Service Ledger APIs documentation

A company in SL can be represented by creating a group `organisation`. In SL, there are two roles for an entity belonging to an `organisation`:

- `admin` : each `organisation` has only one admin, which is responsible for registering users and creating TAXII apiroot and collection.
- `user` : an `organisation` has multiple users, which can retrieve information from SL or store STIX objects to it.

Authentication Server

The *Authentication Server* of SL, listening on port 6023, exposes the following APIs:

URL	Methods	Authentication	Authorised Roles for POST
/v1/signup/admin	POST	no	none
/v1/signup/**{organisation}**/user	POST	yes	`admin`
/v1/**{organisation}**/admin/algo_account	GET	yes	`admin`
/v1/login	POST	yes	none

The "none" value means that no specific role is required to POST on the requested endpoint. In the 1st URL this means, since there is no authentication, that anyone can register an organisation (and its admin) into SL. The authentication performed by the other URLs are different. The 2nd and 3rd URLs authenticate via *Bearer Token*, whereas the 4th one via username/password.

Registering an organisation and its admin

```
curl -sk --request POST \
    --header "Content-Type: application/json" \
    --data '{"org_name": "hospital", "username":
"hospital_admin","password": "hosadm"}' \
    https://localhost:6023/v1/signup/admin | jq -r "."
```

This API creates an organisation named `hospital` in SL and associate to it the admin `hospital_admin`. Furthermore, it automatically creates for the organisation:

- a pair (pk, sk) of cryprographic keys, stored in the HashiCorp Vault datastore, to encrypt/decrypt STIX objects;
- an Algorand account to generate NFTs. The account's public address and list of secret mnemonic words are returned in response. The account's private key is securely stored on the HashiCorp Vault datastore.

TODO. The `account_address` returned by this API **MUST BE FUNDED** at the Algorand Dispenser.

Sample response


```
{
  "organisation": {
    "name": "hospital",
    "id": "7bf5e454-e377-da0d-bc31-12fb4bce0e7e",
    "creation_time": "2023-03-27T19:12:22.781488856Z",
    "policies": [
      "hospital"
    ],
    "vault": {
      "message": "A pair (pk, sk) of cryptographic keys created to encrypt
and decrypt STIX objects"
    },
    "algorand": {
      "account_address":
"XKM6SS5JHHAEJICZEWGPDLNIKOVHUKNMKRFTOJFM6YSLYXQDO5NUEHLA5A",
      "account_mnemonic": "puzzle abstract erosion valid infant gallery
attack eager drill thing enlist satisfy mutual section chaos paddle
disorder human hammer sphere area sound girl abstract mind"
    }
  },
  "entity": {
    "name": "hospital_admin",
    "id": "919730fd-4c92-497d-4fe7-cc8ae56154ed",
    "metadata": {
      "organisation": "hospital",
      "role": "admin"
    },
    "policies": [
      "default",
      "sl-admins"
    ],
    "creation_time": "2023-03-27T19:12:23.080935752Z",
    "group_ids": [
      "7bf5e454-e377-da0d-bc31-12fb4bce0e7e"
    ]
  }
}
```


Registering an organisation by importing an Algorand account

```
curl -sk --request POST \
    --header "Content-Type: application/json" \
    --data '{"org_name": "hospital",  "username":
"hospital_admin","password": "hosadm", "algorand_account_mnemonic":
"chimney exchange gold ramp green find abandon pact tragic little jacket
winter receive crew spot blush deal glance verb can purity adjust real
above trip"}' \
    https://localhost:6023/v1/signup/admin | jq -r "."
```

This API is the same of the previous one but allows to insert an already existing Algorand account.

Login

```
curl -sk --request POST \
    --header "Content-Type: application/json" \
    --data '{"username": "hospital_admin","password": "hosadm"}' \
    https://localhost:6023/v1/login | jq -r "."
```

This API allows an organisation's entity to login in SL, e.g., the admin `hospital_admin` in this case. The field `sl-token` of the response contains the token that needs to be passed for authentication in other API calls.

