Eemsgolaan 3
9727 DW Groningen
P.O. Box 1416
9701 BK Groningen
The Netherlands

www.tno.nl

T +31 88 866 70 00
F +31 88 866 77 57
infodesk@tno.nl

# HTTP REST Interface AnySenseConnect Version 3

Date 8 March 2012

Author(s) Bram van der Waaij, Erik Langius, Matthijs Vonder, Mente Konsman, Wiltfried Pathuis

Number of pages 51
Document number 35643
Document version 1.0

# Contents

# 1 Introduction

AnySenseConnect is a data collection and - distribution platform for large scale sensor infrastructures. This document describes the HTTP REST Interface for AnySenseConnect that can be used as Application Programming Interface (API) to request for data and meta data of “observers” and groups of observers over the internet. This HTTP REST interface can be used by third party applications to, for example, build apps for mobile devices, websites, control rooms, etc.

We use the term “observers” instead of sensors because this has a broader meaning. Observers can be physical sensors but could also be text messages, pictures, messages from social media etc.

# 2 Interface Model

## 2.1 Rest protocol

AnySenseConnect uses the REST protocol to interface with clients, REST stands for REpresentational State Transfer. One of the possible implementations is the well-known HTTP protocol. The table below shows four HTTP commands which are used in REST.

| HTTP Protocol | Data action |
|---|---|
| POST | Create a resource within a given collection |
| GET | Retrieve a resource |
| PUT | Update a resource |
| DELETE | Delete a resource |

Based on these four commands, AnySenseConnect uses the following convention:

### *2.1.1 Overall conventions*

- Identification of a group of resources or a single resource goes via the URL
- The URL starts with "/connect/v3" indicating this is the 3rd version of the AnySenseConnect REST interface.
- Specification of the attributes of resources is in the JSON format.
- Resource elements all have a unique ID, using the UUID4 format.

### *2.1.2 HTTP command conventions*

GET Group-resource
Returns a list of all resource elements within this group
Example: "GET /projects" returns a list of all projects

GET Element-resource
Returns all attributes of a specific resource element.
Example: "GET /projects/12345" returns all attributes of project 12345

POST Group-resource
Creates a new element from this group and adds it to the group. After that the HTTP call is redirected to the GET element-resource in order to return the uuid of the newly created resource element.
Example: "POST /projects" creates a new project with a new ID (54321) and redirects the call to "GET /projects/54321"

PUT Element-resource
Updates the attributes of this resource element
Example: "PUT /project/12345" Updates the attributes of this project.

DEL Element-resource
Removes this element from the group
Example: "DEL /projects/12345" Removes project 12345

PUT Parent-element-resource/Child-element-resource
Connects a child resource to a father resource
Example: “PUT /groups/12345/observer/67890” Connects the observer 67890 to the group 12345

DEL Parent-element-resource/Child-element-resource
Disconnects a child resource to a father resource
Example: “DEL /groups/12345/observer/67890” Disconnects the observer 67890 to the group 12345

## 2.2 AnySenseConnect REST Resource model

Figure 1 shows the AnySenseConnect resource model belonging to the REST interface resources. The resource model can be divided into three layers, depicted in the three levels of grey. The top layer is about resources for the management. The middle layer is about the actual instances. The lower layer is about the templates, the instances are based upon.

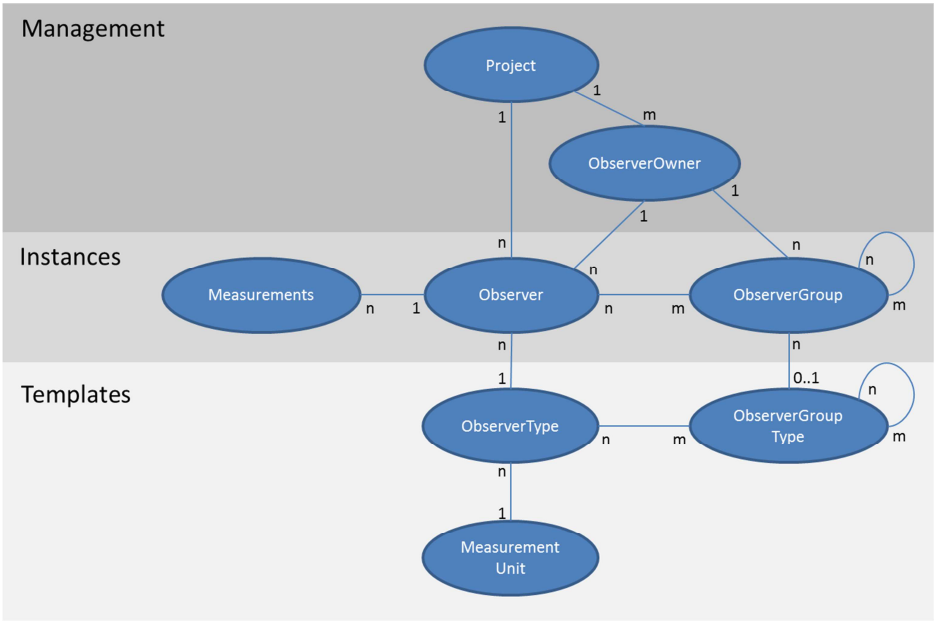


Figure 1: The REST resource model

### 2.2.1 *Resource Project*

*Description*

A resource containing information about a project. AnySenseConnect can host several projects. Each project is entirely separated from other projects.

*Relations with*

A project can have several observerOwners and observers. Note that the other resources are also bound to the project, via all the relations.

### 2.2.2 *Resource ObserverOwner*

*Description*

A resource containing information about an owner of observers.

*Relations with*

An observerOwner belongs to one project and can have several observers and observerGroups.

### 2.2.3 *Resource Observer / RootObserver*

*Description*

A resource containing information about an observer. This can be a physical sensor, or virtual sensor, or any other source of measurement data.

A special kind of observers are rootObservers[1]. These are observers which exist as a single entity (not part of a device). So a temperature sensor is a rootObserver if it is a single temperature sensor. It is not a rootObserver if it is one of several sensors within a device.

Note that in case of a cable of observers, the observers are rootObservers because they also can exist as a single entity. The cable can be described using the rootGroup resource.

*Relations with*

An observer has several measurements. Belongs to a project and is owned by an observerOwner. Can belong to one or more observerGroups. Is from a specific observerType.

### 2.2.4 *Resource ObserverType*

*Description*

A resource containing information about the type of an observer. The observer is one of many instances of one observerType. For example the type is a temperature observer of which there are four observers.

*Relations with*

An observerType can have several instantiations as an observer. Its measures measurementunits. It can belong to several observerGroupTypes.

### 2.2.5 *Resource MeasurementUnit*

*Description*

A resource containing information about the unit of measurement. For example temperature can be described in Celsius, Fahrenheit, etc. All different measurement units.

*Relations with*

A measurement unit can be used by several observerTypes.

[1] The word Root is chosen because the RootObserver (and the RootGroups) form the root, the starting point of all available observers.

### 2.2.6 *Resource Measurement*

*Description*

A resource containing information about a measurement. Possible measurements are:

- Long
- Double
- Boolean
- String
- URL

*Relations with*

A measurement belongs to one observer.

### 2.2.7 *Resource ObserverGroup / RootGroup / LogicalGroup*

*Description*

A resource containing information about a group of observers and /or other observerGroups. Observergroups can be used for two different purposes:

1. To model devices containing more than one sensor.
   For example a weather station contains temperature, air pressure, wind speed and direction sensors. All within the same device = observergroup.
2. To make groupings of observers and other groups which have a meaning for the user.
   For example all sensors within a certain room, or all temperature sensors in a building, etc.

In the REST interface these two different uses are represented by three different subclasses of the ObserverGroup:

1. RootGroup[2]
   A resource for modeling devices containing several observers

2. LogicalGroup.
   A resource for modeling user defined groups.

3. A third group exists: groups within a device, the innerGroup. For example: within the weather station device (a rootGroup), the windspeed and winddirection are grouped together into “windparameters” (an innerGroup). Innergroups do not have their own resource, because they only exists as a template (ObserverGroupType). Only logical- and rootGroups can be created. The innergroups follow via the template. All groups can be accessed via the oberserverGroup resource.

[2] The word Root is chosen because the RootGroups (and the RootObservers) form the root, the starting point of all available observers. Containing observers and/or innergroups.

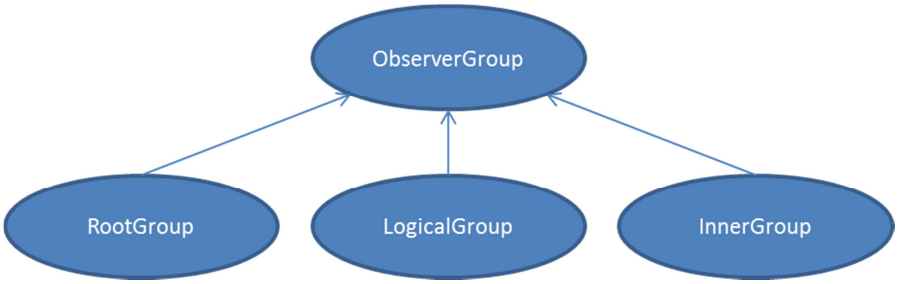


Figure 2: Three different kind of observer groups

*Relations with*
An observerGroup is owned by one observerOwner. Contains observers and/or observerGroups. Can be based on a template observerGroupType.

### *2.2.8 Resource ObserverGroupType*

*Description*
A resource containing information about the template of an observerGroup. ObserverGroups can be used to described devices containing several observers. Those devices can be described using observerGroupTypes in order to easily make many instances of these complex structures.

*Relations with*
An observerGroupType has several instantiations as an observerGroup. It can belong to several observerTypes.

