ThingToData CURL ThingStories

Table of Contents

Metadata API (APIs without Session)

These APIs can be called without session. These two APIs almost all applications should use.

EnumNames

Returns all enum names.

```
curl -X GET --header 'Accept: application/json'
'http://localhost:27122/api/Metadata/EnumNames'
```

EnumValues

Returns values of the enum.

Example: Role enum values.

```
curl -X GET --header 'Accept: application/json'
'http://localhost:27122/api/Metadata/EnumValues?enumName=Role'
```

Authentication

Two stories; Authentication and Anonymous. SessionID will be used in all of the following APIs.

Note: Currently only SessionId is returned. In the next versions there will be SessionId and a Secret (probably in JWT format).

Session concept

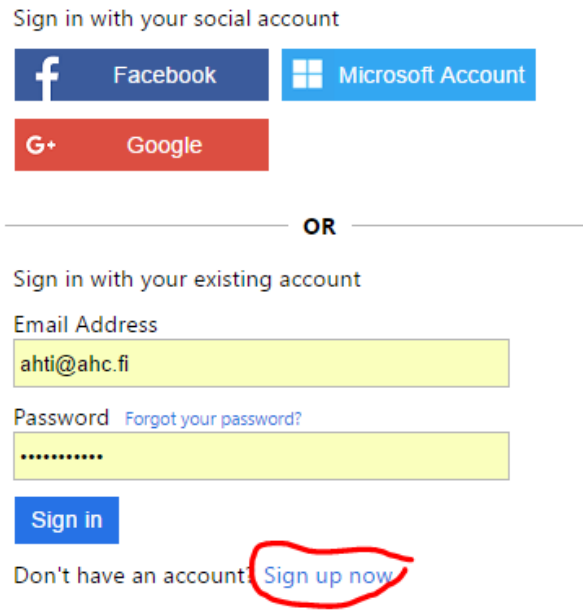
Almost all APIs requires session Id (and a session secret in the future). The exception of this is Metadata API an GetNearbyPublicLocationThings.

Session is either Authenticated Session or Anonymous Session. All things that are accessed during session will be added into Session Access list. When you access any data (create or query attribute or relation) you have to specify in which role this data is accessed. At least one of the things in your session must have that access right for this data.

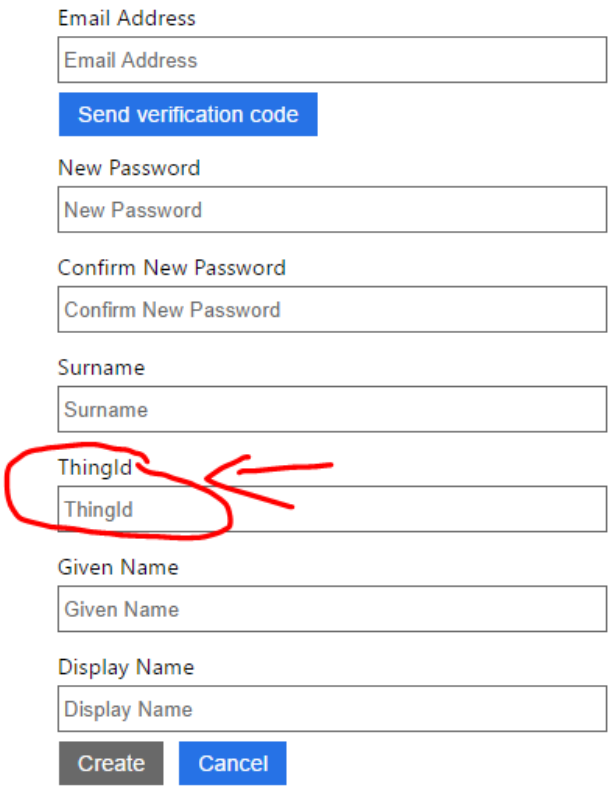
Authenticated Session

You have to create a account first. Go to web site https://thingtodataUI.azurewebsites.net and click “Sign in”.

If you are using social media accounts (and are already signed in into it), click that icon. If you want to create a new custom account, click “Sign up now”



In both cases, set ThingId. It's important, that will be your AuthenticationThing. That thing do not have to be created into Inventory yet, EnterAuthenticatedSession will create it if not already created.



