UserID})

User details can be easily retrieved by issuing a GET request against the */user/{UserID}* endpoint where the userID is an internal identifier.

Curl Example:

```
curl 'https://api.blyott.com/user/test123' |
  -H 'authority: api.blyott.com' |
  -H 'accept: application/json, text/plain, */*' |
  -H 'token: {token}' |
  -H 'origin: https://portal.blyott.com' |
  -H 'sec-fetch-site: same-site' |
  -H 'sec-fetch-mode: cors' |
  -H 'sec-fetch-dest: empty' |
  -H 'referer: https://portal.blyott.com/' |
  -H 'accept-language: en-US,en;q=0.9' |
```

Edit User (PUT /user)

User is edited by issuing a PUT request to the platform against the */user* endpoint with a body of JSON payload consisting of the following parameters:

1. FirstName - user's first name.
2. Last name - user's last name.
3. username - uniquely identifies user, required for users to login.
4. email - user's email address.
5. role - defines access control, can be one of the following:
    a. Administrator
    b. User
6. totalAccess - if true, the user can access all current levels and the ones that will be added in the future.
7. accessLevels - if Total Access parameter is false (limited access), this one needs to be specified and it should exist in AccessLevel list (see Access Levels/Permissions).

Curl Example:

```
curl 'https://api.blyott.com/user' \
  -X 'PUT' \
  -H 'authority: api.blyott.com' \
  -H 'accept: application/json, text/plain, */*' \
  -H 'token: {token}' \
  -H 'content-type: application/json' \
  -H 'origin: https://portal.blyott.com' \
  -H 'sec-fetch-site: same-site' \
  -H 'sec-fetch-mode: cors' \
  -H 'sec-fetch-dest: empty' \
  -H 'referer: https://portal.blyott.com/' \
  -H 'accept-language: en-US,en;q=0.9' \
  --data-binary
'{"FirstName":"Test","LastName":"Test","Username":"test123","Email":"test@test.com","Role":"A
dministrator","AccessLevels":null,"TotalAccess":true}' \
```

Delete User (DELETE /user/{UserID})

User can be easily removed from the system by issuing a DELETE request against the */user/{UserId}* endpoint where the User ID is Blyott internal identifier.

Curl Example:

```
curl 'https://api.blyott.com/user/test123' |
  -X 'DELETE' |
  -H 'authority: api.blyott.com' |
  -H 'accept: application/json, text/plain, */*' |
  -H 'token: {token}' |
  -H 'origin: https://portal.blyott.com' |
  -H 'sec-fetch-site: same-site' |
  -H 'sec-fetch-mode: cors' |
  -H 'sec-fetch-dest: empty' |
  -H 'referer: https://portal.blyott.com/' |
  -H 'accept-language: en-US,en;q=0.9' |
```

Search Users (POST /users)

To search for specific User a POST request to the */users* endpoint should be issued with following Filtering properties in the JSON body payload:

1. Page - Since results are paginated by 20 items, the page of the required result set should be provided. If field is not provided all results will be returned.
2. SortBy - By which Locator property should the results be sorted.
3. PageSize - If there are more than 20 results, specify this filter value.
    a. if not specified, number of items per page is 20.
    b. if number is less than 20, it is interpreted as 20.
    c. if number is greater than 1000, it is interpreted as 1000.
4. GlobalSearch - Searches the content in almost all grid columns.
5. SortOrder - Sorting direction can be:
    a. Asc - Ascending order.
    b. Desc - Descending order.
6. Filters - Arrays of applied filter, contains the list of following elements:
    a. FilterBy - Property by which to filter on.
    b. FilterContent - Array of values to search the value requested property against based on the contains type of logic.

Possible Filter properties:

1. FirstName - filter by First Name.
2. LastName - filter by Last Name.
3. Username - filter by Username.
4. Email - filter by Email.
5. Role - filter by Role.
6. AccessLevel - filter by Access Level (see Access Levels/Permissions).