## 2.3 Example using the REST resource model

In Figure 3 an example shows a real life situation how to model an set of observers and make a user defined view on them.

Physically the following devices are present:
- 1 water height sensor (WH)
- 3 geobead sensor devices containing:
  - 1 Temperature sensor
  - 1 inclination group_sensor
    - A CF component
    - A CL component
- 1 cable connecting the three geobeads.

Logically the following groups are present:
- A cross section containing references to
  - The inclination group_sensor of the three geobeads
  - The water height sensor

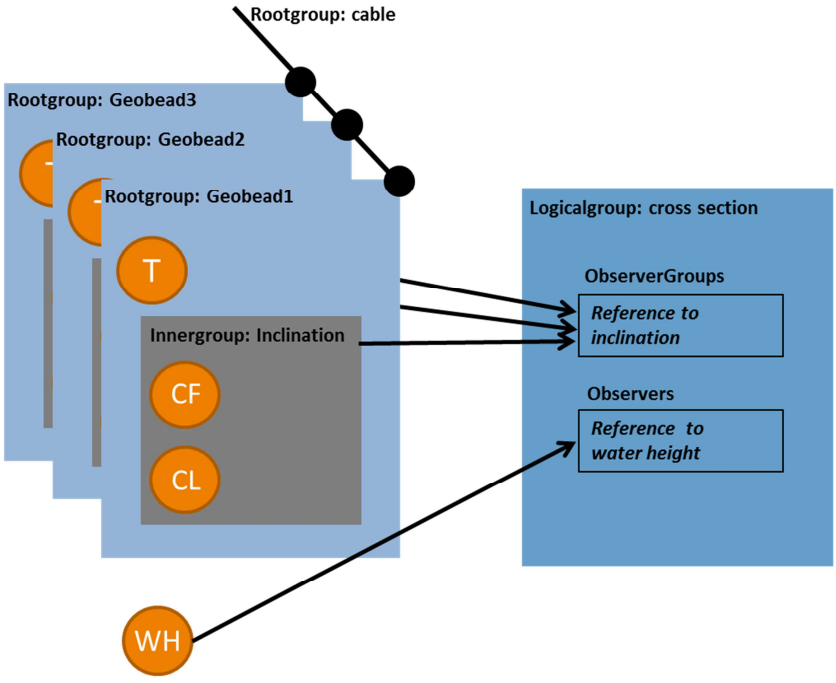


Figure 3: An example of how to model observers in the REST resource model

# 3 Formats

## 3.1 UUID4

The REST API is designed as a computer interface, hence all identifications are computer readable, not necessarily human readable.

The UUID version 4 uses a scheme relying only on random numbers. This algorithm sets the version number as well as two reserved bits. All other bits are set using a random or pseudorandom data source. Version 4 UUIDs have the form xxxxxxxx-xxxx-4xxx-yxxx-xxxxxxxxxxxx where x is any hexadecimal digit and y is one of 8, 9, A, or B. e.g. f47ac10b-58cc-4372-a567-0e02b2c3d479" [see also http://en.wikipedia.org/wiki/Universally_unique_identifier]

## 3.2 Timestamp format

Timestamps within AnySenseConnect can be in two formats:

1. A long containing the time in ms since 1970 (unix epoch)
2. A string with the format YYYY-MM-DDTHH:MM:SS.sss in which the milliseconds (.sss) or optional

Timestamps are always in UTC unless specified otherwise

## 3.3 Mediatypes

The AnysenseConnect HTTP REST interface can support different mediatypes. The current version supports
- application/json

Future versions may support:
- application/xml
- text/csv

Mediatypes can also be described using wildcards (*), in that case AnySenseConnect will use the following mappings:

- application/* maps on application/json
- */* maps on HTML encapsulated application/json

Note that when performing AnySenseConnect REST calls from a browser, the requested mediatype can vary. Most browsers can not directly request application/json.

## 3.4 Timelines

An additional AnySenseConnect module can handle more than only the core timeline. The core timeline contains the raw measurements of the observers. Additional timelines offer functionality like: measurement versions, interpolation & aggregation, multiple simulation storage.

# 4 REST API

This chapter describes all the in AnySenseConnect available resources and the different commands that can be performed on them. The following resource groups are available:

/projects
/projects/observerowners
/projects/observers
/projects/types/observertypes
/projects/types/measurementunits
/projects/rootgroups
/projects/logicalgroups
/projects/observergroups
/projects/types/observergrouptypes
/projects/timelines/observersmeasurements
/timezones

## 4.1 Resource projects

### *4.1.1 POST /projects*

```
Syntax: HTTP://<server:port>/connect/v3/projects

Method: POST

Description:
  Creates a new project
  <name> is mandatory, must be unique and may not be empty
  <description> is optional
  <startdate> is mandatory and may not be empty
  <enddate> is optional and may not be empty
  The return is a redirect to the created project

Payload (application/json):
{
  "name":"<name>",
  "description":"<description>",
  "startdate":"<timestamp utc formatted as long>",
  "enddate":"<timestamp utc formatted as long>"
}

Return code:
  HTTP Redirect see other (303) to GET /projects/<projectid>
  HTTP CLIENT_ERROR_CONFLICT (409) when the project <name> already exists
       HTTP UNSUPPORTED_MEDIA_TYPE (415) when mediatype is not
       application/json
       HTTP BAD_REQUEST (400) when the json is not correct
```

### *4.1.2 GET /projects*

```
Syntax: HTTP://<server:port>/connect/v3/projects

Method: GET

Description:
  Returns a list of all projects containing all attributes

Return message (application/json):
[
       {
         "uuid":"<uuid>",
         "name":"<name>",
         "description":"<description>",
         "startdate":"<timestamp utc formatted as a string>",
         "enddate":"<timestamp utc formatted as a string or empty string >"
       }
]


Return code:
  HTTP OK (200)
  HTTP UNSUPPORTED_MEDIA_TYPE (415) when mediatype is not application/json,
       appliation/xml, or text/csv
```

### 4.1.3 *GET /projects/<projectid>*

```
Syntax: HTTP://<server:port>/connect/v3/projects/<projectid>

Method: GET

Description:
  Returns info about this specific project

Return message (application/json):
{
  "uuid":"<uuid>",
  "name":"<name>",
  "description":"<description>",
  "startdate":"<timestamp utc formatted as a string>",
  "enddate":"<timestamp utc formatted as a string or empty string >"
}

Return code:
  HTTP OK (200) if <projectid> existed
  HTTP BAD_REQUEST (400) when the projectid is not correct
  HTTP NOT_FOUND (404) if <projectid> does not exist
  HTTP UNSUPPORTED_MEDIA_TYPE (415) when mediatype is not application/json
       or application/xml
```

### 4.1.4 *PUT /projects/<projectid>*

```
Syntax: HTTP://<server:port>/connect/v3/projects/<projectid>

Method: PUT

Description:
  Update this specific project
  <name> is mandatory, must be unique and may not be empty
  <description> is optional
  <startdate> is mandatory and may not be empty
  <enddate> is optional

Payload (application/json):
{
  "name":"<name>",
  "description":"<description>",
  "startdate":"<timestamp utc formatted as a long>",
  "enddate":"<timestamp utc formatted as a long>"
}

Return code:
  HTTP OK (200) if <projectid> existed
  HTTP BAD_REQUEST (400) when the projectid or json is not correct
  HTTP CLIENT_ERROR_CONFLICT (409) when the project <name> already exists
  HTTP NOT_FOUND (404) if <projectid> does not exist
       HTTP UNSUPPORTED_MEDIA_TYPE (415) when mediatype is not
       application/json
```

### 4.1.5 *DEL /projects/<projectid>*

```
Syntax: HTTP://<server:port>/projects/<projectid>

Method: DEL

Description:
  Deletes this specific project

Return code:
  HTTP OK (200) if <projectid> existed
  HTTP BAD_REQUEST (400) when the projectid is not correct
  HTTP NOT_FOUND (404) if <projectid> does not exist
```

## 4.2 Resource projects/observerowners

### *4.2.1 POST /projects/<projectid>/observerowners*

```
Syntax: HTTP://<server:port>/connect/v3/projects/<projectid>/observerowners

Method: POST

Description:
  Creates a new observerowner in this project
  <name> is mandatory and must be unique
  <description> is optional (when missing it is made an empty string)
  The return is a redirect to the created observerowner

Payload (application/json):
{
  "name":"<name>",
  "description":"<description>"
}

Return code:
HTTP Redirect see other (303) to GET
     /projects/<projectid>/observerowner/<oberserverownerid>
HTTP CLIENT_ERROR_CONFLICT (409) when the observerowner <name> already exists
HTTP UNSUPPORTED_MEDIA_TYPE (415) when mediatype is not application/json
HTTP BAD_REQUEST (400) when the json is not correct
```

### *4.2.2 GET /projects/<projectid>/observerowners*

```
Syntax: HTTP://<server:port>/connect/v3/projects/<projectid>/observerowners

Method: GET

Description:
  Returns a list of all observerowners within this project
Return message (application/json):
[
       {
         "uuid":"<uuid>",
         "name":"<name>",
         "description":"<description>"
       }
]


Return code:
HTTP OK (200)
HTTP UNSUPPORTED_MEDIA_TYPE (415) when mediatype is not application/json,
     appliation/xml, or text/csv
```

### 4.2.3 GET /projects/<projectid>/observerowners/<observerownerid>

```
Syntax:
HTTP://<server:port>/connect/v3/projects/<projectid>/observerowners/<observer
ownerid>

Method: GET

Description:
  Returns info about this specific observerowner

Return message (application/json):
{
  "uuid":"<uuid>",
  "name":"<name>",
  "description":"<description>"
}

Return code:
  HTTP OK (200) if <projectid> and <observerownerid> exist
  HTTP BAD_REQUEST (400) when the <projectid> and/or <observerownerid> is
       not correct
  HTTP NOT_FOUND (404) if <projectid> and or <observerownerid> does not exist
  HTTP UNSUPPORTED_MEDIA_TYPE (415) when mediatype is not application/json
       or application/xml
```