After correctly signed if (or Signed up), you can find out your token by clicking ShowToken. That is only for testing purposes, normally client app just uses this JWT token. Copy to token to the clipboard.

Access Token

Copy this token to clipboard:

eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6IlIg1ZVhrNHh5b2pORnVtMWtsMll0djhkbE5QNC1jNTdkTzZRR1RWQndhTmsifQ.eyJpc3MiOiJodHRwczovL2xvZ2luLm1pY3Jvc2
SmUQz9wTAaPIvZEXMwon6taWRXLR9GAxcAF318t1Dw5ErU0cf3MhBVioONpIC_tCDgIX0dNXmJd1QYf11AEgahY_sBzqQ29JYYCxzCviX9XD2dBLNfTXUHZ9c6ihyyD2o4xzYCb0
zYJjbQ1ZG7fgnajrlgUULdnAzavNnw1TarG9ZlUd7L-MbvjugkJN2shijLUSHnULxnr401Jb9cWa-Q2GUB2zUpk0gyfPFAFKGCutVoSAIMs9Rf25FMOw

Claims of this JWT token

Type	Value
iss	https://login.microsoftonline.com/2aea2973-b456-4a6b-9985-5f88074ec570/v2.0/
exp	1496892762
nbf	1496889162
aud	16c9a9fa-f85e-4f8d-b968-ac1b55e98c17
oid	6e8cac27-ed30-4d49-bdc9-c00bf8fa3dbe
sub	6e8cac27-ed30-4d49-bdc9-c00bf8fa3dbe
family_name	Haukilehto
extension_ThingId	inv1.sovelto.fi/Ahti2
given_name	Ahti
name	Ahti (b2c local)
emails	ahti@ahc.fi
tfp	B2C_1_SignUpOrSignIn
nonce	636324857334533307.ZmNlN2YxNDItMjg0Zi00OTBjLThkYjEtMmM3YTk2YmFlYzliMmRmYzFjNzMtNzQ3NC00YjRkLWFiYTktMDIxYmNiMGVmNjlh
scp	read
azp	d49e02a6-36cd-4a58-b19f-c7602d8e61f4
ver	1.0

If you want to change AutentecationThing that is connected to your account, click “Change ThingId for ...” and provide a new ThingId. You have to sign out / Sign in again to get a new JWT which have this new ThingId!

And then get SessionId (and in the future also SessionSecret)

```
curl -X POST --header 'Accept: application/json' --header
'Authorization: Bearer eyJ0eXAiOi...'
'http://localhost:27122/api/inventory/Authentication/EnterAuthenticate
dSession'
```

Save the response, you’ll need it in the following calls.

Note: in the test environment, there is a Session id: **00000000-0000-0000-0000-000000000001** (and role Omnipotent) which can always be used. Using this test session and role, all functions will work.

Note: in the test environment, this call will always succeed (ThingID in the JWT must be in valid format). Authentication Thing will be created if it does not exist.

Note: This call will use normal OAuth2 token in future.

Anonymous Session

```
curl -X POST --header 'Accept: application/json'
'http://localhost:27122/api/inventory/Authentication/EnterAnonymousSes
sion'
```

Save the response, you'll need it in the following calls.

Note: in the test environment, there is a Session id: **00000000-0000-0000-0000-000000000002** (and role Anonumous) which can always be used.

Core API, Create a thing, access attributes

Something of the concepts:

1. **Role**: Roles are used for access rights. Rights are set for each role. Roles are predefined, you cannot create new roles. You can just give rights for each Role and set Role Members to each Role.
2. **Relation**: Relation are used to navigate from one thing to another. Relations are predefined, you cannot create new relation names. Relations do have direction and most of them are in pairs (like Contains and ContainedBy)
3. Relation **RoleIn** is a special role. The Thing can require, that Inventory adds automatically Another Thing into this role whenever The Thing has a role in Another Thing. Note: this functionality is not available in current test environment.
4. **Role parameter** in API calls: at least one of the Things in a Session (Session contains all of the things we have accessed during the session) must be in this role and this role must have enough Access Rights to perform the function
5. **ThingId**: Thing id is formed by FQDN/US
    - **FQDN** (Fully Qualified Domain Name)
    - **/** FQDN is followed by slash (/)
    - **US (**Unique String) can contain any characters