Curl Example:

```
curl 'https://api.blyott.com/users' \
  -H 'authority: api.blyott.com' \
  -H 'accept: application/json, text/plain, */*' \
  -H 'token: {token}' \
  -H 'content-type: application/json' \
  -H 'origin: https://portal.blyott.com' \
  -H 'sec-fetch-site: same-site' \
  -H 'sec-fetch-mode: cors' \
  -H 'sec-fetch-dest: empty' \
  -H 'referer: https://portal.blyott.com/' \
```

```
  -H 'accept-language: en-US,en;q=0.9' \

  --data-binary
'{"Page":1,"PageSize":50,"GlobalSearch":"test","SortBy":"FirstName","SortOrder":"Asc","Filters":[
{"FilterBy":"FirstName","FilterName":"First
Name","FilterContent":[""]},{"FilterBy":"LastName","FilterName":"Last
Name","FilterContent":[""]},{"FilterBy":"Username","FilterName":"Username","FilterContent":[""]},
{"FilterBy":"Email","FilterName":"Email","FilterContent":[""]},{"FilterBy":"Role","FilterName":"Role"
,"FilterContent":[""]},{"FilterBy":"AccessLevel","FilterName":"Access     Levels","FilterContent":[""]}]}'
|
```

Resend Invite (POST /user/{UserID}/resendInvitation)

Resend invitation can be easily triggered by issuing a PUT request against the */user/{UserID}}/resendInvitation* endpoint where the userID is an internal identifier.

Curl Example:

```
curl 'https://api.blyott.com/user/{UserID}/resendInvitation' |
  -H 'authority: api.blyott.com' |
  -H 'accept: application/json, text/plain, */*' |
  -H 'token: {token}' |
  -H 'content-type: application/json' |
  -H 'origin: https://portal.blyott.com' |
  -H 'sec-fetch-site: same-site' |
  -H 'sec-fetch-mode: cors' |
  -H 'sec-fetch-dest: empty' |
  -H 'referer: https://portal.blyott.com/' |
  -H 'accept-language: en-US,en;q=0.9' |
```

LAYOUT BUILDER OPERATIONS:

In order to add customized information for each entity (Tag, Asset, Locator and Location), dynamic properties are implemented.

Create dynamic property (POST /layoutBuilder/{entityName})

Dynamic property is created by issuing a POST request to the platform against the */layoutBuilder/{entityName}* endpoint where Entity Name is name of one of the following items:

- tag
- locator
- asset
- location

A body of JSON payload consists of the parameter:

1. Name - name of the Dynamic Property.

Curl Example:

```
curl 'https://api.blyott.com/layoutBuilder/asset' \
  -H 'authority: api.blyott.com' \
  -H 'accept: application/json, text/plain, */*' \
  -H 'token: {token}' \
  -H 'content-type: application/json' \
  -H 'origin: https://portal.blyott.com' \
  -H 'sec-fetch-site: same-site' \
  -H 'sec-fetch-mode: cors' \
  -H 'sec-fetch-dest: empty' \
  -H 'referer: https://portal.blyott.com/' \
  -H 'accept-language: en-US,en;q=0.9' \
  --data-binary '{"Name":"New dynamic property"}' \
```

Get dynamic property (GET /layoutBuilder/{entityName})

Dynamic properties can be easily retrieved by issuing a GET request to the platform against the */layoutBuilder/{entityName}* endpoint where Entity Name is name of one of the following items:

- tag
- locator
- asset
- location

Curl Example:

```
curl 'https://api.blyott.com/layoutBuilder/asset' |
  -H 'authority: api.blyott.com' |
  -H 'accept: application/json, text/plain, */*' |
  -H 'token: {token}' |
  -H 'content-type: application/json' |
  -H 'origin: https://portal.blyott.com' |
  -H 'sec-fetch-site: same-site' |
  -H 'sec-fetch-mode: cors' |
  -H 'sec-fetch-dest: empty' |
  -H 'referer: https://portal.blyott.com/' |
  -H 'accept-language: en-US,en;q=0.9' |
```