### 4.2.4 PUT /projects/<projectid>/observerowners/<observerownerid>

```
Syntax:
HTTP://<server:port>/connect/v3/projects/<projectid>/observerowners/<observer
ownerid>

Method: PUT

Description:
  Update this specific observerowner
  <name> may not be empty and must be unique
  <description> is optional

Payload (application/json):
{
  "name":"<name>",
  "description":"<description>"
}

Return code:
  HTTP OK (200) if <projectid> and <observerownerid> exist
  HTTP BAD_REQUEST (400) when the <projectid> and/or <observerownerid> is
       not correct
  HTTP CLIENT_ERROR_CONFLICT (409) when the observerowner <name> already
       exists
  HTTP NOT_FOUND (404) if <projectid> and or <observerownerid> does not exist
  HTTP UNSUPPORTED_MEDIA_TYPE (415) when mediatype is not application/json
       or application/xml
```

### 4.2.5 DEL /projects/<projectid>/observerowners/<observerownerid>

```
Syntax:
HTTP://<server:port>/connect/v3/projects/<projectid>/observerowners/<observer
ownerid>

Method: DEL

Description:
       Deletes this specific observerowner. An observerowner can only be
       deleted if there are no other reference to this observerowner.

Return code:
  HTTP OK (200) if <projectid> and <observerownerid> exists and there are no
       references to this observerownerid
  HTTP CLIENT_ERROR_EXPECTATION_FAILED (417) if there are still references
       to this observerownerid.
  HTTP BAD_REQUEST (400) when the <projectid> or <observerownerid> is not
       correct
  HTTP NOT_FOUND (404) if <projectid> or <observerownerid> does not exist
  HTTP UNSUPPORTED_MEDIA_TYPE (415) when mediatype is not application/json
```

## 4.3 Resource projects/observers

### *4.3.1 POST /projects/<projectid>/rootobservers*

```
Syntax: HTTP://<server:port>/connect/v3/projects/<projected>/observers

Method: POST

Description:
  Create a new rootable observer in this specific project based on an
existing observertype
  Description, displayname and location are optional, the other elements are
mandatory
  <idusedbyowner> may not be empty and must be unique
  <displayname> is default the same as idusedbyowner
  <observertypeuuid> must be a valid UUID, must exist and must be rootable
  <observerowneruuid> must be a valid UUID and must exist
  The return is a redirect to the created observer

Payload (application/json):
{
  "idusedbyowner":"<idusedbyowner>",
  “displayname”,”<displayname>”,
  "description":"<description>",
  "location":"<location>",
  “timezone”:”<timezone>”,
  "observertypeuuid":"<observertypeuuid>",
  "observerowneruuid":"<observerowneruuid>"
}

Return code:
 HTTP Redirect see other (303) to GET
       /projects/<projectid>/observers/<observerid>
 HTTP BAD_REQUEST (400) when the projectid or json is not correct
 HTTP NOT_FOUND (404) if the provided <projectid> or <observertypeuuid> does
       not exist
 HTTP CLIENT_ERROR_CONFLICT (409) when the <idusedbyowner> already exists
 HTTP CLIENT_ERROR_CONFLICT (409) when the observertype <observertypeuuid>
       is not rootable
 HTTP UNSUPPORTED_MEDIA_TYPE (415) when mediatype is not application/json
```

### 4.3.2 *GET /projects/<projectid>/rootobservers*

```
Syntax: HTTP://<server:port>/connect/v3/projects/<projectid>/observers

Method: GET

Description:
  Returns a list of all available rootable observers

Return message (application/json):
[
       {
         “uuid”:”<uuid>”,
         "idusedbyowner":"<idusedbyowner>",
         “displayname”,”<displayname>”,
         "description":"<description>",
         "location":"<location>",
         “timezone”:”<timezone>”,
         “observerowner”:{
           "uuid":"<uuid>",
           "name":"<name>",
           "description":"<description>"
         },
         "observertype":{
           "uuid":"<uuid>",
           "name":"<name>",
             "measurementtype":"<measurementtype>",
           "description":"<description>",
           "specification":"<specification>",
           "rootable":"true",
           "measurementunit":{
             "uuid":"<uuid>",
             "quantity":"<quantity>",
             "unit":"<unit>",
             "description":"<description>"
           }
         }
       }
]

Return code:
 HTTP OK (200)
 HTTP BAD_REQUEST (400) when the projectid is not correct
 HTTP NOT_FOUND (404) if <projectid> does not exist
 HTTP UNSUPPORTED_MEDIA_TYPE (415) when mediatype is not application/json,
       appliation/xml
```

### 4.3.3 *GET /projects/<projectid>/observers/<observerid>*

```
Syntax:
HTTP://<server:port>/connect/v3/projects/<projectid>/observers/<observerid>

Method: GET

Description:
Returns information about this specific observer
including information about the observerowner, observertype and
measurementunit

Return message (application/json):
{
  "idusedbyowner":"<idusedbyowner>",
  “displayname”,”<displayname>”,
  "description":"<description>",
  "location":"<location>",
  “timezone”:”<timezone>”,
  “owner”:{
    "uuid":"<uuid>",
    "name":"<name>",
    "description":"<description>"
  },
  "observertype":{
    "uuid":"<uuid>",
    "name":"<name>",
      "measurementtype":"<measurementtype>",
    "description":"<description>",
    "specification":"<specification>",
    "rootable":"<rootable>",
    "measurementunit":{
      "uuid":"<uuid>",
      "quantity":"<quantity>",
      "unit":"<unit>",
      "description":"<description>"
    }
  }
}

Return code:
 HTTP OK (200) if <projectid> and <observerid> existed
 HTTP BAD_REQUEST (400) when the projectid or observerid is not correct
 HTTP NOT_FOUND (404) if <projectid> or <observerid> does not exist
 HTTP UNSUPPORTED_MEDIA_TYPE (415) when mediatype is not application/json
```

### 4.3.4 *PUT /projects/<projectid>/observers/<observerid>*

```
Syntax:
HTTP://<server:port>/connect/v3/projects/<projectid>/observers/<observerid>

Method: PUT

Description:
  Update this specific observer in this specific project
  Idusedbyowner is mandatory, the other elements are optional
  <idusedbyowner> may not be empty and must be unique
  observertypeuuid and observerowneruuid may not be updated

Payload (application/json):
{
  "idusedbyowner":"<idusedbyowner>",
  “displayname”,”<displayname>”,
  "description":"<description>",
  "location":"<location>",
  “timezone”:”<timezone>”,
}

Return code:
 HTTP OK (200) if <projectid> and <observerid> existed
 HTTP BAD_REQUEST (400) when the projectid or observerid is not correct
 HTTP CLIENT_ERROR_CONFLICT (409) when the <isusedbyowner> already exists
 HTTP NOT_FOUND (404) if <projectid> or <observerid> does not exist
 HTTP UNSUPPORTED_MEDIA_TYPE (415) when mediatype is not application/json
```

### 4.3.5 DEL /projects/<projectid>/observers/<observerid>

```
Syntax:
HTTP://<server:port>/connect/v3/projects/<projectid>/observers/<observerid>

Method: DEL

Description:
Deletes this specific observer from this specific project

Return code:
 HTTP OK (200) if <projectid> and <observerid> existed
 HTTP BAD_REQUEST (400) when the projectid or observerid is not correct
 HTTP NOT_FOUND (404) if <projectid> or <observerid> does not exist
 HTTP UNSUPPORTED_MEDIA_TYPE (415) when mediatype is not application/json
```

## 4.4 Resource projects/types/observertypes

### *4.4.1 POST /projects/<projectid>/types/observertypes*

```
Syntax:
HTTP://<server:port>/connect/v3/projects/<projecteid>/types/observertypes

Method: POST

Description:
Create a new observertype in this specific project
description and specification are optional, the other elements are mandatory
  <name> may not be empty and must be unique
  <measurementtype> is an enumeration of <double, long, string, boolean,
url>
  <rootable> is a boolean value (true/false)
  <measurementunituuid> must be a valid UUID and must exist
  The return is a redirect to the created observer

Payload (application/json):
{
  "name":"<name>",
  "measurementtype":"<measurementtype>",
  "description":"<description>",
  "specification":"<specification>",
  "rootable":"<rootable>",
  "measurementunituuid":"<measurementunituuid>",
}

Return code:
 HTTP Redirect see other (303) to GET
       /projects/<projectid>/types/observertypes/<observertypeid>
 HTTP BAD_REQUEST (400) when the projectid or json is not correct
 HTTP NOT_FOUND (404) if the provided <projectid> or <measurementunituuid>
       does not exist
 HTTP UNSUPPORTED_MEDIA_TYPE (415) when mediatype is not application/json
 HTTP CLIENT_ERROR_CONFLICT (409) when the <name> already exists
```

### *4.4.2 GET /projects/<projectid>/types/observertypes*

```
Syntax:
HTTP://<server:port>/connect/v3/projects/<projectid>/types/observertypes

Method: GET

Description:
  Returns a list of all available observertypes

Return message (application/json):
[
       {
         "uuid":"<uuid>",
         "name":"<name>",
         "measurementtype";"<measurementtype>",
         "description":"<description>",
         "specification":"<specification>",
         "rootable":"<rootable>",
         “measurementunit”:{
                "uuid":"<uuid>",
                "quantity":"<quantity>",
                "unit":"<unit>",
                "description":"<description>"
              }
       }
]

Return code:
  HTTP OK (200)
  HTTP BAD_REQUEST (400) when the projectid is not correct
  HTTP NOT_FOUND (404) if <projectid> does not exist
  HTTP UNSUPPORTED_MEDIA_TYPE (415) when mediatype is not application/json,
       appliation/xml
```