Create a thing and set some of its attributes

We'll create a new thing (ThingID inv1.sovelto.fi/NewThing#001) and set some of its attributes (IsGpsPublic, Gps Location and Archetype is `inv1.sovelto.fi/SolidTrowel`)

1. EnterAnonymousSession (we'll use SessionId 00000000-0000-0000-0000-000000000001 in the following calls)

```
2. curl -X POST --header 'Content-Type: application/json' -d '{ \
"newThingId": "inv1.sovelto.fi/NewThing#001", \
"title": "Example of a new Thing", \
"thingType": "RegularThing", \
"session": "00000000-0000-0000-0000-000000000001 ", \
"thingId": "inv1.sovelto.fi/M100", \
"role": "Omnipotent" \
}' 'http://localhost:27122/api/inventory/Core/CreateLocalThing'
```

Note:

- **newThingId** must be a new unique thing in this Inventory. Currently test Inventory allows all FQDN:s, but obviously this will change in production Inventories.
- **thingId** is the thing that will have Omnipotent role for the new Thing
- **role** the Role in which thingId is accessed. Obviously this role must have Create RoleRight

```
   3. curl -X POST --header 'Content-Type: application/json' --header
      'Accept: application/json' -d '{ \
   "attributeValues": [ \
     { \
       "attribute": "Description", \
       "value": "This is description of the Thing" }, \
      { \
       "attribute": "ArchetypeThingId", \
       "value": "inv1.sovelto.fi/SolidTrowel" \
    }, \
     { \
       "attribute": "IsGpsPublic", \
       "value": true \
     }, \
{ \
 "attribute": "Location_Gps", \
       "value": { \
     "latitude": 62.78875546595712, \
     "longitude": 22.846313826953065 \
   } \
 } \
   ], \
   "session": "00000000-0000-0000-0000-000000000001 ", \
   "thingId": "inv1.sovelto.fi/NewThing#001", \
   "role": "Omnipotent" \
 }' 'http://localhost:27122/api/inventory/Core/SetAttributes'
```

Testing

1. inv1.sovelto.fi/M100 should have Omnipotent role for this new thing.

```
curl -X POST --header 'Content-Type: application/json' --header
'Accept: application/json' -d '{ \
   "roleForMemberList": "Omnipotent", \
   "session": "00000000-0000-0000-0000-000000000001 ", \
   "thingId": "inv1.sovelto.fi/NewThing#001", \
   "role": "Omnipotent" \
 }' 'http://localhost:27122/api/inventory/Core/GetRoleMemberList'
```

Returns:

```
{
  "thingIds": [
    "inv1.sovelto.fi/M100"
  ]
}
```

2. Get Attributes should have those values we have set previously

```
curl -X POST --header 'Content-Type: application/json' --header
'Accept: application/json' -d '{ \
   "attributes": [ \
     "Title", \
 "Description", \
"ArchetypeThingId",\
 "IsGpsPublic", \
 "GpsLocation" \
   ], \
   "session": "00000000-0000-0000-0000-000000000001", \
   "thingId": "inv1.sovelto.fi/NewThing#001", \
   "role": "Omnipotent" \
 }' 'http://localhost:27122/api/inventory/Core/GetAttributes'
```

Returns:

```
  "attributeValues": [
    {
      "attribute": "Title",
      "isOk": true,
      "errorDescription": null,
      "value": "Example of a new Thing"
    },
    {
      "attribute": "Description",
      "isOk": true,
      "errorDescription": null,
      "value": "This is description of the Thing"
    },
    {
      "attribute": "ArchetypeThingId",
      "isOk": true,
      "errorDescription": null,
      "value": "inv1.sovelto.fi/SolidTrowel"
    },
    {
      "attribute": "IsGpsPublic",
      "isOk": true,
      "errorDescription": null,
      "value": true
    },
```

```
    {
      "attribute": "GpsLocation",
      "isOk": false,
      "errorDescription": "Attribute do not exists.",
      "value": null
    }
  ]
}
```

Extension attribute

Extension attribute is named in the same way as a Thing (FQDN/US). There will be API RegisterExtensionType, but it is not implemented yet. Now you just set attribute using correctly formed extension attribute name.