Edit dynamic property (PUT /layoutBuilder/{entityName})

Dynamic property is edited by issuing a PUT request to the platform against the */layoutBuilder/{entityName}* endpoint where Entity Name is name of one of the following items:

- tag
- locator
- asset
- location

A body of JSON payload consists of the following parameters:

1. Id - unique internal Blyott Dynamic Property identifier.
2. Name - name of the Dynamic Property.

Curl Example:

```
curl 'https://api.blyott.com/layoutBuilder/asset' |
  -X 'PUT' |
  -H 'authority: api.blyott.com' |
  -H 'accept: application/json, text/plain, */*' |
  -H 'token: {token}' |
  -H 'content-type: application/json' |
  -H 'origin: https://portal.blyott.com' |
  -H 'sec-fetch-site: same-site' |
  -H 'sec-fetch-mode: cors' |
  -H 'sec-fetch-dest: empty' |
  -H 'referer: https://portal.blyott.com/' |
  -H 'accept-language: en-US,en;q=0.9' |
  --data-binary '{"Id":33,"Name":"New dynamic property"}' |
```

Archive dynamic property (PUT /layoutBuilder/{entityName}/archive)

Dynamic property is archived by issuing a PUT request to the platform against the */layoutBuilder/{entityName}/archive* endpoint where Entity Name is name of one of the following items:

- tag
- locator
- asset
- location

A body of JSON payload consists of the parameter:

1. CustomFieldIDs - unique internal Blyott Dynamic Property identifiers.

Curl Example:

```
curl 'https://api.blyott.com/layoutBuilder/asset/archive' \
  -H 'authority: api.blyott.com' \
  -H 'accept: application/json, text/plain, */*' \
  -H 'token: {token}' \
  -H 'content-type: application/json' \
  -H 'origin: https://portal.blyott.com' \
  -H 'sec-fetch-site: same-site' \
  -H 'sec-fetch-mode: cors' \
  -H 'sec-fetch-dest: empty' \
  -H 'referer: https://portal.blyott.com/' \
  --data-binary '{"CustomFieldIds":[33]}' \
```

Restore dynamic property (PUT /layoutBuilder/{entityName}/restore)

Dynamic property is archived by issuing a PUT request to the platform against the */layoutBuilder/{entityName}/restore* endpoint where Entity Name is name of one of the following items:

- tag
- locator
- asset
- location

A body of JSON payload consists of the parameter:

1. CustomFieldIDs - unique internal Blyott Dynamic Property identifiers

Curl Example:

```
curl 'https://api.blyott.com/layoutBuilder/asset/restore \
  -H 'authority: api.blyott.com' \
  -H 'accept: application/json, text/plain, */*' \
  -H 'token: {token}' \
  -H 'content-type: application/json' \
  -H 'origin: https://portal.blyott.com' \
  -H 'sec-fetch-site: same-site' \
  -H 'sec-fetch-mode: cors' \
  -H 'sec-fetch-dest: empty' \
  -H 'referer: https://portal.blyott.com/' \
  --data-binary '{"CustomFieldIds":[33,34]}' \
```

Delete dynamic property (PUT /layoutBuilder/{entityName})

Dynamic property is archived by issuing a PUT request to the platform against the */layoutBuilder/{entityName}/restore* endpoint where Entity Name is name of one of the following items:

- tag
- locator
- asset
- location

A body of JSON payload consists of the parameter:

1. CustomFieldIDs - unique internal Blyott Dynamic Property identifiers.

Curl Example:

```
curl 'https://api.blyott.com/layoutBuilder/asset/delete' \
  -X 'PUT' \
  -H 'authority: api.blyott.com' \
  -H 'accept: application/json, text/plain, */*' \
  -H 'token: {token}' \
  -H 'content-type: application/json' \
  -H 'origin: https://portal.blyott.com' \
  -H 'sec-fetch-site: same-site' \
  -H 'sec-fetch-mode: cors' \
  -H 'sec-fetch-dest: empty' \
  -H 'referer: https://portal.blyott.com/' \
  -H 'accept-language: en-US,en;q=0.9' \
  --data-binary '{"CustomFieldIds":[17,20]}' \
```

ADDITIONAL INFORMATION:

List of Tag Hardware Models (GET /hardware/0)

Issuing a GET request to */hardware/0* will retrieve a list of all possible HardwareModel values and their associated HardwareId entries that can be used to create/edit a Tag.