### 4.4.3 *GET /projects/<projectid>/types/observertypes/<observertypeid>*

```
Syntax:
HTTP://<server:port>/connect/v3/projects/<projectid>/
       types/observertypes/<observertypeid>

Method: GET

Description:
  Returns info about this observertype in this specific project

Return message (application/json):
{
  "uuid":"<uuid>",
  "name":"<name>",
  "measurementtype";"<measurementtype>",
  "description":"<description>",
  "specification":"<specification>",
  "rootable":"<rootable>",
  “measurementunit”:{
         "uuid":"<uuid>",
         "quantity":"<quantity>",
         "unit":"<unit>",
         "description":"<description>"
       }
}

Return code:
 HTTP OK (200) if <projectid> existed
 HTTP BAD_REQUEST (400) when the projectid and/or observertypeid is not
       correct
 HTTP NOT_FOUND (404) if <projectid> and/or observertypeid does not exist
 HTTP UNSUPPORTED_MEDIA_TYPE (415) when mediatype is not application/json or
       application/xml
```

### 4.4.4 *PUT /projects/<projectid>/types/observertypes/<observertypeid>*

```
Syntax: HTTP://<server:port>/connect/v3/projects/<projectid>/
       types/observertypes/<observertypeid>

Method: PUT

Description:
 Update this specific observertype in this specific project
 description and specification are optional, the other elements are
 mandatory
 <name> may not be empty and must be unique measurementunituuid may not be
 updated

Payload (application/json):
{
  "name":"<name>",
  "measurementtype";"<measurementtype>",
  "description":"<description>",
  "specification":"<specification>",
  "rootable":"<rootable>"
}

Return code:
 HTTP OK (200) if <projectid> and <observertypeid> existed
 HTTP BAD_REQUEST (400) when the projectid or observertypeid is not correct
 HTTP NOT_FOUND (404) if <projectid> or <observertypeid> does not exist
 HTTP UNSUPPORTED_MEDIA_TYPE (415) when mediatype is not application/json
 HTTP CLIENT_ERROR_CONFLICT (409) when the <name> already exists
```

### 4.4.5 DEL /projects/<projectid>/types/observertypes/<observertypeid>

```
Syntax: HTTP://<server:port>/connect/v3/projects/<projectid>/
        types/observertypes/<observertypeid>

Method: DEL

Description:
   Deletes this specific observertype from this specific project

Return code:
  HTTP OK (200) if <projectid> and <observertypeid> existed
  HTTP BAD_REQUEST (400) when the projectid or observertypeid is not correct
  HTTP NOT_FOUND (404) if <projectid> or <observertypeid> does not exist
  HTTP UNSUPPORTED_MEDIA_TYPE (415) when mediatype is not application/json
```

## 4.5 Resource projects/types/measurementunits

### *4.5.1 POST /projects/<projectid>/types/measurementunits*

```
Syntax:
HTTP://<server:port>/connect/v3/projects/<projected>/types/measurementunits

Method: POST

Description:
  Create a new measurementunit in this specific project
  description is optional, the other elements are mandatory
  <quantity> may not be empty
  <unit> may not be empty
  The return is a redirect to the created observer

Payload (application/json):
{
  "quantity":"<quantity>",
  "unit":"<unit>",
  "description":"<description>"
}

Return code:
 HTTP Redirect see other (303) to GET
       /projects/<projectid>/types/measurementunits/
         <measurementunitid>
 HTTP BAD_REQUEST (400) when the projectid or json is not correct
 HTTP NOT_FOUND (404) if the provided <projectid> does not exist
 HTTP UNSUPPORTED_MEDIA_TYPE (415) when mediatype is not application/json
```

### *4.5.2 GET /projects/<projectid>/types/measurementunits*

```
Syntax:
HTTP://<server:port>/connect/v3/projects/<projectid>/types/measurementunits

Method: GET

Description:
  Returns a list of all available measurementunits

Return message (application/json):
[
       {
         "uuid":"<uuid>",
         "quantity":"<quantity>",
         "unit":"<unit>",
         "description":"<description>"
       }
]

Return code:
 HTTP OK (200)
 HTTP BAD_REQUEST (400) when the projectid is not correct
 HTTP NOT_FOUND (404) if <projectid> does not exist
 HTTP UNSUPPORTED_MEDIA_TYPE (415) when mediatype is not application/json,
       appliation/xml
```

### 4.5.3 GET /projects/<projectid>/types/measurementunits/<measurementunitid>

```
Syntax: HTTP://<server:port>/connect/v3/projects/<projectid>/
       types/measurementunits/<measurementunitid>

Method: GET

Description:
  Returns info about this measurementunit in this specific project

Return message (application/json):
{
  "uuid":"<uuid>",
  "quantity":"<quantity>",
  "unit":"<unit>",
  "description":"<description>"
}

Return code:
 HTTP OK (200) if <projectid> existed
 HTTP BAD_REQUEST (400) when the projectid and/or measurementunitid is not
       correct
 HTTP NOT_FOUND (404) if <projectid> and/or measurementunitid does not exist
 HTTP UNSUPPORTED_MEDIA_TYPE (415) when mediatype is not application/json or
       application/xml
```

### 4.5.4 PUT /projects/<projectid>/types/measurementunits/<measurementunitid>

```
Syntax: HTTP://<server:port>/connect/v3/projects/<projectid>/
       types/measurementunits/<measurementunitid>

Method: PUT

Description:
  Update this specific measurementunit in this specific project
  description is optional, the other elements are mandatory
  <quantity> may not be empty
  <unit> may not be empty

Payload (application/json):
{
  "quantity":"<quantity>",
  "unit":"<unit>",
  "description":"<description>"
}

Return code:
 HTTP OK (200) if <projectid> and <measurementunitid> existed
 HTTP BAD_REQUEST (400) when the projectid or measurementunitid is not
       correct
 HTTP NOT_FOUND (404) if <projectid> or <measurementunitid> does not exist
 HTTP UNSUPPORTED_MEDIA_TYPE (415) when mediatype is not application/json
```

### 4.5.5 DEL /projects/<projectid>/types/measurementunits/<measurementunitid>

```
Syntax: HTTP://<server:port>/connect/v3/projects/<projectid>/
       types/measurementunits/<measurementunitid>

Method: DEL

Description:
  Deletes this specific measurementunit from this specific project

Return code:
 HTTP OK (200) if <projectid> and <measurementunitid> existed
 HTTP BAD_REQUEST (400) when the projectid or measurementunitid is not
       correct
 HTTP NOT_FOUND (404) if <projectid> or <measurementunitid> does not exist
 HTTP UNSUPPORTED_MEDIA_TYPE (415) when mediatype is not application/json
```

## 4.6 Resource projects/rootgroups

### *4.6.1 POST /projects/<projectid>/rootgroups*

```
Syntax: HTTP://<server:port>/connect/v3/projects/<projectid>/rootgroups

Method: POST

Description:
 Create a new root group in this specific project
 Idusedbyowner is mandatory and may not be empty observerowneruuid and
 rootgrouptypeuuid are mandatory <rootgrouptypeuuid> will be used the create
 all specified observers and groups

 Description, displayname and location are optional
 The default value voor displayname is idusedbywoner the return is a
 redirect to the created rootgroup

payload (application/json):
{
  “idusedbyowner”:”<idusedbyowner>”,
       "displayname":"<displayname>",
  "description":"<description>",
  "location":"<location>",
       “observerowneruuid”:”<observerownerid>”,
  "rootgrouptypeuuid":"<rootgrouptypeuuid>"
}

Return code:
 HTTP Redirect see other (303) to GET
       /projects/<projectid>/observergroups/<rootgroupid>
 HTTP BAD_REQUEST (400) when the projectid or json is not correct
 HTTP NOT_FOUND (404) if <projectid> or <rootgrouptypeuuid> or
       <observerowneruuid> does not exist
 HTTP UNSUPPORTED_MEDIA_TYPE (415) when mediatype is not application/json
 HTTP BAD_REQUEST (400) when the <rootgrouptypeuuid> is not rooatble
```

### 4.6.2 GET /projects/<projectid>/rootgroups

```
Syntax: HTTP://<server:port>/connect/v3/projects/<projectid>/rootgroups

Method: GET

Description:
  Returns a list of all available rootgroups for this specific project

Return message (application/json):
[
       {
         "uuid":"<uuid>",
         “isusedbyowner”:”<idusedbyowner>”,
         "displayname":"<displayname>",
         "description":"<description>",
         "location":"<location>",
         “observerowner”:{
           "uuid":"<uuid>",
           "name":"<name>",
           "description":"<description>"
         },
         "rootgrouptype": {
           “uuid”:”<uuid>”,
           “name”:”<name>”,
           "description":"<description>",
           “rootable”:”<rootable>”
         }
       }
]

Return code:
 HTTP OK (200)
 HTTP BAD_REQUEST (400) when the projectid is not correct
 HTTP NOT_FOUND (404) if <projectid> does not exist
 HTTP UNSUPPORTED_MEDIA_TYPE (415) when mediatype is not application/json
```

