UniLock
System 10

Manual
UniLock WebAPI

Projekt	PCS125-20
Version	1.0
Revision	240703

WebAPI provides a two-way secure web connection for external systems for UniLock access control.

External systems, such as SCADA, PSIM, monitoring systems, cloud systems, smartphone apps, etc., can perform operational monitoring, control and administration of UniLock access control.

WebAPI uses SignalR for communication, which provides real-time two-way functionality via a Web connection (WebSocket) between clients and server. In addition to exchanging data, this also provides the ability to monitor the connection itself.

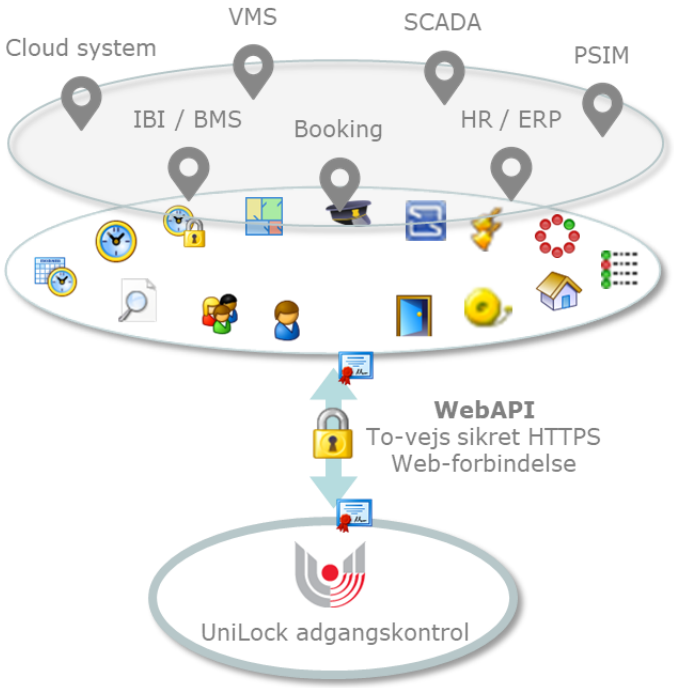
The standard JSON data format is used.

To help developers get started communicating with UniLock WebAPI, the SDK comes with a fully functional client program including .NET source code and Visual Studio project. In addition, the development tool Swagger can be activated in the development phase.

Table of Contents

1. Description

1.1 General description



Application

WebAPI provides a two-way secure web connection for external systems for UniLock access control.

External systems, such as SCADA, PSIM, monitoring systems, cloud systems, smartphone apps, etc., can perform operational monitoring, control and administration of UniLock access control.

Operational monitoring is performed by subscribing to current status of relevant doors, DAS, intrusion alarms, entrances and exits, logging, areas, alarms, etc.

Control allows remote control of doors and DAS (intrusion alarms).

Administration provides the ability to create / retrieve / update / delete people, groups of people, k-points, DAS, time controls, etc.

Use your own system on top of the UniLock access control.

Description

WebAPI uses SignalR for communication, which provides real-time two-way functionality via a Web connection (WebSocket) between clients and server. In addition to exchanging data, this also provides the ability to monitor the connection itself.

The standard JSON data format is used.

To help developers get started communicating with UniLock WebAPI, the SDK comes with a fully functional client program including .NET source code and Visual Studio project. In addition, the development tool Swagger can be activated in the development phase.

A description of the UniLock access control functionality in the PC program can be found in the manual for the PC program.

1.2 Limitations

WebAPI can be used in full when a right has been acquired for the WebAPI module, and in demo mode WebAPI can be used for a limited time in relation to the number of times it is activated.

A number of object types can already be controlled through WebAPI, while more object types are continuously added.