List of Locator Hardware Models (GET /hardware/1)

Issuing a GET request to */hardware/1* will retrieve a list of all possible HardwareModel values and their associated HardwareId entries that can be used to create/edit a Locator.

Activity Values

Values returned when searching and filtering for Assets/Tags/Locators can be any of the following values:

- 1 - seen 5 min ago & moving.
- 2 - seen 5 min ago.
- 3 - seen somewhere between 5 min to 2 hours ago.
- 4 - seen more than 2 hours ago.
- 5 - never seen.

COMMON USAGE SCENARIOS:

Link Tag scenarios:

Link unassigned Asset to unassigned Tag

1. Retrieve the list of all unassigned Assets.
2. Edit a Tag's AssetId field to one of the AssetId from the list of unassigned Tags in previous step.

Link unassigned Tag to a new Asset

1. Retrieve the list of all unassigned Tags.
2. Create a new Asset and set Asset's TagId field to one of the Tag Ids from the list of unassigned Tags in previous step.

Link new Asset to a new Tag

1. Create a new Tag and set AssetId field as null, thus creating a new unassigned Tag.
2. Create a new Asset and set Asset's TagId field to a Tag set in previous steps.

Unlink Tag scenarios:

Unlink Asset from Tag

1. Edit a Tag and set AssetId to null, thus creating an unassigned Tag.

Unlink Tag from Asset

Edit an Asset and set TagId to null, thus creating an unassigned Asset.

View Assets's Location

Issue a Search Assets call and set filtering property of AssetCode to desired value. Result will display Asset's current LocationName and LocationCode (implying Asset was assigned to a Tag and was seen by a Locator).

FINGERPRINTING:

Start Fingerprinting (POST /startFingerprinting)

Fingerprinting is started by issuing a POST request to the platform against the */startFingerprinting* endpoint with a body of JSON payload consisting of the following parameters:

1. TagId - BLE MAC address of the Tag.
2. LocationId - value should be the set to the internal Id of the Location (see Locations).
3. HardwareId - Tag's Manufacturer and model information, should be set to Id of the HardwareModel (see Hardware).

After the request is sent, Tag Type is set to 'Machine Learning' to avoid the possibility of user error. It is not possible to start fingerprinting for Tag which is already in the system. In that case, Tag needs to be deleted and added again using this endpoint.

Example:

```
{
    "TagId": "60C0BF209622",
    "LocationId": 11,
    "HardwareId": 1
}
```

End Fingerprinting (POST /endFingerprinting)

Fingerprinting is stopped and removed from the system by issuing a POST request to the platform against the */endFingerprinting* endpoint with a body of JSON payload consisting of only one parameter:

1. TagId - BLE MAC address of the Tag

Example:

```
{
    "TagId": "60C0BF209622"
}
```

Status of Fingerprinting Tags (GET /fingerprintings)

Fingerprinting status (list of all tags that are currently fingerprinting) can be easily retrieved by issuing a GET request to the platform against the */fingerprintings* endpoint.