Sample response

```
{
  "name": "hospital_admin",
  "id": "919730fd-4c92-497d-4fe7-cc8ae56154ed",
  "metadata": {
    "organisation": "hospital",
    "role": "admin"
  },
  "sl_token": "01GWJ7S71QJDAHSX5E3X097ASN"
}
```

Retrieving the Algorand account address of an organisation

```
curl -sk --request GET \
    --header "Authorization: Bearer ${ADMIN_TOKEN}" \
    https://localhost:6023/v1/${ORG_NAME}/admin/algo_account | jq -r "."
```

This API allows an admin to retrieve the Algorand account address for its `ORG_NAME` organisation. Note that this API includes a Bearer token `ADMIN_TOKEN` for authentication, i.e. a token obtained from a login API in the field `sl_token`.

Sample response

```
{
  "algorand": {
      "account_address":
"XKM6SS5JHHAEJICZEWGPDLNIKOVHUKNMKRFTOJFM6YSLYXQDO5NUEHLA5A"
  }
```

```
}
```

Adding Users to the organisation

```
curl -sk --request POST \
    --header "Content-Type: application/json" \
    --header "Authorization: Bearer ${ADMIN_TOKEN}" \
    --data '{"username": "hospital_user1","password": "password"}' \
    https://localhost:6023/v1/signup/${ORG_NAME}/user | jq -r "."
```

This API registers a user `hospital_user1` in the `ORG_NAME` organisation, i.e. `hospital` in this case. Note that this API includes a Bearer token `ADMIN_TOKEN` for authentication, i.e. a token obtained from a login API in the field `sl_token`.

Sample response


```
{
  "entity": {
    "name": "hospital_user1",
    "id": "326b9e1e-75c0-5244-4026-722428354816",
    "metadata": {
      "organisation": "hospital",
      "role": "user"
    },
    "policies": [
      "default",
      "hospital-users",
      "sl-users"
    ],
    "creation_time": "2023-03-27T22:52:04.933286479Z",
    "group_ids": [
      "7bf5e454-e377-da0d-bc31-12fb4bce0e7e"
    ]
  }
}
```


TAXII Server

The *TAXII Server* of SL, listening on port 6023, exposes the following APIs:

URL	Methods	Authentication	Authorised Roles for POST
/taxii2	GET	no	none

URL	Methods	Authentication	Authorised Roles for POST
/{apiroot}	GET POST	yes	`admin`
/{apiroot}/status/**{status-id}**	GET	yes	none
/{apiroot}/collections	GET POST	yes	`admin`
/{apiroot}/collections/**{col-id}**	GET	yes	none
/{apiroot}/collections/**{col-id}**/manifest	GET	yes	none
/{apiroot}/collections/**{col-id}**/objects	GET POST	yes	`admin` `user`
/{apiroot}/collections/**{col-id}**/objects/**{obj-id}**	GET	yes	none
/{apiroot}/collections/**{col-id}**/objects/**{obj-id}**/versions	GET	yes	none

In the TAXII Server both `admin` and `user` need to authenticate to GET/POST on all URLs, except for */taxii*. When `admin` and `user` request an **{apiroot}**, they are both authorised to GET information from its following URLs (e.g., ***/{apiroot}****/collections*) only if they belong to the **{apiroot}** organisation. Only the organisation's `admin` is authorised to POST an apiroot or a collection. Both `admin` and `user` are authorised to POST a STIX object.

STIX objects filtering

The TAXII Server provides STIX objects filtering via URL query, by adding a list of parameters after the question mark `?`, with each parameter linked to the other by the symbol `&`.

The TAXII Server allows the following URL query parameters:

URL query parameter	Type	Number of occurences in query	Description
`added_after`	timestamp	1	Return only those STIX objects added after the specified timestamp.
`limit`	integer	1	Maximum number of STIX objects that the client would like to receive in response. Must be a positive integer greater than zero. If it is lesser than the number of STIX objects retrieved from the requested SL's URL, it returns the earlier. Otherwise, it returns all STIX objects retrieved from the SL's URL.
`match[id]`	string	N	The identifier of the STIX object(s) that are being requested.

URL query parameter	Type	Number of occurences in query	Description
`match[type]`	string	N	The type of the STIX object(s) that are being requested. The allowed types are defined in STIX documentation.
`match[version]`	`first` or `last` or timestamp	N	The version(s) of the STIX object(s) that are being requested, where `first` is the earliest version, `last` is the latest, and timestamp in ISO 8601 format exactly matches the creation date of the STIX object in SL. The keywords `first` and `last` cannot occur more than once in a query, whereas multiple different specific timestamps may be specified in the same query.

