1. GENERAL

Lambda function names should be in **CamelCase**.

Event payload and response keys use **camelBack** notation

1.1. Calling Lambda functions

Note that the Lambda.invoke() error callback is only called if the request itself fails, and not when some error occurs inside the function

Errors are communicated through the success callback, but with the errorMessage key set

Example request

```
{
  FunctionName: 'dev-UserLambda-12345',
  Payload: '{"action":"LIST"}'
}
```

Example response

```
{
}
```

1.2. Error messages

If something goes wrong the response will contain an error message. The error message is split into different parts to make it possible to localize the errors. The **message** field is a text representation of the error and that is the only field that is always present and it is also used for unknown errors that may occur. The **messageKey** field is used to localize the message and the possible values are described with each action below.
The **messageParams** field can be used to provide extra information about the error and if the error is related to a specific property it is specified in **property** field.

Example error

```
{
  errorMessage: {
    "message": "Number is smaller than 2",
    "messageKey: "NUMBER_TOO_SMALL",
    "messageParams": {
      "minValue": 2
    },
    "property": "age"
  }
}
```

1.2.1. General errors

These are the known errors that can occur that is not related to a specific action.

Key	Params	Property	Description
INVALID_ACTION	action		Returned if an invalid action is passed to the lambda.
NOT_AUTHORIZED	operation, objectType		Returned if the user is not authorized to perform the specified action.

1.3. Life cycle events

Some API:s also publishes life cycle events on MQTT. This can be used to listen to event concerning a specific thing or thing type and react when a new thing is created, a thing type removed or something else. The API:s that trigger life cycle events have that documented.

All events are published to the topic `event/<domainPath>` so for a thing that belongs to the sub domain sub1 the events will be published to `event/root/sub1`.

Example event

```
{
  timestamp: 1477456596272,
  message: 'Thingtype with id: 1 and label: Pump created',
  classification: 'INTERNAL',
  type: 'THING_TYPE.CREATE',
  source: {
    thingType: 1,
    domain: 'sub1'
  }
}
```

2. AUTH API

The Auth API (`AuthLambda`) is used to authenticate users and let new users sign up.

2.1. Actions

The following actions can be performed.

2.1.1. LOGIN – FINISHED

Checks if a user is authorized to login and returns credentials that should be used when communicating with AWS. The access token is valid for 15 minutes, whereafter it needs to be updated.

Example request payload

```
{
  action: 'LOGIN',
  attributes: {
    userName: 'demo',
    password: 'Demo1234'
  }
}
```

2.1.1.1. Input

Property	Required	Description
Attributes		
• userName	Yes	The user name of the user.
• password	Yes	The password of the user.

2.1.1.2. Output

Property	Always present	Description
User		
• userName	Yes	The user name of the user.
• roleName	Yes	The name of the role that the user has. (`Read` | `ReadWrite`)
• firstName	Yes	The first name of the user.
• lastName	Yes	The last name of the user.
• email	Yes	The e-mail address of the user.
Credentials		
• identityId	Yes	A Cognito IdentityId to use when communicating with AWS.
• token	Yes	A OpenID Connect token to use when communicating with AWS.
• refreshToken	Yes	A refresh token to use for getting a new access token

2.1.1.3. Errors

Key	Params	Property	Description
PROPERTY_REQUIRED		userName	Returned if no user name was provided.

Key	Params	Property	Description
PROPERTY_REQUIRED		password	Returned if no password was provided.
INVALID_LOGIN			Returned if the user name or password was incorrect.

2.1.2. SIGN_UP

Lets a user initiate the process to create a Cloud Connect user account. A successful signup will result in an email containing a link is sent to their provided email address. Following that link verifies their email address. If a user didn't get that email or if the link has expired, there is a possibility to resend the email via the endpoint RESEND_CONFIRMATION_CODE.

Example request payload

```
{
  action: 'SIGN_UP',
  attributes: {
    userName: 'demo',
    password: 'demo',
   email: ‘email@example.com’
  }
}
```

2.1.2.1. Input

Property	Required	Description
attributes		
• userName	Yes	The user name of the user.
• password	Yes	The password of the user.
• *email*	*Yes*	*The email address of the user.*
• *firstName*	*No*	*First name of the user*
• *lastName*	*No*	*Last name of the user*
• *phone*	*No*	*User’s phone number, needs to start with a country code (eg. +46) and cannot contain any whitespace or delimiters*
• *company*	*No*	*User’s company*
• *address*	*No*	*Address of the user*
• *zip*	*No*	*Zip code of the user’s address*
• *city*	*No*	*City of the user’s address*
• *country*	*No*	*Country of the user’s address*

2.1.2.2. Output

Property	Always present	Description
• userName	Yes	The user name of the created user.
• ...		

2.1.2.3. Errors

Key	Params	Property	Description
PROPERTY_REQUIRED		userName	Returned if no user name was provided.
PROPERTY_REQUIRED		password	Returned if no password was provided.
PROPERTY_REQUIRED		email	Returned if no email address was provided.
USER_USERNAME_EXISTS		userName	Returned if the user name that was provided is used by another user in the system.

2.1.3. RESEND_CONFIRMATION_CODE

Example request payload

```
{
  action: 'RESEND_CONFIRMATION_CODE',
  attributes: {
    userName: <username>
  }
}
```

2.1.4. FORGOT_PASSWORD

If a user has forgotten the account password, this endpoint can be used to let them reset their password. The user will get an email message containing a link to set a new password. The link is usable only once. *Note: After having invoked this endpoint, it is still possible to login using the correct credentials; the password is not changed until the user has responded to the email sent by the FORGOT_PASSWORD endpoint. Therefore, this endpoint can be exposed to unauthorized users.*

Example request payload

```
{
 action: 'FORGOT_PASSWORD',
  attributes: {
    userName: <username>
  }
}
```

2.1.5. REFRESH

Lets a user refresh the token. Use the refreshToken acquired from the LOGIN call descried above. By default, the refresh token is set to expire after 30 days but this can be changed to an arbitrary period.

To use the refresh token to get a new set of tokens, do the same call as you did when you logged in but use the refresh action instead and pass the refresh token as the argument. Please note that you need to reset your logins before you do this, otherwise you will try to do an authenticated call and since the access token is no longer valid, this will fail.

The response from refresh is exactly the same as from the login request, and should be handled the same way. In Cloud Connect Appboard the refresh token is implemented by checking for errors and if the token has expired, it is refreshed and the operation is retried, and if it still fails, the user is redirected to the login page.

Example request payload

```
{
  action: 'REFRESH',
  attributes: {
    refreshToken: 'xxxxxxxxx'
  }
}
```

3. DOMAIN API

The Domain API (`DomainLambda`) is used to manage domains. The domains are represented as a tree and the user can be restricted to only see part of the domain tree and that applies to all actions.

3.3.1. Actions

The following actions can be performed on the domain tree.

3.3.2. CREATE (role: ReadWrite) – FINISHED

Creates a new domain and adds it to the domain tree.

Example request payload

```
{
  action: 'CREATE',
  attributes: {
    id: 'sub',
    parentId: 'root',
    name: 'Sub Domain',
    description; 'A sub domain of root',
    data: {
      email: 'info@sub.root.se'
    }
  }
}
```

3.3.2.1. Input

Property	Required	Description	Since version
attributes			
• id	Yes	The id of the domain to add.	
• parentId	Yes	The id of the parent of the new domain.	
• name	Yes	The name of the domain.	2.1.0
• description	No	A longer description of the domain.	2.1.0
• data	No	Custom data about the domain. The data should match the structure defined in domainMetadata, see GET for more information.	2.1.0

3.3.2.2. Output

The entire domain tree will be returned so for the simple case with just a root node called `root` the following will be the output.

Property	Always present	Description	Since version
root	Yes	The id of the parent of the new domain.	

Property	Always present	Description	Since version
• attributes	Yes	Every node in the tree has an element called attributes where information about the domain is returned.	2.1.0
• ...			2.1.0
• sub	Yes	The id of the new domain.	
• attributes	Yes	Every node in the tree has an element called attributes where information about the domain is returned.	2.1.0
• name	Yes	The name of the domain.	2.1.0
• description	No	A longer description of the domain.	2.1.0
• data	No	Custom data about the domain.	2.1.0
• …	No	Other domains.	

3.3.2.3. Errors

Key	Params	Property	Description
NOT_AUTHORIZED_DOMAIN		parentId	Returned if the user tries to create a new domain under a domain that the user is not authorized to see.
PROPERTY_REQUIRED		id	Returned if no id was provided.
PROPERTY_REQUIRED		parentId	Returned if no parentId was provided.
PROPERTY_REQUIRED		name	Returned if no name was provided.
DOMAIN_ID_EXISTS		id	Returned if a domain with the specified id already exists.

3.3.3. GET (role: Read) – FINISHED

Added in version: **2.1.0**

Gets information about a domain.