### 4.6.3 PUT /projects/<projectid>/rootgroups/<rootgroupid>

```
Syntax:
HTTP://<server:port>/connect/v3/projects/<projectid>/rootgroups/<rootgroupid>

Method: PUT

Description:
 Update this specific root group in this specific project
 Idusedbyowner is mandatory and may not be empty
 Description, displayname and location are optional
 The default value voor displayname is idusedbywoner
 The observerowneruuid and rootgrouptypeuuid cannot be updates

payload (application/json):
{
  “idusedbyowner”:”<idusedbyowner>”,
  "displayname":"<displayname>",
  "description":"<description>",
  "location":"<location>"
}

Return code:
 HTTP OK (200) if <projectid> and <rootgroupid> exist
 HTTP BAD_REQUEST (400) when the <projectid> or <rootgroupid> is not correct
 HTTP NOT_FOUND (404) if <projectid> or <rootgroupid> does not exist
 HTTP UNSUPPORTED_MEDIA_TYPE (415) when mediatype is not application/json
```

### 4.6.4 *DEL /projects/<projectid>/rootgroups/<rootgroupid>*

```
Syntax:
HTTP://<server:port>/connect/v3/projects/<projectid>/rootgroups/<rootgroupid>
Method: DEL

Description:
   Deletes this specific rootgroup from this specific project

Return code:
  HTTP OK (200) if <projectid> and <rootgroupid> exist
  HTTP BAD_REQUEST (400) when the <projectid> or <rootgroupid> is not correct
  HTTP NOT_FOUND (404) if <projectid> or <rootgroupid> does not exist
  HTTP UNSUPPORTED_MEDIA_TYPE (415) when mediatype is not application/json
```

## 4.7 Resource projects/logicalgroups

### *4.7.1 POST /projects/<projectid>/logicalgroups*

```
Syntax: HTTP://<server:port>/connect/v3/projects/<projectid>/logicalgroups

Method: POST

Description:
 Create a new logical group in this specific project
 Idusedbyowner and observerowneruuid are mandatory
 Description, displayname and location are optional
 The default value voor displayname is idusedbywoner
 The return is a redirect to the created logicalgroup

payload (application/json):
{
  “idusedbyowner”:”<idusedbyowner>”,
       "displayname":"<displayname>",
  "description":"<description>",
  "location":"<location>",
       “observerowneruuid”:”<observerownerid>”
}

Return code:
 HTTP Redirect see other (303) to GET
       /projects/<projectid>/observergroups/<logicalgroupid>
 HTTP BAD_REQUEST (400) when the projectid or json is not correct
 HTTP NOT_FOUND (404) if <projectid> or <observerowneruuid> does not exist
 HTTP UNSUPPORTED_MEDIA_TYPE (415) when mediatype is not application/json
```

### *4.7.2 GET /projects/<projectid>/logicalgroups*

```
Syntax: HTTP://<server:port>/connect/v3/projects/<projectid>/logicalgroups

Method: GET

Description:
  Returns a list of all available logicalgroups for this specific project

Return message (application/json):
[
       {
         "uuid":"<uuid>",
         "displayname":"<displayname>",
         "description":"<description>",
         "location":"<location>",
              “idusedbyowner”:”<idusedbyowner>”,
         “observerowner”:{
           "uuid":"<uuid>",
           "name":"<name>",
           "description":"<description>"
         }
       }
]

Return code:
 HTTP OK (200)
 HTTP BAD_REQUEST (400) when the projectid is not correct
 HTTP NOT_FOUND (404) if <projectid> does not exist
 HTTP UNSUPPORTED_MEDIA_TYPE (415) when mediatype is not application/json
```

### 4.7.3 *PUT /projects/<projectid>/logicalgroups/<logicalgroupid>*

```
Syntax: HTTP://<server:port>/connect/v3/projects/<projectid>/logicalgroups/
<logicalgroupid>

Method: PUT

Description:
 Update this specific logical group in this specific project
 Idusedbyowner is mandatory and may not be empty
 Description, displayname and location are optional
 The default value voor displayname is idusedbywoner
 The observerowneruuid cannot be updates

payload (application/json):
{
  “idusedbyowner”:”<idusedbyowner>”,
       "displayname":"<displayname>",
  "description":"<description>",
  "location":"<location>"
}

Return code:
 HTTP OK (200) if <projectid> and <logicalgroupid> existed
 HTTP BAD_REQUEST (400) when the <projectid> or <logicalgroupid> is not
       correct
 HTTP NOT_FOUND (404) if <projectid> or <logicalgroupid> does not exist
 HTTP UNSUPPORTED_MEDIA_TYPE (415) when mediatype is not application/json
```

### 4.7.4 *DEL /projects/<projectid>/logicalgroups/<logicalgroupid>*

```
Syntax:
HTTP://<server:port>/connect/v3/projects/<projectid>/logicalgroups/<logicalgr
oupid>

Method: DEL

Description:
  Deletes this specific logicalgroup from this specific project

Return code:
 HTTP OK (200) if <projectid> and <logicalgroupid> existed
 HTTP BAD_REQUEST (400) when the <projectid> or <logicalgroupid> is not
       correct
 HTTP NOT_FOUND (404) if <projectid> or <logicalgroupid> does not exist
 HTTP UNSUPPORTED_MEDIA_TYPE (415) when mediatype is not application/json
```

### 4.7.5 *PUT /projects/<projectid>/logicalgroups/<logicalgroupid>/observergroups/ <observergroupid>*

```
Syntax: HTTP://<server:port>/connect/v3/projects/<projectid>/
        logicalgroups/<logicalgroupid>/observergroups/<observergroupid>

Method: PUT

Description:
   Add a new child observergroup (observergroupid) to this logicalgroup
   <locationwithingroup> is optional

payload (application/json):
{
   "locationwithingroup":"<locationwithingroup>"
}

Return code:
  HTTP OK (200) if <projectid> and <logicalgroupid> and <observergroupid>
        exist
  HTTP NOT_FOUND (404) if <projectid> or <logicalgroupid> or
        <observergroupid> not exist
  HTTP UNSUPPORTED_MEDIA_TYPE (415) when mediatype is not application/json
```

### 4.7.6 *DEL /projects/<projectid>/logicalgroups/<logicalgroupid>/observergroups/ <observergroupid>*

```
Syntax: HTTP://<server:port>/connect/v3/projects/<projectid>/
logicalgroups/<logicalgroupid>/observergroups/<observergroupid>

Method: DEL

Description:
   Remove this child observergroup (observergroupid) from this logicalgroup

Return code:
  HTTP OK (200) if <projectid> and <logicalgroupid> and <observergroupid>
        exist
  HTTP NOT_FOUND (404) if <projectid> or <logicalgroupid> or
        <observergroupid> not exist
  HTTP UNSUPPORTED_MEDIA_TYPE (415) when mediatype is not application/json
```

### 4.7.7 *PUT /projects/<projectid>/logicalgroups/<logicalgroupid>/ observers/<observerid>*

```
Syntax: HTTP://<server:port>/connect/v3/projects/<projectid>/
logicalgroups/<logicalgroupid>/observers/<observerid>

Method: PUT

Description:
   Add a new child observer (observerid) to this logicalgroup
   <locationwithingroup> is optional

payload (application/json):
{
   "locationwithingroup":"<locationwithingroup>"
}

Return code:
  HTTP OK (200) if <projectid> and <logicalgroupid> and <observerid> exist
  HTTP NOT_FOUND (404) if <projectid> or <logicalgroupid> or <observerid> not
        exist
  HTTP UNSUPPORTED_MEDIA_TYPE (415) when mediatype is not application/json
```

### 4.7.8 DEL /projects/<projectid>/logicalgroups/<logicalgroupid>/ observers/<observerid>

```
Syntax: HTTP://<server:port>/connect/v3/projects/<projectid>/
        logicalgroups/<logicalgroupid>/observers/<observerid>

Method: DEL

Description:
   Remove this child observer (observerid) from this logicalgroup

Return code:
  HTTP OK (200) if <projectid> and <logicalgroupid> and <observerid> exist
  HTTP NOT_FOUND (404) if <projectid> or <logicalgroupid> or <observerid> not
       exist
  HTTP UNSUPPORTED_MEDIA_TYPE (415) when mediatype is not application/json
```

## 4.8 Resource projects/observergroups

### *4.8.1 GET /projects/<projectid>/observergroups/<observergroupid>*

```
Syntax:
HTTP://<server:port>/connect/v3/projects/<projectid>/observergroups/<observer
groupid>

Method: GET

Description:
 Returns information about this specific observergroup in this specific
 project.

 These can be rootgroups, logicalgroups and/or internalgroups
        including:
  - list of all including observers
  - list of all underlying child observergroups

Return message (application/json):
{
   "uuid":"<uuid>",
   “idusedbyowner”:”<idusedbyowner>”,
   "displayname":"<displayname>",
   "description":"<description>",
   "location":"<location>",
        “observerowner”:{
          "uuid":"<uuid>",
          "name":"<name>",
          "description":"<description>"
        },
   "rootgrouptype": {
     “uuid”:”<uuid>”,
     “name”:”<name>”,
     "description":"<description>",
     “rootable”:”<rootable>”
   },
   "observers":[
                {
        "uuid":"<uuid>"
                        "idusedbyowner":"<idusedbyowner>",
        “displayname”,”<displayname>”,
        "description":"<description>",
        "location":"<location>",
        “timezone”:”<timezone>”,
        “observerownerid”:”<observerownerid>”,
        "observertypeid":”<observertypeid>”,
        ”observeringroup”:{
          “uuid”:”<uuid>”,
                                “observernamewithingroup”:””,
          “locationwithingroup”:”<locationwithingroup>”
                                “logicalocation”:””,
                                “logicallocationdimensions”:””
        }
      }
         ],
   "observergroups": [
                {
        <!-- Here This entire json again for all child observergroups-->
        ”groupingroup”:{
                                “groupnamewithingroup”:””,
          “locationwithingroup”:”<locationwithingroup>”,
                                “logicalocation”:””,
                                “logicallocationdimensions”:””
        }
      }
         ]
}

Return code:
 HTTP OK (200)
```

```
 HTTP BAD_REQUEST (400) when the projectid or observerlgroupid is not
       correct
 HTTP NOT_FOUND (404) if <projectid> or <observergroupid> does not exist
 HTTP UNSUPPORTED_MEDIA_TYPE (415) when mediatype is not application/json
```