The filtering of STIX objects is enabled in the TAXII Server for the following APIs:

URL	Methods	Query parameters allowed
/{apiroot}/collections/**{col-id}**/manifest	GET	`added_after` `limit` `match[id]` `match[type]` `match[version]`
/{apiroot}/collections/**{col-id}**/objects	GET	`added_after` `limit` `match[id]` `match[type]` `match[version]`
/{apiroot}/collections/**{col-id}**/objects/**{obj-id}**	GET	`added_after` `limit` `match[version]`
/{apiroot}/collections/**{col-id}**/objects/**{obj-id}**/versions	GET	`added_after` `limit`

Creating a TAXII apiroot for the organisation

```
curl -sk --request POST \
    --header "Content-Type: application/json" \
    --header "Authorization: Bearer ${ADMIN_TOKEN}" \
    --data '{"title": "The Hospital","description": "Hospital Group"}' \
    https://localhost:6023/${ORG_NAME} | jq -r "."
```

This API creates a TAXII apiroot called `ORG_NAME` for the organisation `ORG_NAME` (e.g, `hospital` in this example). Beyond the name equality, note that there is 1-to-1 relation between an apiroot and an organisation.

Sample response

```
{
  "apiroot": {
    "id": "aa18d7d5f30e45abc6b4a19b8abc0018",
    "title": "The Hospital",
    "description": "Hospital Group",
    "versions": [
      "application/taxii+json;version=2.1"
    ],
    "max_content_length": 104857600,
    "domain": "https://localhost:6023",
    "slug": "hospital",
    "discoveryId": "taxii2"
  }
}
```


Discovering the TAXII Server information

```
curl -sk --request GET \
    --header "Accept: application/json" \
    https://localhost:6023/taxii2 | jq -r "."
```

This API returns the TAXII Server information and the domain of each offered apiroot.

Sample response


```
{
  "title": "TAXII V2.1 Implementation",
  "description": "This is the discovery listing all available api roots",
  "contact": "University of Southampton",
  "api_roots": [
    "https://localhost:6023/hospital/"
  ]
}
```


Retrieving the apiroot information

```
curl -sk --request GET \
    --header "Accept: application/json" \
    --header "Authorization: Bearer ${USER_TOKEN}" \
    https://localhost:6023/${ORG_NAME} | jq -r "."
```

This API returns information about the `${ORG_NAME}` apiroot, e.g. `hospital` in this example.

Sample response


```
{
  "title": "The Hospital",
  "description": "Hospital Group",
  "versions": [
    "application/taxii+json;version=2.1"
  ],
  "max_content_length": 104857600
}
```


Creating a TAXII collection

```
curl -sk --request POST \
    --header "Content-Type: application/json" \
    --header "Authorization: Bearer ${ADMIN_TOKEN}" \
    --data '{"title": "Attacks","alias": "attack"}' \
    https://localhost:6023/${ORG_NAME}/collections | jq -r "."
```

This API creates a TAXII collection called `attack` under the `${ORG_NAME}` apiroot.

Sample response


```
{
  "collection": {
    "id": "a826e0d8-b128-49b3-950b-28455c4b9a37",
    "title": "Attacks",
    "description": null,
    "alias": "attack",
    "media_types": [
      "application/taxii+json;version=2.1"
    ],
    "can_read": true,
    "can_write": true,
    "apiRootId": "aa18d7d5f30e45abc6b4a19b8abc0018"
  }
}
```


Retrieving all collections of the apiroot

```
curl -sk --request GET \
    --header "Accept: application/json" \
```

```
    --header "Authorization: Bearer ${USER_TOKEN}" \
    https://localhost:6023/${ORG_NAME}/collections/ | jq -r "."
```

This API retrieves all collections under the `${ORG_NAME}` apiroot, e.g. `hospital` in this example.