Extension attribute data is just a string. The Client Application is fully responsible for all business logic of the data.

Access Rights are always for all whole Extension data (of particular type), you cannot specify that this and this role will have read access to only part of an extension data.

In this example we'll set and read some extension data. Extension type is inv1.sovelto.fi/MyExtension#001

```
curl -X POST --header 'Content-Type: application/json' --header
'Accept: application/json' -d '{ \
   "attributeValues": [ \
     { \
       "attribute": "inv1.sovelto.fi/MyExtension#001", \
       "value": "This is the value of this extension" } \
   ], \
   "session": "00000000-0000-0000-0000-000000000001 ", \
   "thingId": "inv1.sovelto.fi/NewThing#001", \
   "role": "Omnipotent" \
 }' 'http://localhost:27122/api/inventory/Core/SetAttributes'
```

And let's read the value

```
curl -X POST --header 'Content-Type: application/json' --header
'Accept: application/json' -d '{ \
   "attributes": [ \
     "inv1.sovelto.fi/MyExtension#001" \
   ], \
   "session": "00000000-0000-0000-0000-000000000001", \
   "thingId": "inv1.sovelto.fi/NewThing#001", \
   "role": "Omnipotent" \
 }' 'http://localhost:27122/api/inventory/Core/GetAttributes'
```

Result:

```
{
  "attributeValues": [
    {
      "attribute": "inv1.sovelto.fi/MyExtension#001",
      "isOk": true,
      "errorDescription": null,
      "value": "This is the value of this extension"
    }
  ]
}
```

Core API, Set Access Rights

Set Anonymous Read and Update access rights to a thing.

We'll set anonymous Read and Update access rights for Description attribute to the thing that we have created in the previous chapter. We'll use test SessionId 00000000-0000-0000-0000-000000000001, but in real life you should have a Session that have enough rights to the thing.

Note: Currently there is no access right checking in the test Inventory. But there will be :)

```
curl -X POST --header 'Content-Type: application/json' --header
'Accept: text/plain' -d '{ \
   "roleForRights": "Anonymous", \
   "attributeRoleRights": [ \
     { \
       "attribute": "Description", \
       "roleAccessRights": [ \
         "Read", "Update" \
       ] \
     } \
   ], \
   "session": "00000000-0000-0000-0000-000000000001", \
   "thingId": "inv1.sovelto.fi/NewThing#001", \
   "role": "Omnipotent" \
 }' 'http://localhost:27122/api/inventory/Core/SetRoleAccessRight'
```

Testing. Anonymous role should have Read/Update rights only for Description attribute

```
curl -X POST --header 'Content-Type: application/json' --header
'Accept: application/json' -d '{ \
   "roleForRights": "Anonymous", \
   "session": "00000000-0000-0000-0000-000000000001", \
   "thingId": "inv1.sovelto.fi/NewThing#001", \
   "role": "Omnipotent" \
```

```
}' 'http://localhost:27122/api/inventory/Core/GetRoleAccessRight'
```

Result:

```
{
  "attributeRoleRights": [
    {
      "attribute": "Description",
      "roleAccessRights": [
        "Read",
        "Update"
      ]
    }
  ]
}
```

Set Role, RoleMember and RoleAccessRight to a Thing.

We'll set thing **T1** into Owner role for **NewThing#001** and set Read/Update access rights for Description attribute to the Owner role.

1. Set access right to role Owner

```
curl -X POST --header 'Content-Type: application/json' --header
'Accept: text/plain' -d '{ \
   "roleForRights": "Owner", \
   "attributeRoleRights": [ \
     { \
       "attribute": "Description", \
       "roleAccessRights": [ \
         "Read", "Update" \
       ] \
     } \
   ], \
   "session": "00000000-0000-0000-0000-000000000001", \
   "thingId": "inv1.sovelto.fi/NewThing#001", \
   "role": "Omnipotent" \
}' 'http://localhost:27122/api/inventory/Core/SetRoleAccessRight'
```