Example request payload

```
{
  action: 'GET',
  attributes: {
    id: 'sub',
    parentId: null,
    name: null,
    description: null,
    data: null,
    domainMetadata: null,
    settings: null
  }
}
```

3.3.3.1. Input

Property	Required	Description	Since version
attributes			2.1.0
• id	Yes	The id of the domain to get.	2.1.0
• parentId	No	Set to `null` if the id of the parent of the domain should be included in the output.	2.1.0
• name	No	Set to `null` if the name of the domain should be included in the output.	2.1.0
• description	No	Set to `null` if a longer description of the domain should be included in the output.	2.1.0
• data	No	Set to `null` if custom data about the domain should be included in the output.	2.1.0
• domainMetadata	No	Set to `null` if metadata about the custom data should be included in the output. If the domain doesn't have any domain metadata specified the domain tree is searched upwards towards the root until domain metadata is found.	2.1.0
• thingTypeMetadata	No	Set to `null` if metadata about the custom data for thing typesshould be included in the output. If the domain doesn't have any thing type metadata specified the domain tree is searched upwards towards the root until thing type metadata is found.	2.5.0
• settings	No	Set to `null` if settings should be included in the output. If the domain doesn't have any specific settings the domain tree is search upwards towards the root until settings is found.	2.1.0

3.3.3.2. Output

Property	Always present	Description	Since version
id	Yes	The id of the domain	2.1.0
parentId	No	The id of the parent of the domain.	2.1.0
name	No	The name of the domain.	2.1.0
description	No	A longer description of the domain.	2.1.0
data	No	Custom data about the domain. Each field in the data object corresponds to a field in domainMetadata so the key is the id from domainMetadata.	2.1.0
domainMetadata[]	No	A list of metadata definitions for the custom data.	2.1.0
• id	(Yes)	A generated id for the field.	2.1.0
• label	(Yes)	A label describing the field. This is the information that should be shown to the user.	2.1.0
• type	(Yes)	The type of the field, can be one of `text` | `number`.	2.1.0

Property	Always present	Description	Since version
settings	No	The settings for the domain.	2.1.0
• theme	(Yes)	The theming settings for the domain.	2.1.0
• ...			

3.3.3.3. Errors

Key	Params	Property	Description
NOT_AUTHORIZED_DOMAIN		parentId	Returned if the user tries to get a domain that the user is not authorized to see.
PROPERTY_REQUIRED		id	Returned if no id was provided.
DOMAIN_NO_FOUND		id	Returned if the domain cannot be found.

3.3.4. LIST (role: Read) – FINISHED

Returns the entire domain tree.

Example request payload

```
{
  action: 'LIST',
  attributes: {
    name: null,
    description: null,
    data: null
 }
}
```

3.3.4.1. Input

Property	Required	Description	Since version
attributes			2.1.0
• name	No	Set to `null` if the name of the domain should be included in the output.	2.1.0
• description	No	Set to `null` if a longer description of the domain should be included in the output.	2.1.0
• data	No	Set to `null` if custom data about the domain should be included in the output.	2.1.0

3.3.4.2. Output

The entire domain tree will be returned so for the simple case with just a root node called `root` and a sub-domain called `sub` the following will be the output.

Property	Always present	Description	Since version
root	Yes	The id of the root domain.	

Property	Always present	Description	Since version
• attributes	Yes	Every node in the tree has an element called attributes where information about the domain is returned.	2.1.0
• name	No	The name of the root domain	2.1.0
• description	No	A longer description of the root domain.	2.1.0
• data	No	Custom data about the root domain.	2.1.0
• sub	Yes	The id of the new domain.	
• • attributes	Yes	Every node in the tree has an element called attributes where information about the domain is returned.	2.1.0
• name	Yes	The name of the sub domain.	2.1.0
• description	No	A longer description of the sub domain.	2.1.0
• data	No	Custom data about the sub domain.	2.1.0
• ...	No	Other domains.	

3.3.4.3. Errors

No errors.

3.3.5. REMOVE (role: ReadWrite) – FINISHED

Removes a domain and all of its sub-domains. A domain cannot be removed if there are users or things associated with the domain or any of its sub-domains.

Example request payload

```
{
  action: 'REMOVE',
  attributes: {
    id: 'sub'
  }
}
```

3.3.5.1. Input

Property	Required	Description
attributes		
• id	Yes	The id of the domain to remove.

3.3.5.2. Output

The entire domain tree will be returned so if the domain `sub` was removed from `root` the following can be the output.

Property	Always present	Description
root	Yes	The id of the parent of the removed domain.
• ...	No	Other domains.

3.3.5.3. Errors

Key	Params	Property	Description
NOT_AUTHORIZED_DOMAIN		id	Returned if the user tries to remove a domain that the user is not authorized to see.
PROPERTY_REQUIRED		id	Returned if no id was provided.
DOMAIN_HAS_THINGS			Returned if the domain cannot be removed because it has things attached to it or any of its sub-domains.
DOMAIN_HAS_USERS			Returned if the domain cannot be removed because it has users attached to it or any of its sub-domains.

3.3.6. UPDATE (role: ReadWrite) – FINISHED

Updates a domain and/or moves it and its sub-domains to a new parent domain.

Example request payload

```
{
  action: 'UPDATE',
  attributes: {
    id: 'sub',
    parentId: 'sub1',
    name: 'Sub Domain 1.1',
    description: 'A sub domain of sub1',
    data: {
      email: 'info@sub.sub1.root.se'
    }
  }
}
```

3.3.6.1. Input

Id is always required but all other properties are optional and only the properties provided are updated.

Property	Required	Description	Since version	Notes
attributes				
• id	Yes	The id of the domain to update/move.		

Property	Required	Description	Since version	Notes
• parentId	No	The id of the new parent to move the domain to.	(2.1.0)	Since version 2.1.0 this property is not required any more.
• name	No	The new name of the domain.	2.1.0	
• description	No	The new description of the domain.	2.1.0	
• data	No	The new custom data about the domain. The data should match the structure defined in domainMetadata, see GET for more information.	2.1.0	
• domainMetadata[]	No	The new list of metadata definitions for the custom data. The list is replaced so all fields must be provided.	2.1.0	
• id	(Yes)	A generated id for the field.	2.1.0	
• label	(Yes)	A label describing the field.	2.1.0	
• type	(Yes)	The type of the field, can be one of `text` | `number`.	2.1.0	
• settings	No	The new settings for the domain.	2.1.0	
• theme	No	The new theme setting for the domain.	2.1.0	

3.3.6.2. Output

The entire domain tree will be returned so if the domain `sub` was moved from `root` to `sub1` the following can be the output.

Property	Always present	Description	Since version
root	Yes	The id of the the old parent.	
• attributes	Yes		2.1.0
• ...			2.1.0
• sub1	Yes	The id of the new parent.	
• attributes			2.1.0
• ...			2.1.0
• sub	Yes	The id of the domain that was moved.	2.1.0
• attributes	Yes	Every node in the tree has an element called attributes where information about the domain is returned.	2.1.0
• name	Yes	The name of the root domain.	2.1.0
• description	No	A longer description of the domain.	2.1.0

Property	Always present	Description	Since version
• data	No	Custom data about the root domain.	2.1.0
• domainMetadata[ ]	No	A list of metadata definitions for the custom data. This is only included if domainMetadata is specified in the input.	2.1.0
• settings	No	The settings for the domain. This is only included if settings is specified in the input.	2.1.0

3.3.6.3. Errors

Key	Params	Property	Description
NOT_AUTHORIZED_DOMAIN		id	Returned if the user tries to move a domain that the user is not authorized to see.
NOT_AUTHORIZED_DOMAIN		parentId	Returned if the user tries to move the domain to a domain that the user is not authorized to see.
PROPERTY_REQUIRED		id	Returned if no id was provided.

4. FILE API

The File API (`FileLambda`) is used to manage files in the system.

4.1. Actions

The following actions can be performed.

4.1.1. LIST (role: Read) – DRAFT

Lists files in a specific bucket.

Status notes: The output may be changed before receiving FINISHED status.

Example request payload

```
{
  action: 'LIST',
  attributes: {
    bucketName: 'ThingFilesBucket',
    maxKeys: 1
  }
}
```

4.1.1.1. Input

Property	Required	Description
attributes		
• bucketName	Yes	The name of the bucket to get the files from.
• maxKeys	No	The maximum number of files to get.
• marker	No	The name of the file to start with when listing files (used for pagination).
• prefix	No	A name prefix that all files in the response must have.

4.1.1.2. Output

Property	Always present	Description
isTruncated	Yes	`false` if all files that matched the search criteria is included in the result, `true` otherwise.
marker	No	The name of the file that was used determine where to start listing files.
nextMarker	No	The name of the file to use as the marker to get the next list of files. If the response does not include nextMaker and it is truncated, you can use the name of the last file in the response as the marker in the subsequent request to get the next set of files.
name	Yes	The full name of the bucket.
prefix	No	The prefix that was used to get the files.
maxKeys	Yes	The maximum number of files that was returned.
encodingType	No	The encoding type used to encode file names in the response. (`url`)

Property	Always present	Description
files [ ]	Yes	A list of files matching the search criteria.
• Key	Yes	The name of the file.
• LastModified	Yes	The time when the files was last modified.
• ETag	Yes	The ETag of the file that can be used for caching.
• Size	Yes	The size of the file.
• StorageClass	Yes	The class of storage that was used. (`STANDARD` | `REDUCED_REDUNDANCY` | `GLACIER`)
• Owner	Yes	The owner of the file.
• DisplayName	Yes	
• ID	Yes	
• url	Yes	The full url to use to access the file.

4.1.1.3. Errors

Key	Params	Property	Description
PROPERTY_REQUIRED		bucketName	Returned if no bucket name was provided.

4.1.2. PUT_THING_IMAGE (role: ReadWrite) – DRAFT

Upload image for a thing

Status notes: The output may be changed before receiving FINISHED status.

Example request payload

```
{
  action: 'PUT_THING_IMAGE',
  attributes: {
    thingId: 'device-001',
    thingType: 123,
    encodedFile: 'data:image/jpeg;base64,/9j/4AAQSkZJRgABAQAAAQABAAD//
gA7QQyB2'
  }
}
```

4.1.2.1. Input

Property	Required	Description
attributes		
• thingId	Yes	The id of the specific thing.
• thingtype	Yes	The thingType for the specific thing.
• encodedFile	Yes	The image as a string in base64 image encoding.

4.1.2.2. Output

Property	Always present	Description
attributes		
• successMessage	No	A message to the user about the successful action.

4.1.2.3. Errors

No errors.