Sample response


```
{
  "collections": [
    {
      "id": "3b0d26b1-63ba-4199-b3a0-102c877969dc",
      "title": "Attacks",
      "can_read": true,
      "can_write": true,
      "alias": "attack",
      "media_types": [
        "application/taxii+json;version=2.1"
      ]
    }
  ]
}
```


Retrieving a specific collection

```
curl -sk --request GET \
    --header "Accept: application/json" \
    --header "Authorization: Bearer ${USER_TOKEN}" \
    https://localhost:6023/${ORG_NAME}/collections/${COL_ID} | jq -r "."
```

This API retrieves all the information related to the `${COL_ID}` collection, e.g. `attack` in this example.

Sample response


```
{
  "id": "3b0d26b1-63ba-4199-b3a0-102c877969dc",
  "title": "Attacks",
  "can_read": true,
  "can_write": true,
  "alias": "attack",
  "media_types": [
    "application/taxii+json;version=2.1"
  ]
}
```

Storing a STIX object

```
curl -sk --request POST \
    --header "Content-Type: application/json" \
    --header "Authorization: Bearer ${USER_TOKEN}" \
    --data @stix-obj-example.json \
    https://localhost:6023/${ORG_NAME}/collections/${COL_ID}/objects | jq -
r "."
```

This API stores a STIX object (i.e. the file `stix-obj-example.json`) into SL. The response contains the transaction ID of the NFT created on Algorand blockchain representing the STIX object.

Sample response


```
{
  "id": "70778cf1-1078-45cc-8598-2aa3e718b94e",
  "status": "complete",
  "request_timestamp": "2023-03-27T19:16:00.460Z",
  "total_count": 1,
  "success_count": 1,
  "successes": [
    {
      "id": "bundle--2a25c3c8-5d88-4ae9-862a-cc3396442317",
      "version": "2023-03-27T19:16:00.304Z",
      "message": "[Algorand blockchain] Transaction ID of the NFT
representing the STIX object stored on IPFS:
WXZWTMI7WFEE3TJEU5O2IDR6QGGJGCFRGUPC57T7ASTWG2RJPANA"
    }
  ],
  "failure_count": 0,
  "pending_count": 0
}
```


Discovering the status of a STIX object

```
curl -sk --request GET \
    --header "Accept: application/json" \
    --header "Authorization: Bearer ${USER_TOKEN}" \
    --data @stix-obj-example.json \
    https://localhost:6023/${ORG_NAME}/status/${STATUS_ID} | jq -r "."
```

This API returns the request status of a STIX object creation, by specifying its associated `${STATUS_ID}` in the URL endpoint, e.g. `70778cf1-1078-45cc-8598-2aa3e718b94e` in this example. The response of this API is the same of the one for storing a STIX object.

Sample response

```
{
  "id": "70778cf1-1078-45cc-8598-2aa3e718b94e",
  "status": "complete",
  "request_timestamp": "2023-03-27T19:16:00.460Z",
  "total_count": 1,
  "success_count": 1,
  "successes": [
    {
      "id": "bundle--2a25c3c8-5d88-4ae9-862a-cc3396442317",
      "version": "2023-03-27T19:16:00.304Z",
      "message": "[Algorand blockchain] Transaction ID of the NFT
representing the STIX object stored on IPFS:
WXZWTMI7WFEE3TJEU5O2IDR6QGGJGCFRGUPC57T7ASTWG2RJPANA"
    }
  ],
  "failure_count": 0,
  "pending_count": 0
}
```


Retrieving all STIX objects of a collection

```
curl -sk --request GET \
    --header "Accept: application/json" \
    --header "Authorization: Bearer ${USER_TOKEN}" \
    https://localhost:6023/${ORG_NAME}/collections/${COL_ID}/objects | jq -
r "."
```

This API returns all STIX objects under the `${COL_ID}` collection, e.g. `attack` in this example.

Sample response


```
{
  "objects": [
    {
      "type": "bundle",
      "id": "bundle--2a25c3c8-5d88-4ae9-862a-cc3396442317",
    ...
    }
  ]
}
```


Filtering STIX objects of a collection

```
curl -skg --request GET \
    --header "Accept: application/json" \
```

```
    --header "Authorization: Bearer ${USER_TOKEN}" \
    'https://localhost:6023/${ORG_NAME}/collections/${COL_ID}/objects?
added_after=2023/01/01&limit=2&match[type]=malware' | jq -r "."
```

This API returns a maximum of 2 STIX objects, added in SL after 01/01/2023, of type `malware`, stored under the `${COL_ID}` collection, e.g. `attack` in this example.