2. Set T1 into Owner role.

```
curl -X POST --header 'Content-Type: application/json' --header
'Accept: text/plain' -d '{ \
   "roleForMemberList": "Owner", \
   "memberThingIds": [ \
     "inv1.sovelto.fi/T1" \
   ], \
   "session": "00000000-0000-0000-0000-000000000001", \
   "thingId": "inv1.sovelto.fi/NewThing#001", \
```

```
   "role": "Omnipotent" \
 }' 'http://localhost:27122/api/inventory/Core/SetRoleMemberList'
```

Testing

1. Only T1 should be in Owner role

```
curl -X POST --header 'Content-Type: application/json' --header
'Accept: application/json' -d '{ \
   "roleForMemberList": "Owner", \
   "session": "00000000-0000-0000-0000-000000000001 ", \
   "thingId": "inv1.sovelto.fi/NewThing#001", \
   "role": "Omnipotent" \
 }' 'http://localhost:27122/api/inventory/Core/GetRoleMemberList'
```

Result:

```
{
  "thingIds": [
    "inv1.sovelto.fi/T1"
  ]
}
```

2. Owner role should have read/update access rights

```
curl -X POST --header 'Content-Type: application/json' --header
'Accept: application/json' -d '{ \
   "roleForRights": "Owner", \
   "session": "00000000-0000-0000-0000-000000000001", \
   "thingId": "inv1.sovelto.fi/NewThing#001", \
   "role": "Omnipotent" \
 }' 'http://localhost:27122/api/inventory/Core/GetRoleAccessRight'
```

Result:

```
{
  "attributeRoleRights": [
    {
      "attribute": "Description",
      "roleAccessRights": [
        "Read",
        " Update"
      ]
    }
  ]
}
```

Core API, Navigation based on known Thing

Concepts

In this ThingStory, application user (we'll use name *user* from this point forward) knows ThingId (either reads it for example from QR Code or gets a thing by authentication (Authentication Thing)). From this thing user can navigate to other things by using GetRealations API. To access a thing user must specify a Role. QueryMyRoles returns all the roles current Session has for a thing.

Setting Relations

This chapter will be added later. There is no SetRelation API yet.

Navigation based on Relations

This chapter will be added later.

Navigation based on public location

GetNearbyPublicLocationThings returns all things that near specified location and have public location. In test material there is one Thing in Seinäjoki (T2), one Thing is PeräSeinäjoki (T1) and one in Helsinki (ThingNb3). You can use https://www.maps.ie/coordinates.html -site to find out Gps Coordinates on Google Map.

We'll query which things are neared that 500 km's from Seinäjoki.

```
curl -X POST --header 'Content-Type: application/json' --header
'Accept: application/json' -d '{ \
   "gpsLocation": { \
     "latitude": 62.78875546595712, \
     "longitude": 22.846313826953065 \
   }, \
   "distance": 500000, \
   "currentPage": 0, \
   "pageSize": 10 \
 }'
'http://localhost:27122/api/inventory/Core/GetNearbyPublicLocationThin
gs'
```

Note: Distance is in meters.

Note: Query uses paged syntax. First page is 0.

Returns:

```
{
  "things": [
    {
      "thingId": "inv1.sovelto.fi/T2",
      "title": "A Container",
      "distance": 745.8587000110062
    },
    {
      "thingId": "inv1.sovelto.fi/T1",
```

```
      "title": "MySuitcase",
      "distance": 27754.014016739896
    },
    {
      "thingId": "inv1.sovelto.fi/ThingNb3",
      "title": "A Thing",
      "distance": 312371.38540819706
    }
  ]
}
```


Page info is in Headers


```
"x-pagination":
"{\"CurrentPage\":0,\"MorePages\":false,\"PageSize\":10,\"TotalCount\"
:8}"
```


Services

Concept

- **Service**: the definition of a service. Each service must have a unique name. Each Thing has it's own service definitions. Currently a thing do not inherit services from it's Archetype Thing (but it will in the future versions)
- **ServiceRequest**: we request that this service must be executed. This will set all actions to the corresponding thins.
- **Action**: actions that must be executed when a Service is requested. There are several list of actions in each Service
    - **mandatoryActions** These action must be executed
    - **optionalActions** at least one of these actions must be executed
    - **selectedActions** Exactly one of these actions must be executed
    - **pendingActions**

Create a Service (definition)

Let's create a service named 'Notification of broken Streetlight' for an existing thing inv1.sovelto.fi/SL1.