4.1.3. DELETE_THING_IMAGE (role: ReadWrite) – DRAFT

Delete an uploaded image for a thing

Status notes: The output may be changed before receiving FINISHED status.

Example request payload

```
{
  action: 'DELETE_THING_IMAGE',
  attributes: {
    thingId: 'device-001',
    thingType: 123
  }
}
```

4.1.3.1. Input

Property	Required	Description
attributes		
• thingId	Yes	The id of the specified thing.
• thingtype	Yes	The thingType for the specified thing.

4.1.3.2. Output

Property	Always present	Description
attributes		
• successMessage	No	A message to the user about the successful action.

4.1.3.3. Errors

No errors.

5. OBSERVATION API

The Observation API (`ObservationLambda`) is used to manage observations.

5.1. Actions

The following actions can be performed.

5.1.1. FIND (role: Read) – FINISHED

Added in version: **2.0.4**

Executes an Elasticsearch 2.3 query to find observations.

Example request payload

```
{
  action: 'FIND',
  query: {
    size: 0,
    trackScores: false,
    query: {
      bool: {
        must: [
          {
            range: {
              timestamp: {
                gte: 1460661813164,
                lte: 1460705013163
              }
            }
          },
          {
            term: {
              thingName: thingName
            }
          }
        ]
      }
    },
    aggs: {
      hist: {
        date_histogram: {
          field: 'timestamp',
          interval: '30m',
          pre_zone: '+01:00',
          pre_zone_adjust_large_interval: true,
          min_doc_count: 1,
```

5.1.1.1. Indata

Property	Required	Description

query	Yes	The body of an Elasticsearch Query DSL query. Any query and any aggregations are ok but the query must satisfy the following requirements:
		1. The query section in the supplied DSL must be possible to insert as-is into a "filtered query".
		2. The query must contain a range clause with a timestamp containing at least a `gte` key.
		3. The query cannot contain global aggregations.
		To search for a domain a term lookup can be used like this:
		`terms: {`
		`thingName: {`
		`index: 'domains',`
		`type: 'thing_mapping',`
		`id: 'root', // The domain to search for.`
		`path: 'thingNames'`
		`}`
		`}`

5.1.1.2. Output

The output from the Elasticsearch query.

5.1.1.3. Errors

Key	Params	Property	Description
PROPERTY_REQUIRED		query	Returned if no query was provided.
INVALID_ARGUMENTS		query	Returned if the query is invalid or doesn't meet the requirements specified above.

5.2. Accessing observations on S3

All observations are stored in raw format on Amazon S3. The bucket name can be found in the manifest under the key `ObservationsBucket`. The ingestion mechanism in Cloud Connect creates a separate file for each 5 minutes or 5MB (whatever happens first). The files are organized in the following folder structure: year > month > day > hour.

Inside each hour-folder are files which contains the raw JSON payload for all devices that communicated during the specified time-period. The file name uses the following convention:

```
<stack name>-firehose-1-<YYYY>-<MM>-<DD>-<HH>-<mm>-<dd>-<random_string>
```

Note that the JSON-messages are not separated by newline.

5.2.1. Example

Downloading all observation files for 23'rd of March 2016 using the AWS CLI:

```
aws cp s3://prod-observationsbucket-xxxxxxxxxxxxx/2016/03/23 . --
recursive
```

Note: Access to the S3 bucket requires custom IAM credentials, which are available from Telenor on request.

6. RULE API

The Rule API (`RuleLambda`) is used to manage rules.

6.1. Rule

A rule have the following properties.

Property	Required	Description
id	Yes	The id of the rule.
name	Yes	The name of the rule.
description	No	A longer description of the rule.
enabled	Yes	true if the rule should be enabled, false otherwise.
filter	Yes	Filter to decide what things this rule should trigger for.
• domain	Yes	The domain that the thing must belong to.
• thingType	Yes	The thing type that the thing must have.
• thingNames[]	No	A list of explicit thing names that the thing must be included in.
expression	Yes	The expression to evaluate when an observation is received for a thing that matches the filter.
• operator	Yes	The operator to use for the comparison in the expression. Can be one of the following: • `gt` – Greater than • `gte` – Greater than or equal • `lt` – Less than • `lte` – Less than or equal • `eq` – Equal • `ne` – Not equal
• resource	Yes	The resource to use in the expression. This can be a resource on a thing in which case this is just the name of the resource or it can be a resource of as sub thing but than the name should be the `<subthing_type>.<resource_name>`. If the sub thing doesn't have a type `untyped` should be used.
• Value	Yes	The value to compare the resource value to in the expression.
actions[]	Yes	A list of actions to perform when the rule triggers.

Property	Required	Description

• config	Yes	The config for the action. The config is specific to each action and is described on their own pages.
		
		All fields in the configuration are passed through a substitution phase where a payload from the rule engine can be used to replace variables with actual values. So for instance if a field in the config contains the following string "`Rule triggered for thing: ${thing.thingName}`" it will result in "`Rule triggered for thing: thing-001`" if the rule triggers for thing-001. The payload available is different for things and sub things but can look something like this:
		
		Payload for resources on things
		`resource: {`
		`  name: 'temp',`
		`  path: 'temp',`
		`  value: 50`
		`},`
		`timestamp: 1464081794634,`
		`rule: {`
		`  id: 'rule-001',`
		`  name: 'Test rule'`
		`},`
		`thing: {`
		`  thingName: 'thing-001',`
		`  thingType: 'Pump',`
		`  domain: 'root',`
		`  state: {`
		`    temp: 50`
		`  }`
		`}`
		
		Payload for resources on sub things
		`resource: {`
		`  name: 'temp',`
		`  path: 'sub1.temp',`
		`  value: 50,`
		`  parent: {`
		`    id: 'sub1',`
		`    state: {`
		`      temp: 50,`
		`      type: 'pump'`
		`    }`
		`  }`
		`},`
		`timestamp: 1464081794634,`
		`rule: {`

Property	Required	Description
• type	Yes	The type of action to perform. Currently only `WEBHOOK` is supported.
threshold	No	Additional conditions to throttle the number of times the rule is triggered.
• count	No	The minimum number of times that the rule must trigger before it actually calls the actions.
• interval	No	The minimum time (in milliseconds) since the last trigger before it actually calls the actions again.
readonly	No	A computed property that is true if the rule belongs to a domain higher up in the domain tree towards root. A user is always allowed to see rules that can effect things in the domain tree of the user but to update the rule it has to belong to the domain or a sub domain of the domain that the user belongs to.

6.2. Actions

The following actions can be performed.

6.2.1. CREATE (role: ReadWrite) – FINISHED

Creates a new rule.

Example request payload

```
{
  action: 'CREATE',
  attributes: {
    name: 'Thing Overheating',
    description: 'Triggered if the temperature is too high',
    enabled: true,
    filter: {
      domain: 'root',
      thingType: 'Pump',
      thingNames: ['thing-001', 'thing-002']
    },
    expression: {
      operator: 'gte',
      resource: 'temp',
      value: 50
    },
    actions: [
      {
        config: { ... },
        type: 'WEBHOOK'
      }
    ],
    threshold: {
      count: 0,
```

6.2.1.1. Input

Property	Required	Description
attributes		
• id	No	The id to give the new rule. If no id is provided the system generates a sequential id padded with 0s to be at least 8 characters long. In all environments except production this is also prefixed with the name of the stack. So in in development the first id is `dev-00000001`.
• name	Yes	See Rule definition above.
• description	No	See Rule definition above.
• enabled	Yes	See Rule definition above.
• filter	Yes	See Rule definition above.
• domain	Yes	See Rule definition above.
• thingType	Yes	See Rule definition above.
• thingNames[]	No	See Rule definition above.
• expression	Yes	See Rule definition above.
• operator	Yes	See Rule definition above.
• resource	Yes	See Rule definition above.
• value	Yes	See Rule definition above.
• actions[]	Yes	See Rule definition above.
• config	Yes	See Rule definition above.
• type	Yes	See Rule definition above.
• threshold	No	See Rule definition above.
• count	No	See Rule definition above.
• interval	No	See Rule definition above.

6.2.1.2. Output

The created rule (possibly including a generated id) as described above.

6.2.1.3. Errors

Key	Params	Property	Description
NOT_AUTHORIZED_DOMAIN			Returned if the user tries to add a rule to a domain that the user is not authorized to see.
PROPERTY_REQUIRED		name	Returned if no name was provided.
PROPERTY_REQUIRED		filter	Returned if no filter was provided.
PROPERTY_REQUIRED		expression	Returned if no expression was provided.
PROPERTY_REQUIRED		actions	Returned if no actions was provided.
PROPERTY_REQUIRED		threshold	Returned if no threshold was provided.

6.2.2. GET (role: Read) – FINISHED

Gets information about a rule.

Example request payload

```
{
  action: 'GET',
  attributes: {
    id: 'dev-001'
  }
}
```

6.2.2.1. Input

Property	Required	Description
attributes		
• id	Yes	The id of the rule to get.

6.2.2.2. Output

The matching rule as described above.

6.2.2.3. Errors

Key	Params	Property	Description
NOT_AUTHORIZED_DOMAIN			Returned if the user tries to get a rule that is assigned to a domain that the user is not authorized to see.
PROPERTY_REQUIRED		id	Returned if no id was provided.
RULE_NOT_FOUND		id	Returned if the rule cannot be found.

6.2.3. LIST (role: Read) – FINISHED

Lists all rules.

Example request payload

```
{
  action: 'LIST'
}
```

6.2.3.1. Input

No input.

6.2.3.2. Output

A list of rules as described above.

6.2.3.3. Errors

No errors.

6.2.4. REMOVE (role: ReadWrite) – FINISHED

Removes a rule.