For each tag that is actively fingerprinting following data will be returned:

- Tag Id
- Location Id
- Hardware Id
- Start
- End
- Fingerprinting Duration (calculates the duration between Start and the current timestamp)
- Received Broadcasts (number of broadcasts during the fingerprinting session)

Example:


```
{
        "TagId": "60C0BF209622",
        "LocationId": 11,
        "HardwareId": 1,
        "Start": "2020-10-23T12:18:17",
        "End": null,
        "FingerprintingDuration": "00:01:08.1382882",
        "ReceivedBroadcasts": 13345
    }
```

List of Fingerprinting Sessions Per Location (GET /fingerprintings/{LocationId})

List of fingerprinting sessions per location (list of all tags that were fingerprinting in the past and are now) can be easily retrieved by issuing a GET request to the platform against the */fingerprintings/{LocationId}* endpoint, where the LocationID is Location's Blyott internal identifier (see Locations).

For each tag following data will be returned:

- Tag Id
- Hardware Id
- Start
- End
- Received Broadcasts (number of broadcasts during the fingerprinting session)

Example:


```
{
    "TagId": "60C0BF209622",
    "HardwareId": 1,
    "Start": "2020-10-22T13:03:58",
    "End": "2020-10-23T12:17:44",
    "ReceivedBroadcasts": 12730
},
{
    "TagId": "60C0BF209623",
    "HardwareId": 1,
    "Start": "2020-10-23T12:18:17",
    "End": null,
    "ReceivedBroadcasts": 15
}
```

Delete Fingerprintings from Location (POST /deleteFingerprintings)

Fingerprintings are deleted from a location by issuing a POST request to the platform against the /deleteFingerprintings endpoint with a body of JSON payload consisting of only one parameter.

Parameter:

1. LocationId - Internal Blyott Location identifier

Example:

```
{
    "LocationId": 3
}
```

WORKFLOW SPECIFIC OPERATIONS:

Create Workflow (POST /addWorkflow)

Workflow is created by issuing a POST request to the platform against the */addWorkflow* endpoint with a body of JSON payload consisting of at least the following parameters:

1. Name - Name of the Workflow.
2. URL - URL where information will be forwarded to.
3. AuthenticationType - the authentication type specifies the security protocol. Currently, only 0 (*secret*) is possible.
4. Secret - used for accessing the URL.
5. ForwardType - defines when the trigger will be triggered (0 - instantly, 1 - every *x* seconds).
6. ForwardEvery - if ForwardType is 1, this field needs to be populated with the number of seconds (value should be larger than 1).
7. IsActive - *false*/ *true*, defines is workflow currently active or not.

Curl Example:

```
curl 'https://api.blyott.com/addWorkflow' \
  -H 'authority: api.blyott.com' \
  -H 'accept: application/json, text/plain, */*' \
  -H 'token: {token}' \
  -H 'content-type: application/json' \
  -H 'origin: https://portal.blyott.com' \
```

```
  -H 'sec-fetch-site: same-site' \
  -H 'sec-fetch-mode: cors' \
  -H 'sec-fetch-dest: empty' \
  -H 'referer: https://portal.blyott.com/' \
  -H 'accept-language: en-US,en;q=0.9' \
  --data-binary $'{"Name":"Workflow
1","URL":"https://test.com","AuthenticationType":0,"Secret":"Test123\u0021","ForwardType":1,"F
orwardEvery":160,"IsActive":true}' \
```

Get Workflow Details (GET /workflow/{WorkflowID})

Workflow details can be easily retrieved by issuing a GET request against the */workflow/{WorkflowID}* endpoint where the WorkflowID is Workflow Blyott internal identifier.

Curl Example:

```
curl 'https://api.blyott.com/workflow/4' |
  -H 'authority: api.blyott.com' |
  -H 'accept: application/json, text/plain, */*' |
  -H 'token: {token}'" |
  -H 'origin: https://portal.blyott.com' |
  -H 'sec-fetch-site: same-site' |
  -H 'sec-fetch-mode: cors' |
  -H 'sec-fetch-dest: empty' |
  -H 'referer: https://portal.blyott.com/' |
  -H 'accept-language: en-US,en;q=0.9' |
```

Edit Workflow (PUT /editWorkflow)

Workflow properties can be edited by issuing a PUT request against the */editWorkflow* endpoint with the body consisting of with a body of JSON payload consisting of at least the following parameters:

1. Name - Name of the Workflow.
2. URL - URL where information will be forwarded to.
3. AuthenticationType - the authentication type specifies the security protocol. Currently, only 0 (*secret*) is possible.
4. Secret - used for accessing the URL.
5. ForwardType - defines when the trigger will be triggered (0 - instantly, 1 - every *x* seconds).
6. ForwardEvery - if ForwardType is 1, this field needs to be populated with the number of seconds (value should be larger than 1).
7. IsActive - *false*/ *true*, defines is workflow currently active or not.