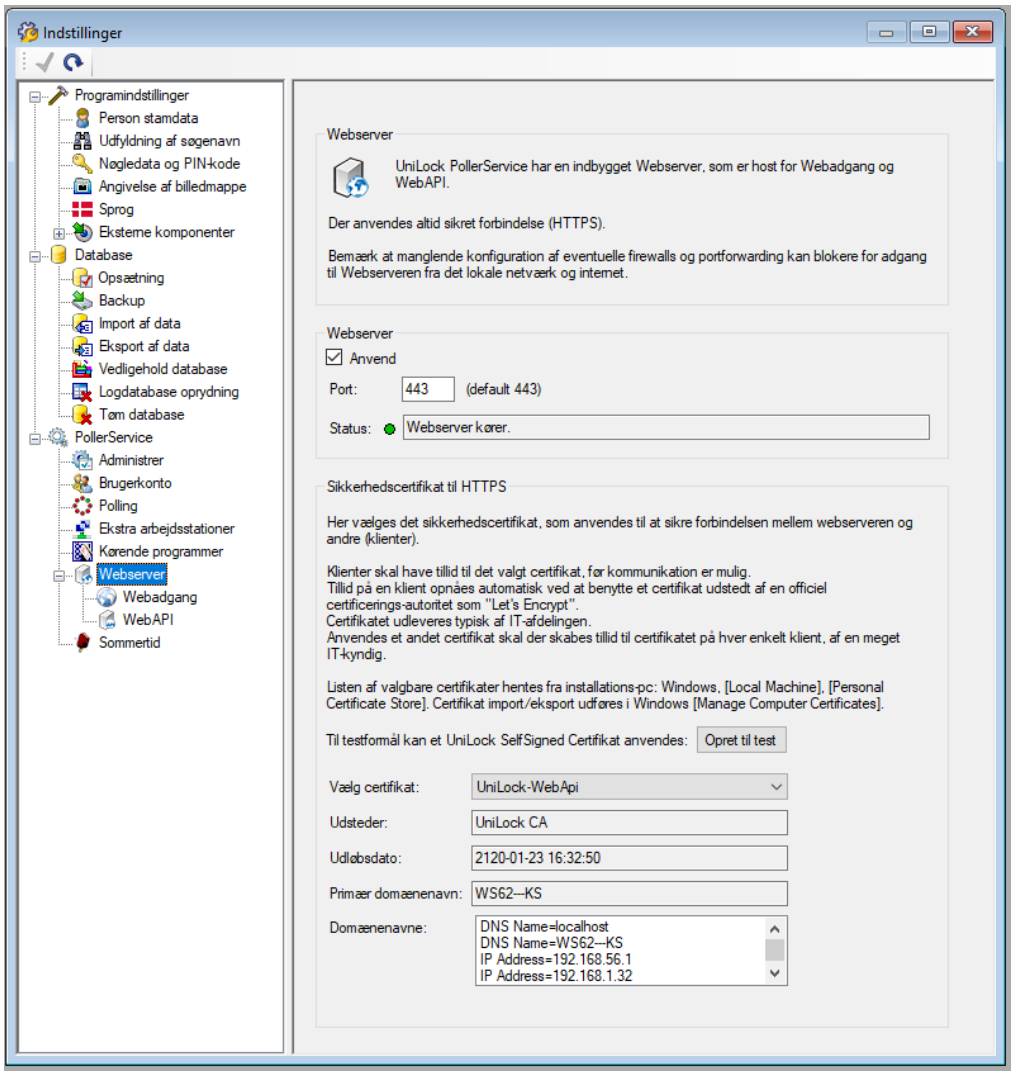
WebAPI returns information about missing rights when rights restrictions are experienced.

2. Setup

WebAPI is hosted in the built-in Web server of UniLock PollerService. The web server must thus be activated before WebAPI can be activated.