Mandatory Actions are

- M100 must reserve a ladder truck (id inv1.sovelto.fi/LT1)
- M100 must change the bulb (thing SL1).

Timespan for this service is 10 mins.

If service is not done in time (or other failure), M100 will get notification (notification is ServiceAction of type Alarm or Fail)

All alarms go to M100

```
curl -X POST --header 'Content-Type: application/json' --header
'Accept: application/json' -d '{ \
   "title": "Notification of broken Streetlight", \
   "timespan": "00:10", \
   "alarmThingId": "inv1.sovelto.fi/M100", \
   "mandatoryActions": [ \
     { \
       "actionType": "GenericAction", \
       "title": "Reserve Ladder Truck", \
       "objectThingId": "inv1.sovelto.fi/LT1", \
       "operatorThingId": "inv1.sovelto.fi/M100" \
     }, \
 { \
       "actionType": "GenericAction", \
       "title": "Change Bulp", \
       "objectThingId": "inv1.sovelto.fi/SL1", \
       "operatorThingId": "inv1.sovelto.fi/M100" \
     } \
   ], \
   "optionalActions": [], \
   "selectedActions": [], \
   "pendingActions": [], \
   "session": "00000000-0000-0000-0000-000000000001", \
   "thingId": "inv1.sovelto.fi/SL1", \
   "role": "Omnipotent" \
 }' 'http://localhost:27122/api/inventory/Service/CreateService'
```

We should set execute rights for this new service. It's not implemented yet (currently all services can be requested anonymously).

Anonymous user – Request for Service

Anonymous users notes that a StreetLight (id: inv1.sovelto.fi/SL1) is broken. The user

1. EnterAnonymousSession
2. finds out which services are available,
3. request a service
4. and possibly queries what is the current status of his/her request.

Find out services.

1. EnterAnonymousSession, but we'll use predefined anonymous testing session 00000000-0000-0000-0000-000000000002

```
2. curl -X POST --header 'Content-Type: application/json' --header
   'Accept: application/json' -d '{ \
   "session": "00000000-0000-0000-0000-000000000002", \
```

```
   "thingId": "inv1.sovelto.fi/SL1", \
   "role": "Anonymous" \
 }' 'http://localhost:27122/api/inventory/Service/GetServices'
```

Returns:

```
{  "services": [
    "Notification of broken Streetlight"
  ]
}
```

```
3. curl -X POST --header 'Content-Type: application/json' -d '{ \
   "service": "Notification of broken Streetlight", \
   "session": "00000000-0000-0000-0000-000000000002", \
   "thingId": "inv1.sovelto.fi/SL1", \
   "role": "Anonymous" \
 }' 'http://localhost:27122/api/inventory/Service/ServiceRequest'
```

```
4. curl -X POST --header 'Content-Type: application/json' --header
   'Accept: application/json' -d '{ \
   "session": "00000000-0000-0000-0000-000000000002", \
   "thingId": "inv1.sovelto.fi/SL1", \
   "role": "Anonymous" \
 }' 'http://localhost:27122/api/inventory/Service/GetServiceStatus'
```

Returns:

```
{
  "statuses": [
    {
      "serviceId": "057cc518-fc5f-4a34-71e9-08d4a763c5f6",
      "title": "Notification of broken Streetlight",
      "requestedAt": "2017-05-30T13:57:13.0926044",
      "state": "NotStarted"
    }
  ]
}
```

The staff finds out which Actions to execute

The user M100

1. Starts a session
2. finds out which actions are set to him
3. Get's further information of the service
4. executes Action1 and update its status
5. executes Action2 and update its status

1. EnterAuthenticatedSession, but we'll use predefined testing session form M100 00000000-0000-0000-0000-000000000001

```
2. curl -X POST --header 'Content-Type: application/json' --header
   'Accept: application/json' -d '{ \
  "session": "00000000-0000-0000-0000-000000000001", \
  "thingId": "inv1.sovelto.fi/M100", \
  "role": "Omnipotent" \
}' 'http://localhost:27122/api/inventory/Service/GetActionStatuses'
```