Example request payload

```
{
```

6.2.4.1. Input

Property	Required	Description
attributes		
• id	Yes	The id of the rule to remove.

6.2.4.2. Output

No output

6.2.4.3. Errors

Key	Params	Property	Description
NOT_AUTHORIZED_DOMAIN			Returned if the user tries to remove a rule that is assigned to a domain that the user is not authorized to see.
PROPERTY_REQUIRED		id	Returned if no id was provided.
RULE_NOT_FOUND		id	Returned if the rule cannot be found.

6.2.5. UPDATE (role: ReadWrite) – FINISHED

Updates a rule.

Example request payload

```
{
  action: 'UPDATE',
  attributes: {
    userName: 'demo',
    firstName: 'First',
    lastName: 'Last',
    email: 'first.last@demo.se',
    phone: '031-123456',
    company: 'Demo AB',
    address: 'Street 1',
    zip: '12345',
    city: 'BigCity',
    country: 'Sweden',
    roleName: 'ReadWrite',
    domainName: 'root'
  }
}
```

6.2.5.1. Input

Property	Required	Description
attributes		
• id	Yes	See Rule definition above.

Property	Required	Description
• name	Yes	See Rule definition above.
• description	No	See Rule definition above.
• enabled	Yes	See Rule definition above.
• filter	Yes	See Rule definition above.
• domain	Yes	See Rule definition above.
• thingType	Yes	See Rule definition above.
• thingNames[]	No	See Rule definition above.
• expression	Yes	See Rule definition above.
• operator	Yes	See Rule definition above.
• resource	Yes	See Rule definition above.
• value	Yes	See Rule definition above.
• actions[]	Yes	See Rule definition above.
• config	Yes	See Rule definition above.
• type	Yes	See Rule definition above.
• threshold	No	See Rule definition above.
• count	No	See Rule definition above.
• interval	No	See Rule definition above.

6.2.5.2. Output

The updated rule as described above.

6.2.5.3. Errors

Key	Params	Property	Description
NOT_AUTHORIZED_DOMAIN			Returned if the user tries to move a user to a domain that the user is not authorized to see.
PROPERTY_REQUIRED		id	Returned if no id was provided.
PROPERTY_REQUIRED		name	Returned if no name was provided.
PROPERTY_REQUIRED		filter	Returned if no filter was provided.
PROPERTY_REQUIRED		expression	Returned if no expression was provided.
PROPERTY_REQUIRED		actions	Returned if no actions was provided.
PROPERTY_REQUIRED		threshold	Returned if no threshold was provided.
RULE_NOT_FOUND		id	Returned if the rule cannot be found.

7. THING BATCH API

The Thing Batch API (`ThingBatchLambda`) is used to managed batches of things.

7.1. Actions

The following actions can be performed.

7.1.1. CREATE (role: ReadWrite) – FINISHED

Creates a number of things by creating a batch that is processed in the background.

Example request payload

```
{
  action: 'CREATE',
  attributes: {
    thingType: 'Lights',
    domain: 'root'
  },
  requestedSize: 100,
  concurrency: 5
}
```

7.1.1.1. Indata

Property	Required	Default	Description
attributes			
• thingType	Yes		The thing type of the things to create.
• domain	Yes		The name of the domain that the things should be assigned to.
requestedSize	Yes		The number of things to create.
concurrency	No	3	The number of things that should be created at the same time.

7.1.1.2. Output

Property	Always present	Description
batchId	Yes	The id of the batch.

7.1.1.3. Errors

Key	Params	Property	Description
NOT_AUTHORIZED_DOMAIN		domain	Returned if the user tries to add a thing to a domain that the user is not authorized to see.
PROPERTY_NOT_A_NUMBER		requestedSize	Returned if the requested size was not a number.
PROPERTY_NOT_IN_RANGE	min:1, max: 1000	requestedSize	Returned if the requested size was not between 1 and 1000.
PROPERTY_REQUIRED		thingType	Returned if no thing type was provided.

Key	Params	Property	Description
PROPERTY_REQUIRED		domain	Returned if no domain was provided.
PROPERTY_REQUIRED		requestedSize	Returned if no requested size was provided.

7.1.2. LIST (role: Read) – DRAFT

Lists meta data for all thing batches for a specific thing type, including a download url for generated thing credentials.

Note: There is no paging functionality, only the first page is delivered from an alphabetically sorted file list. The page contents are then sorted by batch property finishedAt descending, leaving ev. unfinished bathes on top. In order to get the latest batches, the batch file names needs to have a timestamp following the prefix eg. batch-${thingTypeId}-${createdAtTimestamp}.

Status notes: The output may be changed before receiving FINISHED status.

Example request payload

```
{
  action: 'LIST',
  attributes: {
    bucketName: 'ThingCertsBucket',
    prefix: ´batch-${thingTypeId}-´,
    maxKeys: 100
  }
}
```

7.1.2.1. Input

Property	Required	Default	Description	
attributes				
• bucketName	Yes		The name of the bucket to get the files from.	
• maxKeys	No	1000	The maximum number of files to get.	
• marker	No		Not used. Pagination is not possible.	
• prefix	No		A name prefix that all files in the response must have. This property should target the batch name pattern used for the thing type.	
filter				
• domains	No		An array of domain ids that, if specified, is used to limit the list of thing batches.	2.5.0
• thingTypes	No		An array of thing type ids that, if specified, is used to limit the list of thing batches.	2.5.0
• freeTexts	No		An array of any text that, if specified, is used to limit the list of thing batches.	2.5.0

7.1.2.2. Output

Property	Always present	Description
isTruncated	Yes	`false` if all files that matched the search criteria is included in the result, `true` otherwise.
marker	No	The name of the file that was used determine where to start listing files.
nextMarker	No	The name of the file to use as the marker to get the next list of files. If the response does not include nextMaker and it is truncated, you can use the name of the last file in the response as the marker in the subsequent request to get the next set of files.
name	Yes	The full name of the bucket.
prefix	No	The prefix that was used to get the files.
maxKeys	Yes	The maximum number of files that was returned.
encodingType	No	The encoding type used to encode file names in the response. (`url`)
files [ ]	Yes	A list of files matching the search criteria sorted by time the batch finished, with batches currently under processing on top.
• Key	Yes	The name of the file.
• LastModified	Yes	The time when the files was last modified.
• ETag	Yes	The ETag of the file that can be used for caching.
• Size	Yes	The size of the file.
• StorageClass	Yes	The class of storage that was used. (`STANDARD` | `REDUCED_REDUNDANCY` | `GLACIER`)
• Owner	Yes	The owner of the file.
• DisplayName	Yes	
• ID	Yes	
• url	No	The full url to use to access the thing credentials archive file for the things created by the batch job. Only present if the batch has completed successfully.
• batchId	Yes	Name of the related create thing batch job.
• currentSize	Yes	Current size (number of things) of the related create thing batch. This represents the number of things created so far by the batch job.
• requestedSize	Yes	Requested size (number of things) of the related create thing batch.
• concurrency	Yes	The concurrency for the related create thing batch job, ie. the number of things that should be created at the same time. Default is 3.
• attributes	Yes	Attributes of the related create thing batch job:
• thingType	Yes	• The thingType of the things in the related create thing batch job.
• domain	Yes	• The domain of the things in the related create thing batch job.
• createdBy	Yes	• Name of the user that created the thing batch job.

Property	Always present	Description
• createdAt	Yes	The time when the create thing batch job was created.
• finishedAt	No	The time when the create thing batch job finished.
• errors	Yes	Array of errors encountered during the create thing batch job. The array is empty if there are no errors.

7.1.2.3. Errors

Key	Params	Property	Description
PROPERTY_REQUIRED		bucketName	Returned if no bucket name was provided.

8. THING EVENT API - DEPRECATED

NOTE! The Thing Event API is deprecated and will be removed in a future release. Use the MQTT Pub/Sub API instead.

The Thing event service lives in front of the MQTT broker and controls access to realtime events for things, according to the domain model and user role. It allows authenticated clients to publish and subscribe on MQTT topics for all things that the user account has access to. Publishing and subscribing is very similar to how it would work against the actual MQTT broker, however the communication with the client is implemented using a Socket.io channel.

When the client sends a subscribe request to Thing Event, it authorizes the request and creates a corresponding subscription in MQTT. As messages arrive from MQTT, Thing Event relays these to the client by emitting "event" messages. Clients may also send "publish" messages to Thing Event, which are forwarded to the MQTT broker after successful authorization. The message payload for publish, subscribe and event are described by the below example.

Thing event does not perform its own authentication, instead the client must supply the temporary AWS credentials that was received during the normal API authentication flow.

8.1. Javascript example

```
// Connect to the Thing Event service using Socket.io
var socket = io("https://thing-event.cloud.tcxn.net:8443")

// Subscribe to events for thing 00000001
var pl = {
  accountNumber: manifest.AccountNumber,
  stackName: manifest.Env,
  credentials: {
    accessKeyId: "xxxxxxxx",
    secretAccessKey: "yyyyyyy",
    sessionToken: "zzzzzzz",
  },
  thingName: "00000001"
}
socket.emit("subscribe", pl)
socket.on("event", function (data) {
  console.log(JSON.stringify(data)
})

// Publish a desired state for 00000001
var pl = {
  accountNumber: manifest.AccountNumber,
  stackName: manifest.Env,
  credentials: {
    accessKeyId: "xxxxxxxx",
    secretAccessKey: "yyyyyyy",
    sessionToken: "zzzzzzz",
  },
  topic: "$aws/things/00000001/shadow/update",
  payload: '{ "state": { "desired": { "light": "on" }}}'
}

socket.emit("publish", pl)
```

8.2. Direct access to the MQTT broker

For integration scenarios where the fine-grained authorization performed by Thing Event is not required, or even desired, it is possible to publish and subscribe directly against the MQTT broker. Direct access to the MQTT broker requires custom TLS certificates or IAM credentials, which are available from Telenor on request.