2.1 Webserver

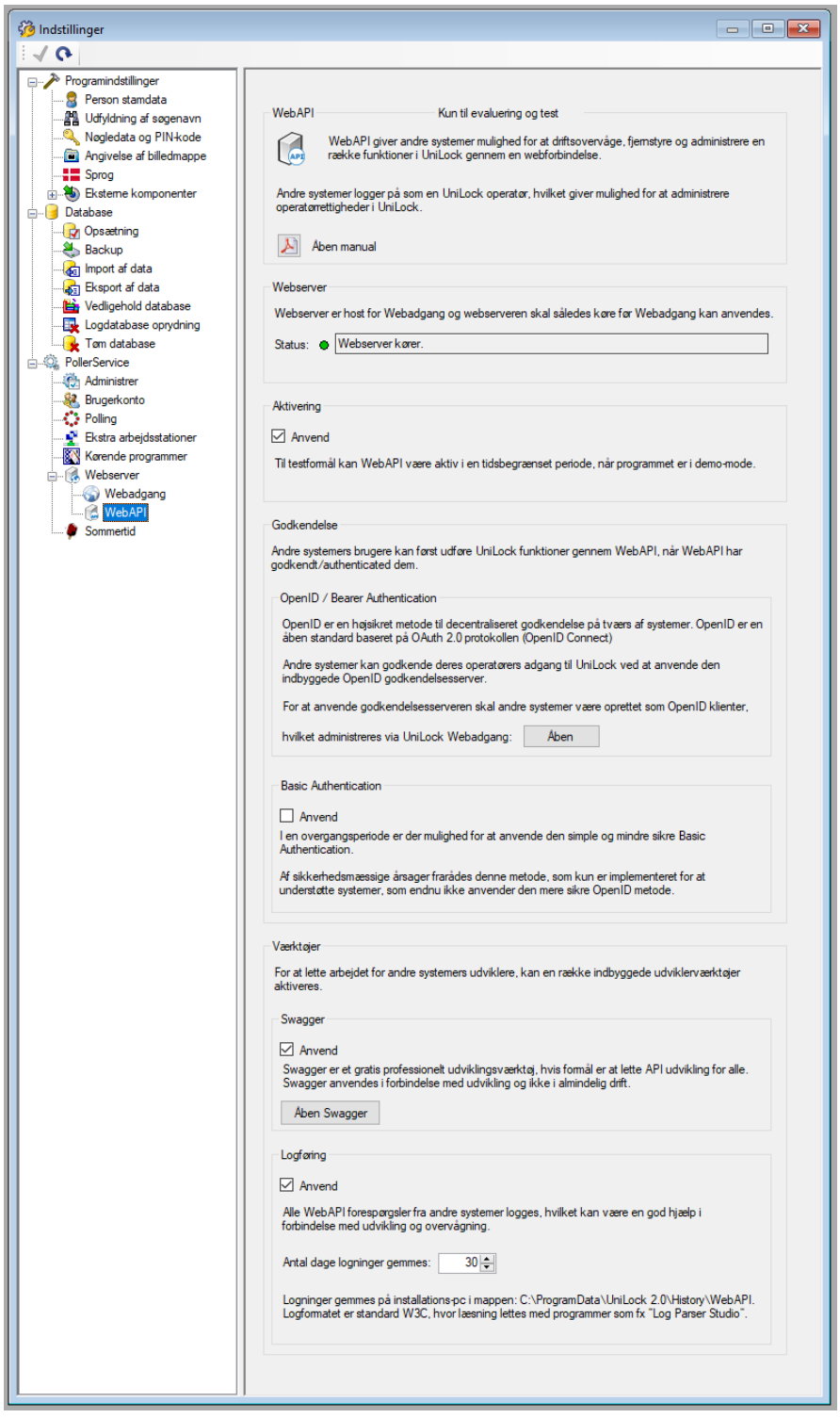
Web server setup is done in UniLock Adgangsditor: [Settings], [PollerService], [Webserver].



UniLock Web Server uses encrypted connection (HTTPS) for the communication connection. For encryption, the operator should select its own certificate or alternatively generate a self-signed UniLock certificate for evaluation and testing. Certificate for encryption can be selected based on the certificates that have the private key available and located on the installation PC in [Windows Certificate Store], [Local Machine], [Personal]. UniLock automatically generates alarms in case of certificate error and upcoming expiration of certificate.

2.2 WebAPI

WebAPI setup is done in UniLock Adgangseditor: [Indstillinger], [PollerService], [Webserver], [WebAPI].



As a rule, logging of requests to WebAPI is switched on with a duration of 30 days. The log files are formatted according to the standard W3C log, which makes it possible to read and analyze the log files with free programs such as: ”*Log Parser Studio”*.

Logs are placed on the installation PC: ”C:\ProgramData\UniLock 2.0\History\WebAPI\”.

3. Get started

To help developers of UniLock WebAPI clients get started, the following help tools are included:

- Client program for demonstration of functionality (WebAPITestClient.exe).
- Project files and source code for client program written in .NET in Visual Studio.
- Built-in Swagger support, which can be activated / deactivated.
- Schemes for JSON auto-generation of .NET classes, TypeScript, etc. Look e.g. https://app.quicktype.io/#l=schema and https://www.jsonschemavalidator.net/.