Curl Example:

```
curl "https://api.blyott.com/editWorkflow" ^
  -X "PUT" ^
  -H "authority: api.blyott.com" ^
  -H "accept: application/json, text/plain, */*" ^
  -H "token: {token}" ^
  -H "content-type: application/json" ^
  -H "origin: https://portal.blyott.com" ^
  -H "sec-fetch-site: same-site" ^
  -H "sec-fetch-mode: cors" ^
  -H "sec-fetch-dest: empty" ^
  -H "referer: https://portal.blyott.com/" ^
  -H "accept-language: en-US,en;q=0.9" ^
  --data-binary $'{"Id":4,"Name":"Workflow
2","URL":"https://test.com","AuthenticationType":0,"Secret":"Test123\u0021","ForwardType":1,"
ForwardEvery":180,"IsActive":true}' |
```

Search Workflow (POST /workflows)

To search for specific Workflow POST request to the */workflows* endpoint should be issued with following Filtering properties in the JSON body payload:

1. Page - Since results are paginated by 20 items, the page of the required result set should be provided. If the field is not provided all results will be returned.
2. SortBy - By which Workflow property should the results be sorted.
3. PageSize - If there are more than 20 results, specify this filter value.
    d. if not specified, number of items per page is 20.
    e. if number is less than 20, it is interpreted as 20.
    f. if number is greater than 1000, it is interpreted as 1000.
4. GlobalSearch - Searches the content in almost all grid columns.
5. SortOrder - Sorting direction can be:
    a. Asc - Ascending order.
    b. Desc - Descending order.
6. Filters - Arrays of applied filter, contains the list of following elements:
    a. FilterBy - Property by which to filter on.
    b. FilterContent - Array of values to search the value requested property against based on the contains type of logic.

Possible Filter properties:

1. Name - filter by Workflow Name.
2. URL - filter by URL.
3. AuthenticationType - filter by Authentication Type.
4. Secret - filter by secret.
5. ForwardType - filter by Forward Type.
6. ForwardEvery - filter by Forward Every.
7. IsActive - filter by Is Active (*false*/ *true).*

Curl Example:

```
curl 'https://api.blyott.com/workflows' \
  -H 'authority: api.blyott.com' \
  -H 'accept: application/json, text/plain, */*' \
  -H 'token: {token}' \
  -H 'content-type: application/json' \
  -H 'origin: https://portal.blyott.com' \
  -H 'sec-fetch-site: same-site' \
  -H 'sec-fetch-mode: cors' \
```

```
  -H 'sec-fetch-dest: empty' \
  -H 'referer: https://portal.blyott.com/' \

  -H 'accept-language: en-US,en;q=0.9' \
  --data-binary
'{"Page":1,"PageSize":25,"GlobalSearch":"","SortBy":"Name","SortOrder":"Asc","Filters":[{"FilterB
y":"Name","FilterName":"Workflow","FilterContent":["W"]},{"FilterBy":"URL","FilterName":"URL","
FilterContent":["https"]},{"FilterBy":"AuthenticationMethod","FilterName":"Authentication
Method","FilterContent":["Secret"]},{"FilterBy":"Secret","FilterName":"Secret","FilterContent":["T
est"]},{"FilterBy":"ForwardInfo","FilterName":"ForwardInfo","FilterContent":["Every"]},{"FilterBy":"
Active","FilterName":"Active","FilterContent":["Yes"]}]}' \
```