9. MQTT PUB/SUB API

This API should be used instead of the deprecated Thing Event API.

This API exposes two MQTT topic structures, one called pub and the other sub, that take the domain of the user and the thing into consideration.

9.1. Subscribing to thing events

By subscribing to a topic of the form ***sub/domainPathOfTheThing/thingName*** a logged in user identity can, if the user identity's domain is in the domain path of the thing, see all events occurring in real time for a thing, i.e. what the thing actually reports on the topic **$aws/things/*thingName*/shadow/update**.

Given a thing called 'myTestThing' with 'root' as its domain and a user with 'root' as its domain the following are valid topic subscriptions (please note that the 'root' domain is implicit in topics).

sub/myTestThing, subscribing to only the events of the thing 'myTestThing'.

sub/#, subscribing to all events on all things under 'root' and on all other things on all other domains.

If the domain of the thing 'myTestThing' was a subdomain called 'subdomain1' and a direct child of 'root' the subscription topics would be:

sub/subdomain1/myTestThing, subscribing to only the events of the thing 'myTestThing'.

sub/subdomain1/#, subscribing to all events on all things under 'subdomain1' and on all other things on all other domains beneath 'subdomain1'. A user identity with the domain 'root' could still of course do a 'catch all' subscription by subscribing to ***sub/#***.

9.2. Publishing to things

The pub topic is used for trying to set desired state on a thing, i.e. scenarios where we want to update a thing with a new value.

The criteria that need to be fulfilled in order for anything to be routed from the pub topic to the thing are:

1. The document must be a state desired JSON document of the form:

```
{
  "state": {
        "desired" : {
            // Resources to update
        }
     }
}
```

2. The full domain path of the thing to publish to must be part of the topic and must be valid, i.e. the domain path must be found in the domain tree.

3. The name of the thing must be the last part of the topic and the last part of the domain path must match the domain of the thing

If all criteria is met the payload on the topic will be reposted on the IoT system topic **$aws/things/*thingName*/shadow/update**.

Some examples:

pub/myTestThing, publish to the thing 'myTestThing' that has the domain 'root',

pub/subdomain1/myTestThing, publish to the thing 'myTestThing' that has the domain 'subdomain1',

pub/subdomain1/subdomain2/subdomain3/myTestThing, publish to the thing 'myTestThing' that has the domain 'subdomain3',

9.3. Topic privileges of a logged in identity

When a user identity is logged in it is only given privileges to the topic structures that match the identities domain.

Given a domain tree structure of:

- Root
  - subdomain1
    - subdomain2

and a user identity with the domain ‘subdomain2’ the pub or sub topics available to this user will be ***sub/subdomain1/subdomain2/*** and ***pub/subdomain1/subdomain2/.***

11. THING MANAGEMENT API

The Thing Management API (`ThingLambda`) is used to manage things and thing certificates.

11.1.Actions

The following actions can be performed.

11.1.1. CREATE (role: ReadWrite) – FINISHED

Creates a new thing.

Example request payload

```
{
  action: 'CREATE',
  attributes: {
    thingName: 'MyThing',
    thingType: 'TestType',
    domain: 'root'
  }
}
```

11.1.1.1. Indata

Property	Required	Description
attributes		
• thingName	No	The name to give the new thing. If no thingName is provided the system generates a sequential name padded with 0s to be at least 8 characters long. In all environments except production this is also prefixed with the name of the stack. So in in development the first id is `dev-00000001`.
• thingType	Yes	The thing type of the thing.
• domain	Yes	The id of the domain that the thing should be assigned to.

11.1.1.2. Output

Property	Always present	Description
thingName	Yes	The name of the thing.
thingType	Yes	The thing type of the thing.
createdAt	Yes	The time when the thing was created.
createdBy	Yes	The user name of the user that created the thing.
domain	Yes	The id of the domain that the thing should be assigned to.
label	Yes	The label describing the thing. Label is always the same as thingName at creation time but can be changed later to something else.

11.1.1.1. Life cycle event

Will trigger a life cycle event with the type `THING.CREATE`.

11.1.1.2. Errors

Key	Params	Property	Description
NOT_AUTHORIZED_DOMAIN		domain	Returned if the user tries to add a thing to a domain that the user is not authorized to see.
PROPERTY_REQUIRED		thingType	Returned if no thing type was provided.
PROPERTY_REQUIRED		domain	Returned if no domain was provided.

11.1.2. DOWNLOAD_CERTIFICATE (role: Read) – FINISHED

Downloads a certificate for a thing.

Example request payload

```
{
  action: 'DOWNLOAD_CERTIFICATE',
  attributes: {
    thingName: 'MyThing'
  }
}
```

11.1.2.1. Indata

Property	Required	Description
attributes		
• thingName	Yes	The name of the thing to download the certificate for.

11.1.2.2. Output

A zip-file containing the certificate for the thing

11.1.2.3. Errors

Key	Params	Property	Description
NOT_AUTHORIZED_DOMAIN			Returned if the user tries to download a certificate for a thing that belongs to a domain that the user is not authorized to see.
PROPERTY_REQUIRED		thingName	Returned if no thing name was provided.
THING_NOT_FOUND		thingName	Returned if the thing cannot be found.

11.1.3. FIND (role: Read) – FINISHED

Executes an Elasticsearch 2.3 query to find things.

Example request payload

```
{
  action: 'FIND',
  query: {
    size: 1000,
```

11.1.3.1. Indata

Property	Required	Description
query	Yes	The body of an Elasticsearch Query DSL query. Any query and any aggregations are ok but the query must satisfy the following requirements: 1. The query section in the supplied DSL must be possible to insert as-is into a "filtered query". 2. The query cannot contain global aggregations.

11.1.3.2. Output

The output from the Elasticsearch query.

11.1.3.3. Errors

Key	Params	Property	Description
PROPERTY_REQUIRED		query	Returned if no query was provided.

11.1.4. GET (role: Read) – FINISHED

Gets information about a thing.

Example request payload

```
{
  action: 'GET',
  attributes: {
    thingName: 'MyThing',
    shadow: null,
    label: null
  }
}
```

11.1.4.1. Indata

Property	Required	Description	Since version
attributes			
• thingName	Yes	The name the thing to get.	
• thingType	No	Set to `null` if the thing type of the thing should be included in the output.	2.1.0
• createdAt	No	Set to `null` if the time when the thing was created should be included in the output.	2.1.0
• createdBy	No	Set to `null` if the username of the user who created the thing should be included in the output.	2.1.0
• domain	No	Set to `null` if the domain of the thing should be included in the output.	2.1.0
• label	No	Set to `null` if the label describing the thing should be included in the output.	2.1.0

Property	Required	Description	Since version
• description	No	Set to `null` if a longer description of the the thing should be included in the output.	2.1.0
• shadow	No	Set to `null` if the thing shadow of the thing should be included in the output.	2.1.0

11.1.4.2. Output

Property	Always present	Description	Since version	Notes
thingName	Yes	The name of the thing to get.	2.1.0	
thingType	No	The thing type of the thing.	2.1.0	
createdAt	No	The time when the thing was created.	2.1.0	
createdBy	No	The user name of the user who created the thing.	2.1.0	
domain	No	The domain of the thing.	2.1.0	
• id	(Yes)	The id of the domain.	2.1.0	
• name	(Yes)	The name of the domain.	2.1.0	
• description	(Yes)	A longer description of the domain.	2.1.0	
• data	(Yes)	Custom data about the domain.	2.1.0	
label	No	The label describing the thing.	2.1.0	
description	No	A longer description of the thing.	2.1.0	
shadow	No	The thing shadow of the thing.	(2.1.0)	In version 2.1.0 the behavior for the action GET was changed to only include the attributes that the user specify so shadow is now only present if the indata includes `shadow: null`.
• state	(Yes)	See AWS Documentation.		
• metadata	(Yes)	See AWS Documentation.		
• timestamp	(Yes)	See AWS Documentation.		
• version	(Yes)	See AWS Documentation.		

11.1.4.3. Errors

Key	Params	Property	Description
NOT_AUTHORIZED_DOMAIN			Returned if the user tries to get a thing that belongs to a domain that the user is not authorized to see.
PROPERTY_REQUIRED		thingName	Returned if no thing name was provided.

11.1.5. REMOVE (role: ReadWrite) – FINISHED

Removes a thing.

Example request payload

```
{
  action: 'REMOVE',
  attributes: {
    thingName: 'MyThing'
  }
}
```

11.1.5.1. Indata

Property	Required	Description
attributes		
• thingName	Yes	The name of the thing to remove.

11.1.5.2. Output

No output.

11.1.5.1. Life cycle event

Will trigger a life cycle event with the type `THING.REMOVE`.

11.1.5.2. Errors

Key	Params	Property	Description
NOT_AUTHORIZED_DOMAIN			Returned if the user tries to remove a thing that belongs to a domain that the user is not authorized to see.
PROPERTY_REQUIRED		thingName	Returned if no thing name was provided.
THING_NOT_FOUND			Returned if the thing cannot be found.

11.1.6. REPLACE_CERTIFICATE (role: ReadWrite) – FINISHED

Revokes the current certificate of a thing and creates a new.

Example request payload

```
{
  action: 'REPLACE_CERTIFICATE',
  attributes: {
    thingName: 'MyThing'
```

11.1.6.1. Indata

Property	Required	Description
attributes		
• thingName	Yes	The name of the thing to replace the certificate for.

11.1.6.2. Output

No output.