### *4.8.2 PUT /projects/<projectid>/observergroups/<observergroupid>*

```
Syntax:
HTTP://<server:port>/connect/v3/projects/<projectid>/observergroups/
<observergroupid>

Method: PUT

Description:
  Update a specific observergroup in this specific project
  description and location are optional
  displayname is mandatory and may not be empty
  observergrouptypeuuid cannot be updated.

payload (application/json):
{
  "displayname":"<displayname>",
  "description":"<description>",
  "location":"<location>"
}

Return code:
 HTTP OK (200) if <projectid> and <observergroupid> existed
 HTTP BAD_REQUEST (400) when the projectid or json is not correct
 HTTP NOT_FOUND (404) if <projectid> or <observergroupid> not exist
 HTTP UNSUPPORTED_MEDIA_TYPE (415) when mediatype is not application/json
```

## 4.9 Resource projects/types/observergrouptypes

### *4.9.1 POST /projects/<projectid>/types/observergrouptypes*

```
Syntax:
HTTP://<server:port>/connect/v3/projects/<projectid>/types/observergrouptypes

Method: POST

Description:
  Create a new observergrouptype in this specific project
 <name> and <rootable> are mandatory
 description is optional
 the return is a redirect to the created observergrouptype

payload (application/json):
{
  "name":"<name>",
  "description":"<description>",
  "rootable":"<rootable>"
}

Return code:
 HTTP Redirect see other (303) to GET
        /projects/<projectid>/observergrouptype/<observergrouptypeid>
 HTTP BAD_REQUEST (400) when the projectid or json is not correct
 HTTP NOT_FOUND (404) if <projectid> does not exist
 HTTP UNSUPPORTED_MEDIA_TYPE (415) when mediatype is not application/json
```

### *4.9.2 GET /projects/<projectid>/types/observergrouptypes*

```
Syntax:
HTTP://<server:port>/connect/v3/projects/<projectid>/types/observergrouptypes

Method: GET

Description:
 Returns a list of all available observergrouptypes for this specific
 project

Return message (application/json):
[
       {
         "uuid":"<uuid>",
         "name":"<name>",
         "description":"<description>",
         "rootable":"<rootable>"
       }
]

Return code:
 HTTP OK (200)
 HTTP BAD_REQUEST (400) when the projectid is not correct
 HTTP NOT_FOUND (404) if <projectid> does not exist
 HTTP UNSUPPORTED_MEDIA_TYPE (415) when mediatype is not application/json
```

### 4.9.3 GET /projects/<projectid>/types/observergrouptypes /<observergrouptypeid>

```
Syntax: HTTP://<server:port>/connect/v3/projects/<projectid>/
       types/observergrouptypes/<observergrouptypeid>

Method: GET

Description:
 Returns information about this specific observergrouptype in this specific
 project.These can be rootgroups, logicalgroups and/or internalgroups
 including:
  - list of all including observertypes
  - list of all underlying child observergroups

Return message (application/json):
{
  "uuid":"<uuid>",
  "name":"<name>",
  "description":"<description>",
  "rootable":"<rootable>",
  "observertypes":[{
    "uuid":"<uuid>",
    "name":"<name>",
    "measurementtype";"<measurementtype>",
    "description":"<description>",
    "specification":"<specification>",
    "rootable":"<rootable>"
  }],
  "observergrouptypes": [{
         "uuid":"<uuid>",
         "name":"<name>",
         "description":"<description>",
         "rootable":"<rootable>"
  }]
}

Return code:
 HTTP OK (200)
 HTTP BAD_REQUEST (400) when the <projectid> or <observergrouptypeid> is not
       correct
 HTTP NOT_FOUND (404) if <projectid> or <observergrouptypeid> does not exist
 HTTP UNSUPPORTED_MEDIA_TYPE (415) when mediatype is not application/json
```

### 4.9.4 PUT /projects/<projectid>/types/observergrouptypes/<observergrouptypeid>

```
Syntax: HTTP://<server:port>/connect/v3/projects/<projectid>/
       types/observergrouptypes/<observergrouptypeid>

Method: PUT

Description:
 Update a specific observergrouptype in this specific project
 description is optional name and rootable are mandatory and may not be
 empty

payload (application/json):
{
  "name":"<name>",
  "description":"<description>",
  "rootable":"<rootable>"
}

Return code:
 HTTP OK (200) if <projectid> and <observergrouptypeid> existed
 HTTP BAD_REQUEST (400) when the projectid or json is not correct
 HTTP NOT_FOUND (404) if <projectid> or <observergrouptypeid> not exist
 HTTP UNSUPPORTED_MEDIA_TYPE (415) when mediatype is not application/json
```

### 4.9.5 *PUT /projects/<projectid>/types/observergrouptypes/<observergrouptypeid>/ observergrouptypes/<observergrouptypeid>*

```
Syntax: HTTP://<server:port>/connect/v3/projects/<projectid>/
       types/observergrouptypes/<observergrouptypeid>/
       observergrouptypes/<observergrouptypeid>

Method: PUT

Description:
 Add a new child observergrouptype (observergrouptypeid) to this
 observergrouptype,  <groupnamewithingroup>, <logicallocation> and
 <logicallocationdimensions> are optional

payload (application/json):
{
   "groupnamewithingroup":"<groupnamewithingroup>",
   “logicallocation”:”<logicallocation>”,
        “logicallocationdimensions”:”<logicallocationdimensions>”,

}

Return code:
 HTTP OK (200) if <projectid> and <observergrouptypeid> and
       <observergrouptypeid> exist
 HTTP NOT_FOUND (404) if <projectid> or <observergrouptypeid> or
       <observergrouptypeid> not exist
 HTTP UNSUPPORTED_MEDIA_TYPE (415) when mediatype is not application/json
```

### 4.9.6 *DEL /projects/<projectid>/types/observergrouptypes/<observergrouptypeid>/ observergrouptypes/<observergrouptypeid>*

```
Syntax: HTTP://<server:port>/connect/v3/projects/<projectid>/types/
       observergrouptypes/<observergrouptypeid>/observergrouptypes/
       <observergrouptypeid>
Method: DEL

Description:
 Remove this child observergrouptype (observergrouptypeid) from this
 observergrouptype

Return code:
 HTTP OK (200) if <projectid> and <observergrouptypeid> and
       <observergrouptypeid> exist
 HTTP NOT_FOUND (404) if <projectid> or <observergrouptypeid> or
       <observergrouptypeid> not exist
 HTTP UNSUPPORTED_MEDIA_TYPE (415) when mediatype is not application/json
```

### 4.9.7 *PUT /projects/<projectid>/types/observergrouptypes/<observergrouptypeid>/observertypes/<observertypeid>*

```
Syntax: HTTP://<server:port>/connect/v3/projects/<projectid>/
       types/observergrouptypes/<observergrouptypeid>/observertypes/<observe
       rtypeid>

Method: PUT

Description:
 Add a new child observertype (observertypeid) to this observergrouptype
 <groupnamewithingroup>, <logicallocation> and <logicallocationdimensions>
 are optional

payload (application/json):
{
   "groupnamewithingroup":"<groupnamewithingroup>",
   “logicallocation”:”<logicallocation>”,
        “logicallocationdimensions”:”<logicallocationdimensions>”,

}

Return code:
 HTTP OK (200) if <projectid> and <observergrouptypeid> and <observertypeid>
       exist
 HTTP NOT_FOUND (404) if <projectid> or <observergrouptypeid> or
       <observertypeid> not exist
 HTTP UNSUPPORTED_MEDIA_TYPE (415) when mediatype is not application/json
```

### 4.9.8 *DEL /projects/<projectid>/types/observergrouptypes/<observergrouptypeid>/observertypes/<observertypeid>*

```
Syntax: HTTP://<server:port>/connect/v3/projects/<projectid>/
       types/observergrouptypes/<observergrouptypeid>/observertypes/<observe
       rtypeid>

Method: DEL

Description:
   Remove this child observertype (observertypeid) from this
   observergrouptype

Return code:
 HTTP OK (200) if <projectid> and <observergrouptypeid> and <observertypeid>
       exist
 HTTP NOT_FOUND (404) if <projectid> or <observergrouptypeid> or
       <observertypeid> not exist
 HTTP UNSUPPORTED_MEDIA_TYPE (415) when mediatype is not application/json
```