Sample response

```
{
  "objects": [
    ...
  ]
}
```

Retrieving a manifest of the STIX objects metadata

```
curl -sk --request GET \
    --header "Accept: application/json" \
    --header "Authorization: Bearer ${USER_TOKEN}" \
    https://localhost:6023/${ORG_NAME}/collections/${COL_ID}/manifest | jq
-r "."
```

This API returns a collection manifest, that is all the STIX objects metadata under the `${COL_ID}` collection, e.g. `attack` in this example.

Sample response

```
{
  "objects": [
    {
      "id": "bundle--2a25c3c8-5d88-4ae9-862a-cc3396442317",
      "date_added": "2023-03-30T19:06:19.918Z",
      "version": "2023-03-30T19:06:19.918Z",
      "media_type": "application/taxii+json;version=2.1"
    }
  ]
}
```

Filtering a manifest of STIX objects

```
curl -skg --request GET \
    --header "Accept: application/json" \
    --header "Authorization: Bearer ${USER_TOKEN}" \
```

```
    'https://localhost:6023/${ORG_NAME}/collections/${COL_ID}/manifest?
match[type]=malware&match[type]=indicator&match[version]=first&match[versio
n]=last' | jq -r "."
```

This API returns the first and last STIX objects of type either `malware` or `indicator`, stored under the `${COL_ID}` collection, e.g. `attack` in this example.

Sample response

```
{
  "objects": [
    ...
  ]
}
```

Retrieving a specific STIX object

```
curl -sk --request GET \
    --header "Accept: application/json" \
    --header "Authorization: Bearer ${USER_TOKEN}" \
    https://localhost:6023/${ORG_NAME}/collections/${COL_ID}/objects/${STIX
_OBJ_ID} | jq -r "."
```

This API returns the `${STIX_OBJ_ID}` STIX object, e.g. `bundle--2a25c3c8-5d88-4ae9-862a-cc3396442317` in this example.

Sample response

```
{
  "objects": [
    {
      "type": "bundle",
      "id": "bundle--2a25c3c8-5d88-4ae9-862a-cc3396442317",
    ...
    }
  ]
}
```

Filtering a specific STIX object

```
curl -skg --request GET \
    --header "Accept: application/json" \
    --header "Authorization: Bearer ${USER_TOKEN}" \
```

```
    'https://localhost:6023/${ORG_NAME}/collections/${COL_ID}/objects/${STI
X_OBJ_ID}?match[version]=last' | jq -r "."
```

This API returns the last version of the `${STIX_OBJ_ID}` STIX object, e.g. `bundle--2a25c3c8-5d88-4ae9-862a-cc3396442317` in this example.

Sample response

```
{
  "objects": [
    ...
  ]
}
```

Retrieving versions of a STIX object

```
curl -sk --request GET \
    --header "Accept: application/json" \
    --header "Authorization: Bearer ${USER_TOKEN}" \
    https://localhost:6023/${ORG_NAME}/collections/${COL_ID}/objects/${STIX
_OBJ_ID}/versions | jq -r "."
```

This API returns the versions of a specified `${STIX_OBJ_ID}` STIX object, e.g. `bundle--2a25c3c8-5d88-4ae9-862a-cc3396442317` in this example. In SL, a STIX object can not be deleated because its NFT is permanently stored on the Algorand blockchain. Each version of a STIX object has an associated NFT.

Sample response

```
{
  "versions": [
    "2023-03-27T19:16:00.304Z",
    "2023-03-27T19:22:37.938Z"
  ]
}
```

Filtering versions of a STIX object

```
curl -skg --request GET \
    --header "Accept: application/json" \
    --header "Authorization: Bearer ${USER_TOKEN}" \
    'https://localhost:6023/${ORG_NAME}/collections/${COL_ID}/objects/${STI
X_OBJ_ID}/versions?limit=3' | jq -r "."
```

This API returns a maximum of 3 versions of the `${STIX_OBJ_ID}` STIX object, e.g. `bundle--2a25c3c8-5d88-4ae9-862a-cc3396442317` in this example.

Sample response

```
{
  "versions": [
    ...
  ]
}
```