11.1.6.3.Errors

Key	Params	Property	Description
NOT_AUTHORIZED_DOMAIN			Returned if the user tries to replace the certificate of a thing that belongs to a domain that the user is not authorized to see.
PROPERTY_REQUIRED		thingName	Returned if no thing name was provided.
THING_NOT_FOUND			Returned if the thing cannot be found.

11.1.7. UPDATE (role: ReadWrite) – FINISHED

Updates a thing.

Example request payload

```
{
  action: 'UPDATE',
  attributes: {
    thingName: 'MyThing',
    domain: 'root',
    label: 'TestLabel',
    description: 'TestDescription'
  }
}
```

11.1.7.1. Indata

Property	Required	Description
attributes		
• thingName	Yes	The name of the thing to update.
• domain	Yes	The id of the domain that the thing should belong to.
• label	Yes	The label describing the thing.
• description	No	A longer description of the thing.

11.1.7.2.Output

Property	Always present	Description
Property	**Always present**	**Description**
thingType	Yes	The thing type of the thing
thingName	Yes	The name of the thing.
domain	Yes	The id of the domain of the thing.
label	Yes	The label describing the thing
description	No	A longer description of the thing.

11.1.7.1. Life cycle event

Will trigger a life cycle event with the type `THING.UPDATE`.

11.1.7.2.Errors

Key	Params	Property	Description
NOT_AUTHORIZED_DOMAIN			Returned if the user tries to update a thing that belongs to a domain that the user is not authorized to see.
NOT_AUTHORIZED_DOMAIN		domain	Returned if the user tries to move a thing to a domain that the user is not authorized to see.
PROPERTY_REQUIRED		thingName	Returned if no thing name was provided.
THING_NOT_FOUND			Returned if the thing cannot be found.

12. THING TYPE API

The Thing Type API (`ThingTypeLambda`) is used to manage thing types.

12.1.Actions

The following actions can be performed.

12.1.1. CREATE (role: ReadWrite) – FINISHED

Creates a new thing type.

Example request payload

```
{
  action: 'CREATE',
  attributes: {
    id: 'Lights',
    domain: 'root',
    label: 'Lights',
    description: 'Light switches'
  }
}
```

12.1.1.1. Indata

Property	Required	Description	Since version
attributes			
• id	Yes	The id of the thing type.	
• domain	Yes	The id of the domain that the thing type should belong to.	
• label	Yes	The label describing the thing type.	
• description	No	A longer description of the thing.	
• data	No	Custom data about the thing type. The data should match the structure defined in thingTypeMetadata, see Domain.GET for more information.	2.5.0

12.1.1.2. Output

Property	Always present	Description	Since version
id	Yes	The id of the thing type.	
domain	Yes	The id of the domain that the thing type belongs to.	
label	Yes	The label describing the thing type.	
description	No	A longer description of the thing.	

Property	Always present	Description	Since version
data	No	Custom data about the domain	2.5.0
viewMode	Yes	The name of the selected view mode (`DefaultView` by default after create).	
viewModes	Yes	The available view modes for the thing type. By default one view mode, `DefaultView`, is created as follows.	
• DefaultView	Yes	The default view mode.	
• id	Yes	The id of the view mode.	
• label	Yes	The label of the view mode.	
• description	No	A longer description of the view mode.	
• thingWidgets	Yes	A list of widgets to show for a thing.	
• collectionWidgets	No	A list of widgets to show in the thing type overview.	
• ...	No	Other view modes.	

12.1.1.1. Life cycle event

Will trigger a life cycle event with the type `THING_TYPE.CREATE`.

12.1.1.2. Errors

Key	Params	Property	Description
PROPERTY_REQUIRED		id	Returned if no id was provided.
PROPERTY_REQUIRED		label	Returned if no label was provided.
THING_TYPE_ID_EXISTS		id	Returned if the id that was provided is i used by another thing type in the system.

12.1.2. GET (role: Read) – FINISHED

Gets information about a thing type.

Example request payload

```
{
  action: 'GET',
  attributes: {
    id: 'Lights'
  }
}
```

12.1.2.1. Indata

The only required attribute is `id` to know which thing type to get but any of the following attributes can be included with the value `null` to get other attributes about the thing type. (`label` | `readOnly` | `thingCount` | `domain` | `data` | `resources` | `viewMode` | `virtualResources`). If only id is provided all attributes are included in the output.

12.1.2.2.

Property	Required	Description
attributes		
• id	Yes	The id of the thing type to get.
• ...	No	Other attributes you want to get.

12.1.2.3. Output

Property	Always present	Description	Since version
id	Yes	The id of the thing type.	
domain	Yes	The id of the domain that the thing type belongs to.	
label	Yes	The label describing the thing type.	
description	No	A longer description of the thing.	
data	No	Custom data about the thing type	2.5.0
readOnly	Yes	True if the user has the Read role or if the thing type belongs to a parent domain of the domain that the user belongs to, false otherwise.	
thingCount	Yes	The number of things that have the thing type.	
resources	Yes	The resource taxonomy of the thing type. This is initialised by the resources of a corresponding thing shadow. It is updated with subsequent received resources.	
viewMode	Yes	The name of the selected view mode.	
viewModes	Yes	The available view modes for the thing type.	
• DefaultView	Yes	The default view mode.	
• id	Yes	The id of the view mode.	
• label	Yes	The label of the view mode.	
• description	No	A longer description of the view mode.	
• thingWidgets	Yes	A list of widgets to show for a thing.	
• collectionWidgets	No	A list of widgets to show in the thing type overview.	
• ...	No	Other view modes.	

12.1.2.4. Errors

Key	Params	Property	Description
PROPERTY_REQUIRED		id	Returned if no id was provided.
THING_TYPE_NOT_FOUND		id	Returned if the thing type cannot be found.

12.1.3. LIST (role: Read) – FINISHED

Lists all thing types.

Example request payload

```
{
  action: 'LIST'
```

12.1.3.1. Indata

There are no required attributes but by including attributes with the value `null` you specify which attributes to get. If you don't add any attributes all attributes are included in the output. You can choose from the following attributes. (`label` | `readOnly` | `thingCount` | `domain` | `data` | `resources` | `viewMode` | `virtualResources`)

Property	Required	Description
attributes		
• ...	Yes	The attributes you want to get.

12.1.3.2. Output

A list of thing types where each item in the list have the following properties.

Property	Always present	Description	Since version
id	Yes	The id of the thing type.	
label	Yes	The label describing the thing type.	
description	No	A longer description of the thing.	
data	No	Custom data about the thing type.	2.5.0
readOnly	Yes	True if the user has the Read role or if the thing type belongs to a parent domain of the domain that the user belongs to, false otherwise.	
thingCount	Yes	The number of things that have the thing type.	
resources	Yes	The resource taxonomy of the thing type. This is initialised by the resources of a corresponding thing shadow. It is updated with subsequent received resources.	
viewMode	Yes	The name of the selected view mode.	
viewModes	Yes	The available view modes for the thing type.	

Property	Always present	Description	Since version
• DefaultView	Yes	The default view mode.	
• id	Yes	The id of the view mode.	
• label	Yes	The label of the view mode.	
• description	No	A longer description of the view mode.	
• thingWidgets	Yes	A list of widgets to show for a thing.	
• collectionWidgets	No	A list of widgets to show in the thing type overview.	
• ...	No	Other view modes.	

12.1.3.3. Errors

No errors.

12.1.4. REMOVE (role: ReadWrite) – FINISHED

Removes a thing type

Example request payload

```
{
  action: 'REMOVE',
  attributes: {
    id: 'Lights'
  }
}
```

12.1.4.1. Indata

Property	Required	Description
attributes		
• id	Yes	The id of the thing type to remove.

12.1.4.2. Output

No output.

12.1.4.1. Life cycle event

Will trigger a life cycle event with the type `THING.REMOVE`.

12.1.4.2. Errors

Key	Params	Property	Description
PROPERTY_REQUIRED		id	Returned if no id was provided.
THING_TYPE_NOT_FOUND		id	Returned if the thing type cannot be found.
THING_TYPE_AS_THINGS			Returned if the thing type cannot be removed because it has things attached to it.

12.1.5. REMOVE_ATTRIBUTE (role: ReadWrite) – FINISHED

Removes an attribute from the thing type. This is normally used to remove view modes.

Example request payload

```
{
  action: 'REMOVE_ATTRIBUTE',
  attributes: {
    id: 'Lights',
    keyPath: 'viewModes.DefaultView'
  }
}
```

12.1.5.1. Indata

Property	Required	Description
attributes		
• id	Yes	The id of the thing type to remove the attribute from.
• keyPath	Yes	The key path to the attribute to remove.

12.1.5.2. Output

Property	Always present	Description
id	Yes	The id of the thing type.
label	Yes	The label describing the thing type.
description	Yes	A longer description of the thing.
resources	Yes	The resource taxonomy of the thing type. This is initialised by the resources of a corresponding thing shadow. It is updated with subsequent received resources.
viewMode	Yes	The name of the selected view mode.
viewModes	Yes	The available view modes for the thing type.
• DefaultView	Yes	The default view mode.
• id	Yes	The id of the view mode.
• label	Yes	The label of the view mode.
• description	No	A longer description of the view mode.
• thingWidgets	Yes	A list of widgets to show for a thing.
• collectionWidgets	No	A list of widgets to show in the thing type overview.
• ...	No	Other view modes.

12.1.5.3. Errors

Key	Params	Property	Description
PROPERTY_REQUIRED		id	Returned if no id was provided.

Key	Params	Property	Description
PROPERTY_REQUIRED		kryPath	Returned if no key path was provided.
THING_TYPE_NOT_FOUND		id	Returned if the thing type cannot be found.

12.1.6. UPDATE (role: ReadWrite) – FINISHED

Update a thing type.