## 4.10 Resource projects/timelines/observers/measurements

### *4.10.1 GET /projects/<projectid>/timelines/core/observers/<observerid>/measurements/<fromtimestamp>/<totimestamp>/<timezonecontinent>/<timezonecity>*

```
Syntax:
http://<server:port>/connect/v3/projects/<projectname>/timelines/core/
       observers/<observerid>/measurements/<fromtimestamp>/<totimestamp>
       ?maxcount=<maxcount>&returntimestampformat=<returntimestampformat>
     or
       http://<server:port>/projects/<projectname>/timelines/core/
       observers/<observerid>/measurements/<fromtimestamp>/<totimestamp>/
       <timezonecontinent>/<timezonecity>
       ?maxcount=<maxcount>&returntimestamptype=<returntimestamptype>

Method: GET

Description:
 Returns a set of measurements from this project, from this observer,
 within the core timeline, within the given time range [fromtimestamp ..
 totimestamp]

 Optionally a timezone (timezonecontinent>/<timezonecity>) can be specified,
 resulting in measurements with this timezone if the returntimestampformat
 is string. If the returntimestampformat is long they will be utc
 timestamps.
 If the timezone is specified, a totimestamp MUST also be specified. If all
 measurements are required, use a very large number (long time into the
 future).

 format of <fromtimestamp> and <totimestamp> is
  - long epoch time in utc
  - string in YYYY-MM-DDTHH:MM:SS.sss in utc time or in <timezonecontinent>/
      <timezonecity> time

Parameters:
  optional maxcount:
    The maximum number of measurements that must be returned.
    The absolute maximum is 1000 measurements in one request.
    To get the next set of measurements, get the measurements with the
    fromtimestamp the same as the last timestamp in the previous request
  optional returntimestamptype:
    The format the measurements timestamps must be returned.
    options are:
    - "long" (default)
    - "string"

Return message general:
  Returned timestamps are formatted with in utc, or in the optionally
  specified timezone.
  Returned measurements always have interpolationmethod none.

Return message (application/json):
  {
    "observerid","<observerid>",
    "timelinename":"core",
    "timezone":"<timezonecontinent>/<timezonecity>",
    "timestamptype":"<timestamptype>",
    "measurementtype":"<measurementtype>",
    "interpolationmethod":"none",
    "measurements":[ {
      "timestamp":"<timestamp>",
      "value":"<value>"
    }]
  }

Example:
- Return message (application/json):
  http://anysense.tno.nl:1234/connect/v3/projects/testproject/
  timelines/core/observers/f47ac10b-58cc-4372-a567-0e02b2c3d479/
  measurements/1278902215000/1278902516000/europe/Amsterdam
  ?maxcount=100&returntimestamptype=string
```

```
  {
    "observerid","f47ac10b-58cc-4372-a567-0e02b2c3d479",
    "timelinename":"core",
    "timezone":"europe/amsterdam",
    "timestamptype":"string",
    "measurementtype":"double",
    "interpolationmethod":"none",
    "measurements":[
    {
      "timestamp":"2010-07-12T02:36:55.0+01:00",
      "value":1054.0
    },
    {
      "timestamp":"2010-07-12T02:41:56.0+01:00",
      "value":1054.0
    }]
  }

Return code:
  HTTP OK (200)
  HTTP BAD_REQUEST (400) when the projectid, observerid, fromtimestamp,
         totimestamp, timezonecontinent, timezonecity or returntimestamptype
         is not correct
  HTTP NOT_FOUND (404) if the projectid, observerid, fromtimestamp,
         totimestamp, timezonecontinent, timezonecity or returntimestamptype
         does not exist
  HTTP UNSUPPORTED_MEDIA_TYPE (415) when mediatype is not application/json
```

### 4.10.2 *GET /projects/<projectid>/timelines/core/observers/<observerid>/measurements/position/<timeposition>/<timezonecontinent>/<timezonecity>*

```
Syntax:
http://<server:port>/connect/v3/projects/<projectname>/timelines/core/
       observers/<observerid>/measurements/position/<timepostion>
       ?returntimestampformat=<returntimestampformat>
     or
       http://<server:port>/connect/v3/projects/<projectname>/timelines/core/
       observers/<observerid>/measurements/position/<timeposition>/
       <timezonecontinent>/<timezonecity>
       ?returntimestamptype=<returntimestamptype>

Method: GET

Description:
 Returns the oldest/newest measurement from this project, from this
 observer,within the core timeline

 timeposition indicates the desired position in time. Possible values are
 first or last

 Optionally a timezone (timezonecontinent>/<timezonecity>) can be specified,
 resulting in measurements with this timezone if the returntimestampformat
 is string. If the returntimestampformat is long they will be utc
 timestamps.

Parameters:
  optional returntimestamptype:
    The format the measurements timestamps must be returned.
    options are:
    - "long" (default)
    - "string"

Return message general:
  Returned timestamps are formatted with in utc, or in the optionally
  specified timezone.
  Returned measurements always have interpolationmethod none.

Return message (application/json):
  {
    "observerid","<observerid>",
    "timelinename":"core",
    "timezone":"<timezonecontinent>/<timezonecity>",
    "timestamptype":"<timestamptype>",
    "measurementtype":"<measurementtype>",
    "interpolationmethod":"none",
    "measurements":[ {
      "timestamp":"<timestamp>",
      "value":"<value>"
    }]
  }

Example:
- Return message (application/json):
       http://anysense.tno.nl:1234/connect/v3/projects/testproject/
       timelines/core/observers/f47ac10b-58cc-4372-a567-0e02b2c3d479/
       measurements/position/first/europe/amsterdam
       ?returntimestamptype=string

  {
    "observerid","f47ac10b-58cc-4372-a567-0e02b2c3d479",
    "timelinename":"core",
    "timezone":"europe/amsterdam",
    "timestamptype":"string",
    "measurementtype":"double",
    "interpolationmethod":"none",
    "measurements":[
    {
      "timestamp":"2010-07-12T02:36:55.0+01:00",
      "value":1054.0
    }]
  }
```

```
Return code:
 HTTP OK (200)
 HTTP BAD_REQUEST (400) when the projectid, observerid, fromtimestamp,
       totimestamp, timezonecontinent, timezonecity or returntimestamptype
       is not correct
 HTTP NOT_FOUND (404) if the projectid, observerid, fromtimestamp,
       totimestamp, timezonecontinent, timezonecity or returntimestamptype
       does not exist
 HTTP UNSUPPORTED_MEDIA_TYPE (415) when mediatype is not application/json
```

## 4.11 Resource timezones

### *4.11.1 GET /timezones*

```
Syntax: HTTP://<server:port>/connect/v3/timezones

Method: GET

Description:
   Returns a list of all available timezones

Return message (application/json):
[
  { "timezone":"<timezone>" }
]

Return code
 HTTP OK (200)
 HTTP UNSUPPORTED_MEDIA_TYPE (415) when mediatype is not application/json
```

# 5 Appendix Timezones

**Format: timezonecontinent/timezonecity**

ACT
AET
Africa/Abidjan
Africa/Accra
Africa/Addis_Ababa
Africa/Algiers
Africa/Asmara
Africa/Asmera
Africa/Bamako
Africa/Bangui
Africa/Banjul
Africa/Bissau
Africa/Blantyre
Africa/Brazzaville
Africa/Bujumbura
Africa/Cairo
Africa/Casablanca
Africa/Ceuta
Africa/Conakry
Africa/Dakar
Africa/Dar_es_Salaam
Africa/Djibouti
Africa/Douala
Africa/El_Aaiun
Africa/Freetown
Africa/Gaborone
Africa/Harare
Africa/Johannesburg
Africa/Kampala
Africa/Khartoum
Africa/Kigali
Africa/Kinshasa
Africa/Lagos
Africa/Libreville
Africa/Lome
Africa/Luanda
Africa/Lubumbashi
Africa/Lusaka
Africa/Malabo
Africa/Maputo
Africa/Maseru
Africa/Mbabane
Africa/Mogadishu
Africa/Monrovia
Africa/Nairobi
Africa/Ndjamena
Africa/Niamey
Africa/Nouakchott
Africa/Ouagadougou
Africa/Porto-Novo
Africa/Sao_Tome
Africa/Timbuktu
Africa/Tripoli
Africa/Tunis
Africa/Windhoek
AGT
America/Adak
America/Anchorage
America/Anguilla
America/Antigua
America/Araguaina
America/Argentina/Buenos_Aires
America/Argentina/Catamarca
America/Argentina/ComodRivadavia
America/Argentina/Cordoba
America/Argentina/Jujuy
America/Argentina/La_Rioja
America/Argentina/Mendoza
America/Argentina/Rio_Gallegos
America/Argentina/Salta
America/Argentina/San_Juan
America/Argentina/San_Luis
America/Argentina/Tucuman
America/Argentina/Ushuaia
America/Aruba
America/Asuncion
America/Atikokan
America/Atka
America/Bahia
America/Bahia_Banderas
America/Barbados
America/Belem
America/Belize
America/Blanc-Sablon
America/Boa_Vista
America/Bogota
America/Boise
America/Buenos_Aires
America/Cambridge_Bay
America/Campo_Grande
America/Cancun
America/Caracas

America/Catamarca
America/Cayenne
America/Cayman
America/Chicago
America/Chihuahua
America/Coral_Harbour
America/Cordoba
America/Costa_Rica
America/Cuiaba
America/Curacao
America/Danmarkshavn
America/Dawson
America/Dawson_Creek
America/Denver
America/Detroit
America/Dominica
America/Edmonton
America/Eirunepe
America/El_Salvador
America/Ensenada
America/Fort_Wayne
America/Fortaleza
America/Glace_Bay
America/Godthab
America/Goose_Bay
America/Grand_Turk
America/Grenada
America/Guadeloupe
America/Guatemala
America/Guayaquil
America/Guyana
America/Halifax
America/Havana
America/Hermosillo
America/Indiana/Indianapolis
America/Indiana/Knox
America/Indiana/Marengo
America/Indiana/Petersburg
America/Indiana/Tell_City
America/Indiana/Vevay
America/Indiana/Vincennes
America/Indiana/Winamac
America/Indianapolis
America/Inuvik
America/Iqaluit
America/Jamaica
America/Jujuy
America/Juneau
America/Kentucky/Louisville
America/Kentucky/Monticello
America/Knox_IN
America/La_Paz
America/Lima
America/Los_Angeles
America/Louisville
America/Maceio
America/Managua
America/Manaus
America/Marigot
America/Martinique
America/Matamoros
America/Mazatlan
America/Mendoza
America/Menominee
America/Merida
America/Metlakatla
America/Mexico_City
America/Miquelon
America/Moncton
America/Monterrey
America/Montevideo
America/Montreal
America/Montserrat
America/Nassau
America/New_York
America/Nipigon
America/Nome
America/Noronha
America/North_Dakota/Beulah
America/North_Dakota/Center
America/North_Dakota/New_Salem
America/Ojinaga
America/Panama
America/Pangnirtung
America/Paramaribo
America/Phoenix
America/Port_of_Spain
America/Port-au-Prince
America/Porto_Acre
America/Porto_Velho
America/Puerto_Rico
America/Rainy_River
America/Rankin_Inlet
America/Recife
America/Regina
America/Resolute
America/Rio_Branco
America/Rosario
America/Santa_Isabel
America/Santarem
America/Santiago
America/Santo_Domingo