Result:


```
{
  "statuses": [
    {
      "actionId": "09d02fd6-ea1a-43ed-43dc-08d4a763c5fa",
      "title": "Reserve Ladder Truck",
      "addedAt": "2017-05-30T13:57:13.1071165",
      "state": "NotStarted",
      "actionType": "Mandatory",
      "actionClass": "GenericAction"
    },
    {
      "actionId": "bca73fbf-1a5d-4998-f23d-08d48b0175c1",
      "title": "Change bulb",
      "addedAt": "2017-04-24T11:02:55.2765432",
      "state": "Done",
      "actionType": "Mandatory",
      "actionClass": "GenericAction"
    }
  ]
}
```


He should save the ActionIds because those are needed in the following calls.

```
3. curl -X POST --header 'Content-Type: application/json' --header
   'Accept: application/json' -d '{ \
  "actionId": "09d02fd6-ea1a-43ed-43dc-08d4a763c5fa", \
  "session": "00000000-0000-0000-0000-000000000001", \
  "thingId": "inv1.sovelto.fi/M100", \
  "role": "Omnipotent" \
}' 'http://localhost:27122/api/inventory/Service/GetActionStatus'
```

Result:


```
{
  "action": {
    "id": "09d02fd6-ea1a-43ed-43dc-08d4a763c5fa",
```

```
    "service": {
      "id": "057cc518-fc5f-4a34-71e9-08d4a763c5f6",
      "title": "Notification of broken Streetlight",
      "thingId": "inv1.sovelto.fi/SL1",
     "alarm_ThingId": "inv1.sovelto.fi/M100",
      "deadLine": "2017-06-08T02:28:55.7563762",
      "requestorThingId": "inv1.sovelto.fi/AnonymousUser",
      "state": "NotStarted",
      "sessionId": "00000000-0000-0000-0000-000000000002",
      "addedAt": "2017-05-30T13:57:13.0926044"
    },
    "title": "Reserve Ladder Truck",
    "actionType": "Mandatory",
    "objectThingId": "inv1.sovelto.fi/LT1",
    "operatorThingId": "inv1.sovelto.fi/M100",
    "actionClass": "GenericAction",
    "state": "NotStarted"
  }
}
```

Note: currently not implemented totally (operatorThingId is missing).

4.
```
curl -X POST --header 'Content-Type: application/json' -d '{ \
"actionId": "09d02fd6-ea1a-43ed-43dc-08d4a763c5fa", \
"state": "Done", \
"session": "00000000-0000-0000-0000-000000000001", \
"thingId": "inv1.sovelto.fi/M100", \
"role": "Omnipotent" \
}' 'http://localhost:27122/api/inventory/Service/UpdateActionStatus'
```

5.
```
curl -X POST --header 'Content-Type: application/json' -d '{ \
   "actionId": "bca73fbf-1a5d-4998-f23d-08d48b0175c1", \
   "state": "Done", \
   "session": "00000000-0000-0000-0000-000000000001", \
   "thingId": "inv1.sovelto.fi/M100", \
   "role": "Omnipotent" \
 }' 'http://localhost:27122/api/inventory/Service/UpdateActionStatus'
```

And then we can find out that the Service Request is done

```
curl -X POST --header 'Content-Type: application/json' --header
'Accept: application/json' -d '{ \
  "serviceId": "057cc518-fc5f-4a34-71e9-08d4a763c5f6", \
   "session": "00000000-0000-0000-0000-000000000002", \
   "thingId": "inv1.sovelto.fi/SL1", \
   "role": "Anonymous" \
```

```
 }' 'http://localhost:27122/api/inventory/Service/GetServiceStatus'
```

Result:


```
{
  "statuses": [
    {
      "serviceId": "057cc518-fc5f-4a34-71e9-08d4a763c5f6",
      "title": "Notification of broken Streetlight",
      "requestedAt": "2017-05-30T13:57:13.0926044",
      "state": "Done"
    }
  ]
}
```


If this service is not done in time, operatorThing will have ServiceActionStatus Alarm (or Failed) after deadline. OperatorThing must pull ServiceActionStatuses to get this “notification”.