Example request payload

```
{
  action: 'UPDATE',
  attributes: {
    id: 'Lights',
    label: 'Lights',
    description: 'Light switches'
  }
}
```

12.1.6.1. Indata

Property	Required	Description	Since version
attributes			
• id	Yes	The id of the thing type.	
• label	Yes	The label describing the thing type.	
• description	No	A longer description of the thing.	
• data	No	Custom data about the thing type	2.5.0

12.1.6.2. Output

Property	Always present	Description
Yes	The id of the thing type.	The id of the thing type.
label	Yes	The label describing the thing type.
description	Yes	A longer description of the thing.
data	No	Custom data about the thing type.
resources	Yes	The resource taxonomy of the thing type. This is initialised by the resources of a corresponding thing shadow. It is updated with subsequent received resources.
viewMode	Yes	The name of the selected view mode.
viewModes	Yes	The available view modes for the thing type.
• DefaultView	Yes	The default view mode.
• id	Yes	The id of the view mode.

Property	Always present	Description
• label	Yes	The label of the view mode.
• description	No	A longer description of the view mode.
• thingWidgets	Yes	A list of widgets to show for a thing.
• collectionWidgets	No	A list of widgets to show in the thing type overview.
• ...	No	Other view modes.

12.1.6.3. Errors

Key	Params	Property	Description
PROPERTY_REQUIRED		id	Returned if no id was provided.
PROPERTY_REQUIRED		label	Returned if no label was provided.
THING_TYPE_NOT_FOUND		id	Returned if the thing type cannot be found.

12.1.7. UPDATE_ATTRIBUTE (role: ReadWrite) – FINISHED

Removes an attribute from the thing type. This is normally used to manage view modes.

Example request payload

```
{
  action: 'UPDATE_ATTRIBUTE',
  attributes: {
    id: 'Lights',
    keyPath: 'viewModes.DefaultView.description',
    item: 'A new description'
  }
}
```

12.1.7.1. Indata

Property	Required	Description
attributes		
• id	Yes	The id of the thing type to update the attribute on.
• keyPath	Yes	The key path to the attribute to update.
• item	Yes	The item to put in the attribute.

12.1.7.2. Output

Property	Always present	Description
Yes	The id of the thing type.	The id of the thing type.
label	Yes	The label describing the thing type.

Property	Always present	Description
description	Yes	A longer description of the thing.
resources	Yes	The resource taxonomy of the thing type. This is initialised by the resources of a corresponding thing shadow. It is updated with subsequent received resources.
viewMode	Yes	The name of the selected view mode.
viewModes	Yes	The available view modes for the thing type.
• DefaultView	Yes	The default view mode.
•• id	Yes	The id of the view mode.
•• label	Yes	The label of the view mode.
•• description	No	A longer description of the view mode.
•• thingWidgets	Yes	A list of widgets to show for a thing.
•• collectionWidgets	No	A list of widgets to show in the thing type overview.
• ...	No	Other view modes.

12.1.7.1. Life cycle event

Will trigger a life cycle event with the type `THING.UPDATE`.

12.1.7.2. Errors

Key	Params	Property	Description
PROPERTY_REQUIRED		id	Returned if no id was provided.
PROPERTY_REQUIRED		keyPath	Returned if no key path was provided.
PROPERTY_REQUIRED		item	Returned if no item was provided.
THING_TYPE_NOT_FOUND		id	Returned if the thing type cannot be found.

13. USER API

The User API (`UserLambda`) is used to manage users and user profiles

13.1. Actions

The following actions can be performed.

13.1.1. CREATE (role: ReadWrite) – DRAFT

Creates a new user.

Status note: *This API needs to handle validation better before receiving FINISHED status.*

Example request payload

```
{
  action: 'CREATE',
  attributes: {
    userName: 'demo',
    password: 'demo',
    firstName: 'First',
    lastName: 'Last',
    email: 'first.last@demo.se',
    phone: '+4631123456',
    company: 'Demo AB',
    address: 'Street 1',
    zip: '12345',
    city: 'BigCity',
    country: 'Sweden',
    roleName: 'ReadWrite',
    domainName: 'root'
  }
}
```

13.1.1.1. Indata

Property	Required	Description
attributes		
• userName	Yes	The user name of the user.
• password	Yes	The password of the user.
• firstName	Yes	The first name of the user.
• lastName	Yes	The last name of the user.
• email	Yes	The e-mail address of the user.
• phone	No	The phone number of the user, needs to start with a country code (eg. +46) and cannot contain any whitespace or delimiters
• company	No	The company that the user works at.
• address	No	The address of the user.

Property	Required	Description
• zip	No	The zip code of the user.
• city	No	The city of the user.
• country	No	The country of the user.
• roleName	Yes	The name of the role that the user has. (`Read` | `ReadWrite`).
• domainName	Yes	The name of the domain that the user should be assigned to.

13.1.1.2. Output

Property	Always present	Description
userName	Yes	The user name of the user.
firstName	Yes	The first name of the user.
lastName	Yes	The last name of the user.
email	Yes	The e-mail address of the user.
phone	No	The phone number of the user.
company	No	The company that the user works at.
address	No	The address of the user.
zip	No	The zip code of the user.
city	No	The city of the user.
country	No	The country of the user.
roleName	Yes	The name of the role that the user has. (`Read` | `ReadWrite`)
domainName	Yes	The name of the domain that the user is assigned to.

13.1.1.3. Errors

Key	Params	Property	Description
NOT_AUTHORIZED_DOMAIN		domainName	Returned if the user tries to add a user to a domain that the user is not authorized to see.
PROPERTY_REQUIRED		userName	Returned if no userName was provided.
PROPERTY_REQUIRED		password	Returned if no password was provided.
PROPERTY_REQUIRED		roleName	Returned if no role name was provided.
PROPERTY_REQUIRED		domainName	Returned if no domain name was provided.
USER_USERNAME_EXISTS		userName	Returned if the user name that was provided is used by another user in the system.

13.1.2. RESET_PASSWORD

Resets the password for the specified user which results in a link being sent by email to the user, that can visit the link to set a new password. The link is usable only once. If the link has expired, this endpoint can be invoked again. Note: Following a password reset, the user account is disabled until the password has been successfully changed.

Example request payload

```
{
  action: 'RESET_PASSWORD',
  attributes: {
    userName: <username>
  }
}
```

13.1.3. GET (role: Read) – FINISHED

Gets information about a user.

Example request payload

```
{
  action: 'GET',
  attributes: {
    userName: 'demo',
    firstName: null
  }
}
```

13.1.3.1. Input

The only required attribute is `userName` to know which user to get. But any of the following attributes can be included with the value `null` to get other attributes about the user. (`firstName` | `lastName` | `email` | `phone` | `company` | `address` | `zip` | `country` | `roleName` | `domainName`)

Property	Required	Description
attributes		
• userName	Yes	The user name of the user to get.
• ...	No	Other attributes you want to get.

13.1.3.2. Output

Property	Always present	Description
userName	Yes	The user name of the user to get.
...	No	Other attributes you want to get.

13.1.3.3. Errors

Key	Params	Property	Description
NOT_AUTHORIZED_DOMAIN			Returned if the user tries to get a user that is assigned to a domain that the user is not authorized to see.
PROPERTY_REQUIRED		userName	Returned if no user name was provided.
USER_NOT_FOUND		userName	Returned if the user cannot be found.

13.1.4. LIST (role: Read) – FINISHED

Lists all users. There are three categories of users. Active users have access to login. Pending users have not been assigned domainName and roleName. Unconfirmed users have signed up but not visited the link in the confirmation email and set a new password. To retrieve all users regardless of category use the ‘all’ category.

Example request payload

13.1.4.1. Input

There are no required attributes but by including attributes with the value `null` you specify which attributes to get. You can choose from the follwing attributes. (`userName` | `firstName` | `lastName` | `email` | `phone` | `company` | `address` | `zip` | `country` | `roleName` | `domainName`)

Property	Required	Description
attributes		
• ...	Yes	The attributes you want to get.
filter	no	
• freeText	no	A string to filter results on.
• category	no	Filter for the category of the user. Possible values are ‘all’, ‘active’, ‘pending’, ‘unconfirmed’. Defaults to ‘active’.
page	no	Page to list, i.e. pagination support.
size	no	Number of users to retrieve per page. If not set all users are returned.
sortProp	no	Name of attribute to sort by, defaults to ‘userName’

13.1.4.2. Output

A list of users where each item in the list have the following properties.

Property	Always present	Description
users	Yes	The list of users in the chosen category (active by default)
• ...	Yes	The attributes you want to get.
totalPages	Yes	Number of pages available, support for pagination. If size was not set this will return 1.
page	Yes	The actual page returned. Will be set to 1 if neither size nor page was set in the request.
metadata	Yes	
• count	Yes	Includes the keys ‘all’, ‘active’, ‘pending’ and ‘unconfirmed’ where the values tell how many users are returned per category

13.1.4.3. Errors

No errors.

13.1.5. REMOVE (role: ReadWrite) – FINISHED

Removes a user.

Example request payload

```
{
  action: 'REMOVE',
  attributes: {
    userName: 'demo'
  }
}
```

13.1.5.1. Input

Property	Required	Description
attributes		
• userName	Yes	Either a single user name as a string of the user to remove or a list of user names to remove. See examples above.

13.1.5.2. Output

No output

13.1.5.3. Errors

Key	Params	Property	Description
NOT_AUTHORIZED_DOMAIN			Returned if the user tries to remove a user that is assigned to a domain that the user is not authorized to see.
PROPERTY_REQUIRED		userName	Returned if no user name was provided.
USER_REMOVE_DEMO			Returned if you try to remove the demo user.
USER_NOT_FOUND		userName	Returned if the user cannot be found.