America/Sao_Paulo
America/Scoresbysund
America/Shiprock
America/Sitka
America/St_Barthelemy
America/St_Johns
America/St_Kitts
America/St_Lucia
America/St_Thomas
America/St_Vincent
America/Swift_Current
America/Tegucigalpa
America/Thule
America/Thunder_Bay
America/Tijuana
America/Toronto
America/Tortola
America/Vancouver
America/Virgin
America/Whitehorse
America/Winnipeg
America/Yakutat
America/Yellowknife
Antarctica/Casey
Antarctica/Davis
Antarctica/DumontDUrville
Antarctica/Macquarie
Antarctica/Mawson
Antarctica/McMurdo
Antarctica/Palmer
Antarctica/Rothera
Antarctica/South_Pole
Antarctica/Syowa
Antarctica/Vostok
Arctic/Longyearbyen
ART
Asia/Aden
Asia/Almaty
Asia/Amman
Asia/Anadyr
Asia/Aqtau
Asia/Aqtobe
Asia/Ashgabat
Asia/Ashkhabad
Asia/Baghdad
Asia/Bahrain
Asia/Baku
Asia/Bangkok
Asia/Beirut
Asia/Bishkek
Asia/Brunei
Asia/Calcutta
Asia/Choibalsan
Asia/Chongqing
Asia/Chungking
Asia/Colombo
Asia/Dacca
Asia/Damascus
Asia/Dhaka
Asia/Dili
Asia/Dubai
Asia/Dushanbe
Asia/Gaza
Asia/Harbin
Asia/Ho_Chi_Minh
Asia/Hong_Kong
Asia/Hovd
Asia/Irkutsk
Asia/Istanbul
Asia/Jakarta
Asia/Jayapura
Asia/Jerusalem
Asia/Kabul
Asia/Kamchatka
Asia/Karachi
Asia/Kashgar
Asia/Kathmandu
Asia/Katmandu
Asia/Kolkata
Asia/Krasnoyarsk
Asia/Kuala_Lumpur
Asia/Kuching
Asia/Kuwait
Asia/Macao
Asia/Macau
Asia/Magadan
Asia/Makassar
Asia/Manila
Asia/Muscat
Asia/Nicosia
Asia/Novokuznetsk
Asia/Novosibirsk
Asia/Omsk
Asia/Oral
Asia/Phnom_Penh
Asia/Pontianak
Asia/Pyongyang
Asia/Qatar
Asia/Qyzylorda
Asia/Rangoon
Asia/Riyadh
Asia/Riyadh87

Asia/Riyadh88
Asia/Riyadh89
Asia/Saigon
Asia/Sakhalin
Asia/Samarkand
Asia/Seoul
Asia/Shanghai
Asia/Singapore
Asia/Taipei
Asia/Tashkent
Asia/Tbilisi
Asia/Tehran
Asia/Tel_Aviv
Asia/Thimbu
Asia/Thimphu
Asia/Tokyo
Asia/Ujung_Pandang
Asia/Ulaanbaatar
Asia/Ulan_Bator
Asia/Urumqi
Asia/Vientiane
Asia/Vladivostok
Asia/Yakutsk
Asia/Yekaterinburg
Asia/Yerevan
AST
Atlantic/Azores
Atlantic/Bermuda
Atlantic/Canary
Atlantic/Cape_Verde
Atlantic/Faeroe
Atlantic/Faroe
Atlantic/Jan_Mayen
Atlantic/Madeira
Atlantic/Reykjavik
Atlantic/South_Georgia
Atlantic/St_Helena
Atlantic/Stanley
Australia/ACT
Australia/Adelaide
Australia/Brisbane
Australia/Broken_Hill
Australia/Canberra
Australia/Currie
Australia/Darwin
Australia/Eucla
Australia/Hobart
Australia/LHI
Australia/Lindeman
Australia/Lord_Howe
Australia/Melbourne
Australia/North
Australia/NSW
Australia/Perth
Australia/Queensland
Australia/South
Australia/Sydney
Australia/Tasmania
Australia/Victoria
Australia/West
Australia/Yancowinna
BET
Brazil/Acre
Brazil/DeNoronha
Brazil/East
Brazil/West
BST
Canada/Atlantic
Canada/Central
Canada/Eastern
Canada/East-Saskatchewan
Canada/Mountain
Canada/Newfoundland
Canada/Pacific
Canada/Saskatchewan
Canada/Yukon
CAT
CET
Chile/Continental
Chile/EasterIsland
CNT
CST
CST6CDT
CTT
Cuba
EAT
ECT
EET
Egypt
Eire
EST
EST5EDT
Etc/GMT
Etc/GMT+0
Etc/GMT+1
Etc/GMT+10
Etc/GMT+11
Etc/GMT+12
Etc/GMT+2
Etc/GMT+3
Etc/GMT+4
Etc/GMT+5

Etc/GMT+6
Etc/GMT+7
Etc/GMT+8
Etc/GMT+9
Etc/GMT0
Etc/GMT-0
Etc/GMT-1
Etc/GMT-10
Etc/GMT-11
Etc/GMT-12
Etc/GMT-13
Etc/GMT-14
Etc/GMT-2
Etc/GMT-3
Etc/GMT-4
Etc/GMT-5
Etc/GMT-6
Etc/GMT-7
Etc/GMT-8
Etc/GMT-9
Etc/Greenwich
Etc/UCT
Etc/Universal
Etc/UTC
Etc/Zulu
Europe/Amsterdam
Europe/Andorra
Europe/Athens
Europe/Belfast
Europe/Belgrade
Europe/Berlin
Europe/Bratislava
Europe/Brussels
Europe/Bucharest
Europe/Budapest
Europe/Chisinau
Europe/Copenhagen
Europe/Dublin
Europe/Gibraltar
Europe/Guernsey
Europe/Helsinki
Europe/Isle_of_Man
Europe/Istanbul
Europe/Jersey
Europe/Kaliningrad
Europe/Kiev
Europe/Lisbon
Europe/Ljubljana
Europe/London
Europe/Luxembourg
Europe/Madrid
Europe/Malta
Europe/Mariehamn
Europe/Minsk
Europe/Monaco
Europe/Moscow
Europe/Nicosia
Europe/Oslo
Europe/Paris
Europe/Podgorica
Europe/Prague
Europe/Riga
Europe/Rome
Europe/Samara
Europe/San_Marino
Europe/Sarajevo
Europe/Simferopol
Europe/Skopje
Europe/Sofia
Europe/Stockholm
Europe/Tallinn
Europe/Tirane
Europe/Tiraspol
Europe/Uzhgorod
Europe/Vaduz
Europe/Vatican
Europe/Vienna
Europe/Vilnius
Europe/Volgograd
Europe/Warsaw
Europe/Zagreb
Europe/Zaporozhye
Europe/Zurich
GB
GB-Eire
GMT
GMT0
Greenwich
Hongkong
HST
Iceland
IET
Indian/Antananarivo
Indian/Chagos
Indian/Christmas
Indian/Cocos
Indian/Comoro
Indian/Kerguelen
Indian/Mahe
Indian/Maldives
Indian/Mauritius
Indian/Mayotte

Indian/Reunion
Iran
Israel
IST
Jamaica
Japan
JST
Kwajalein
Libya
MET
Mexico/BajaNorte
Mexico/BajaSur
Mexico/General
Mideast/Riyadh87
Mideast/Riyadh88
Mideast/Riyadh89
MIT
MST
MST7MDT
Navajo
NET
NST
NZ
NZ-CHAT
Pacific/Apia
Pacific/Auckland
Pacific/Chatham
Pacific/Chuuk
Pacific/Easter
Pacific/Efate
Pacific/Enderbury
Pacific/Fakaofo
Pacific/Fiji
Pacific/Funafuti
Pacific/Galapagos
Pacific/Gambier
Pacific/Guadalcanal
Pacific/Guam
Pacific/Honolulu
Pacific/Johnston
Pacific/Kiritimati
Pacific/Kosrae
Pacific/Kwajalein
Pacific/Majuro
Pacific/Marquesas
Pacific/Midway
Pacific/Nauru
Pacific/Niue
Pacific/Norfolk
Pacific/Noumea
Pacific/Pago_Pago
Pacific/Palau
Pacific/Pitcairn
Pacific/Pohnpei
Pacific/Ponape
Pacific/Port_Moresby
Pacific/Rarotonga
Pacific/Saipan
Pacific/Samoa
Pacific/Tahiti
Pacific/Tarawa
Pacific/Tongatapu
Pacific/Truk
Pacific/Wake
Pacific/Wallis
Pacific/Yap
PLT
PNT
Poland
Portugal
PRC
PRT
PST
PST8PDT
ROK
Singapore
SST
SystemV/AST4
SystemV/AST4ADT
SystemV/CST6
SystemV/CST6CDT
SystemV/EST5
SystemV/EST5EDT
SystemV/HST10
SystemV/MST7
SystemV/MST7MDT
SystemV/PST8
SystemV/PST8PDT
SystemV/YST9
SystemV/YST9YDT
Turkey
UCT
Universal
US/Alaska
US/Aleutian
US/Arizona
US/Central
US/Eastern
US/East-Indiana
US/Hawaii
US/Indiana-Starke
US/Michigan

US/Mountain
US/Pacific
US/Pacific-New
US/Samoa
UTC
VST
WET
W-SU
Zulu