13.1.6. UPDATE (role: ReadWrite) – DRAFT

Updates a user. There is a possibility to disable an active user, using the enabled attribute as shown below.

Status note: This API needs to handle validation better before receiving FINISHED status.

Example request payload

```
{
  action: 'UPDATE',
  attributes: {
    userName: 'demo',
    firstName: 'First',
    lastName: 'Last',
    email: 'first.last@demo.se',
    phone: '031-123456',
    company: 'Demo AB',
    address: 'Street 1',
    zip: '12345',
    city: 'BigCity',
    country: 'Sweden',
    roleName: 'ReadWrite',
    domainName: 'root',
    enabled: 'true'
  }
}
```

13.1.6.1. Input

Property	Required	Description
attributes		
• userName	Yes	The user name of the user.
• firstName	No	The first name of the user.
• lastName	No	The last name of the user.
• email	No	The e-mail address of the user.
• phone	No	The phone number of the user.
• company	No	The company that the user works at.
• address	No	The address of the user.
• zip	No	The zip code of the user.
• city	No	The city of the user.
• country	No	The country of the user.
• roleName	No	The name of the role that the user has. (`Read` | `ReadWrite`)
• domainName	No	The name of the domain that the user should be assigned to.
• enabled	No	Possible values 'true' and 'false'. Note that they need to be encapsulated in quotes since the payload format is JSON which not supports Booleans.

13.1.6.2. Output

Property	Always present	Description
userName	Yes	The user name of the user.
firstName	Yes	The first name of the user.
lastName	Yes	The last name of the user.
email	Yes	The e-mail address of the user.
phone	No	The phone number of the user.
company	No	The company that the user works at.
address	No	The address of the user.
zip	No	The zip code of the user.
city	No	The city of the user.
country	No	The country of the user.
roleName	Yes	The name of the role that the user has. (`Read` | `ReadWrite`)
domainName	Yes	The name of the domain that the user is assigned to.

13.1.6.3. Errors

Key	Params	Property	Description
NOT_AUTHORIZED_DOMAIN		domainName	Returned if the user tries to move a user to a domain that the user is not authorized to see.
PROPERTY_REQUIRED		userName	Returned if no user name was provided.

Key	Params	Property	Description
USER_NOT_FOUND		userName	Returned if the user cannot be found.

13.1.7. UPDATE_PROFILE (role: Read) – DRAFT

Updates the profile of the logged in user. Currently updating the password is not possible at this endpoint. To achieve that the user needs to invoke the FORGOT_PASSWORD and follow the email link as described above.

Status note: *This API needs to handle validation better before receiving FINISHED status.*

Example request payload

```
{
  action: 'UPDATE_PROFILE',
  attributes: {
    userName: 'demo',
    firstName: 'First',
    lastName: 'Last',
    email: 'first.last@demo.se',
    phone: '+4631123456',
    company: 'Demo AB',
    address: 'Street 1',
    zip: '12345',
    city: 'BigCity',
    country: 'Sweden'
  }
}
```

13.1.7.1. Input

Property	Required	Description
attributes		
• userName	Yes	The user name of the user.
• firstName	Yes	The first name of the user.
• lastName	Yes	The last name of the user.
• email	Yes	The e-mail address of the user.
• phone	No	The phone number of the user.
• company	No	The company that the user works at.
• address	No	The address of the user.
• zip	No	The zip code of the user.
• city	No	The city of the user.
• country	No	The country of the user.

13.1.7.2. Output

Property	Always present	Description
userName	Yes	The user name of the user.
firstName	Yes	The first name of the user.
lastName	Yes	The last name of the user.
email	Yes	The e-mail address of the user.
phone	No	The phone number of the user.
company	No	The company that the user works at.
address	No	The address of the user.
zip	No	The zip code of the user.
city	No	The city of the user.
country	No	The country of the user.
roleName	Yes	The name of the role that the user has. (`Read` | `ReadWrite`)
domainName	Yes	The name of the domain that the user is assigned to.

13.1.7.3. Errors

Key	Params	Property	Description
NOT_AUTHORIZED			Returned if the user tries to update the profile of another user.
PROPERTY_REQUIRED		userName	Returned if no user name was provided.
USER_INVALID_PASSWORD		oldPassword	Returned if a new password is provided but the oldPassword doesn't match the password of the user.
USER_NOT_FOUND		userName	Returned if the user cannot be found.

13.1.8. GET_USERDATA

The user data endpoint is available for storage of information connected to a user. The data is stored as a plain json structure, and is created and updated by calling UPDATE_USERDATA with the specified data. Note that there is no need to provide which user to store the data for, the data will be stored for the current logged in user. There is a reserved property 'tcxn' which stores metadata about the user and should not be used to store your information.

Example request payload

```
{
  action: 'GET_USERDATA',
  attributes: {}
}
```

13.1.8.1. Input

No input.

13.1.8.2. Output

Property	Always present	Description
tcxn	Yes	Reserved key for platform usage.

13.1.8.3. Error

No error.

13.1.9. UPDATE_USERDATA

Example request payload

```
{
  action: 'UPDATE_USERDATA',
  attributes: <userdata>
}
```

13.1.9.1. Input

Property	Always present	Description
attributes		
• tcxn	No	Reserved usage for platform, do not override this property.
• …	Yes	Arbitrary object with data you want to store for the user.

13.1.9.2. Output

Property	Always present	Description
attributes		
• tcxn	Yes	Reserved usage for platform.
• …	Yes	Arbitrary object with data you want to store for the user.

13.1.9.3. Error

No error.

14. EXAMPLES

14.1.Javascript (browser)

Example code for logging in and calling an API endpoint using a web browser.

1. Copy the code into a text editor, such as notepad on Windows
2. Edit the appUrl, userName and password variables
3. Save the file as index.html on your hard drive
4. Open the file in a web browser
5. Refresh credentials

```
<html>
  <head>
    <script src="https://code.jquery.com/jquery-1.12.0.min.js"></
script>
    <script src="https://sdk.amazonaws.com/js/aws-sdk-2.2.29.min.js"></
script>
  </head>
  <body>
    <h3>Cloud Connect Javascript example</h3>
    <pre id="output">Please wait...</pre>
    <script>

    AWS.config.region = "eu-west-1";

    /* CONFIGURATION SECTION */
    var appUrl = "";                           // for example
demo.cc.telenorconnexion.com
    var userName = "";                         // your username
    var password = "";                         // your password
    /* END OF CONFIGURATION SECTION */

    /
**********************************************************************
     * 1) Load the manifest - it's used to lookup the "physical" names
of
     * resources we want to use

**********************************************************************/

    $.getJSON(manifestUrl(appUrl), function(mf) {

      /
**********************************************************************
       * 2) Initialize Cognito credentials using the Identity pool from
       * the manifest. This will give us access to the
"unauthenticated" role
       * that allows us to call login function

***********************************************************************/

      AWS.config.credentials = new AWS.CognitoIdentityCredentials({
        IdentityPoolId: mf.IdentityPool
      });
      AWS.config.credentials.clearCachedId();

       /
***********************************************************************
       * 3) Call Cloud Connect Login function. This will return a
session token
       * (JWT) that we can use to "upgrade" our Cognito credentials to
full
       * permissions

***********************************************************************/

      var loginPayload = {
        action: 'LOGIN',
        attributes: {
          userName: userName,
          password: password
        }
      };
      callLambda(mf.AuthLambda, loginPayload, function(res) {
        var creds = res.credentials;
        AWS.config.credentials.params.Logins = {

          ['cognito-idp.' + mf.Region + '.amazonaws.com/' +
mf.UserPool]: creds.token

        };
        AWS.config.credentials.expired = true;

        /
***********************************************************************
         * 4) Finally call an API endpoint, for example find things

***********************************************************************/

        var findThingsPayload = {
          action: 'FIND',
          query: {
            size: 3,
            query: {
              match_all: {}
            }
          }
        };
        callLambda(mf.ThingLambda, findThingsPayload, function(res) {
          $("#output").html(JSON.stringify(res.hits.hits, null, 2));
        });
      });
    });

    function manifestUrl(hostname) {
      return 'https://1u31fuekv5.execute-api.eu-west-1.amazonaws.com' +
             '/prod/manifest/?hostname=' + hostname;
    }
    /
****************************************************************************
     * Helper function for Lambda invoke + error handling

****************************************************************************/

    function callLambda(name, payload, callback) {
      var params = {
        FunctionName: name,
        Payload: JSON.stringify(payload)
      };
      var lambda = new AWS.Lambda();
      lambda.invoke(params, function(err, res) {
      if (!err) {
        var pl = JSON.parse(res.Payload);
        if (!pl.errorMessage) {
          callback(pl);
        } else {
          alert("Lambda function returned an error: " +
pl.errorMessage);
        }
      } else {
        alert("Lambda request failed" + err.toString());
      }
    });
  };

  </script>
</body>
</html>
```

16. REVISION HISTORY

Version	Date	Changes
2.3	2016-08-25	Updated Observation API. Added password parameter to Update in User API.
2.4	2016-09-28	Updated Thing Type API Added information on domain, readOnly, thingCount and added support for selection of attribute to get.
2.5	2016-10-26	Added information about life cycle events. Updated Domain API with information about thing type metadata. Updated Thing Batch API with information about additional filters. Updated Thing Type API with information about custom data.
2.6	2016-11-09	Updated USER API: List and Remove
2.7	2016-11-30	Updated FILE API: Put and Delete image
2.8	2016-11-30	Updated Authentication API: Login. Updated USER API: User data.
2.9	2017-01-20	No updates.
2.10	2017-02-08	Added new API: MQTT Pub/Sub API Announced MQTT Pub/Sub API to replace Thing Event API from now on. Announced Thing Event API to be deprecated in future release (not decided, but at the earliest R2.13).
2.11		