The utility tools are located on the installation PC in the installation folder, where the default location is: ”C:\Program Files (x86)\UniLock 2.0\Documentation\WebAPI\ClientSourceCode.zip”

4. API-Datacommunication

In this section, an overall description of data communication is included UniLock WebAPI.

4.1 Data request

In data queries, clients can specify which category, which type of object and any specific object for which data is desired. The address path has this general structure: ”https://[IP-adresse]:[Port]/api/[version]/[category]/[objecttype]/”

Possibilities for [category]:

- setup: Settings for the object.
- operation: Latest status of the object, such as whether the door is open.
- logging: logs and alarms.

Possibilities for [objecttype]:

- ControlPoint
- DASGroup
- Gateway
- GatewayConnector
- TimeTable
- SecurityLevelTimeTable
- SharedException
- Person
- PersonGroup
- Operator
- OperatorGroup
- Log
- Area

For example, clients can send a query about the operating status of the C-point ID=752e9150-b9ac-4801-b197-59a35207e552 by using: ”https://[IP-adresse]:[Port]/api/v1/operation/controlpoint/752e9150-b9ac-4801-b197-59a35207e552”

4.2 Data formats

UniLock WebAPI can return data in JSON format.

4.3 Date and Time

ISO 8601 is the primare standard used to format Date and Time. This allows for specifying time as UTC, UTC with an offset or local time. Local time is Windows time on the installation PC:

- YYYY-MM-DDTHH:mm:ss (Local time)
- YYYY-MM-DDTHH:mm:ssZ (Z = UTC)
- YYYY-MM-DDTHH:mm:ss+HH:mm (UTC+HH:mm)
- YYYY-MM-DDTHH:mm:ss-HH:mm (UTC-HH:mm)

In places where time is stated without a date, the time is used as local time, when nothing else stated.

4.3.1 Special regarding validityperiods

Since Build 186 ValidityPeriodsDto “Start” and “End” are nullable DateTime. Start=null represents start time infinite back in time and End=null represents infinite forward in time.

4.4 Paging

Clients can query a selection of all objects, as WebAPI supports paging of object collections. This is implemented as selectable Query parameters [skip] and [take]: ”https://[IP-adresse]:[Port]/api/v1/setup/controlpoint?skip=1&take=4”.

[skip] indicates how many of the first objects in the list are to be skipped (default = 0).

[take] indicates how many objects are desired to be returned (default = 100).

Examples:

Objects	=	[1,2,3,4,5,6,7,8,9]
Objects: skip (2)	→	[3,4,5,6,7,8,9]
Objects: take (2)	→	[1,2]
Objects: skip (3), take (2)	→	[4,5]

4.5 Amount of data

To minimize the amount of data, the default response to a client will only contain a summary of objects in the system. Full data representation for objects can be received by the client by adding the Query parameter”includeFullObject=true” or by a request for a specific object.

When full data representation is used, each object type will have a maximum number of objects per query, and paging is used if the client needs more objects. Header in response to queries will contain "TotalRecordCount", which indicates the total number of objects in the system.

JSON summary object:

```
{
    "Guid": "aac30dae-deaf-4df9-b840-97682dde6efc",
    "Name": "Kontor"
}
```

4.6 Error messages

If an error occurs in the server or the request to the server is invalid, the response will contain a descriptive error message.

An example of an error message is the JSON error summary for creating an object whose search name is already in use:

```
{
    "ErrorType": "ValidationError",
    "ErrorCode": 2,
    "Message": "Validation failed",
    "Details": [
        "ErrorType": "NameAlreadyInUse",
```

```
        "ErrorCode": 12,
        "ErrorMessages": [
            "[Name] has to be unique"
        ]
    ]
}
```

4.7 Language

The API supports language variants for error messages, logs, etc

For errors messages the [Details.ErrorMessage] string is translated. For logs all [Description] sub fields are translated.

The default language is English, but can be changed by adding a [Language] header to API queries. The Same header is used when subscribing to events from SignalR Hubs.

IEFT language tags for Danish (da-DK) and English (en-GB) are supported.

5. Two-way communication via Web

In this section, a general description of how clients can establish a two-way communication connection with UniLock via the Web, which is monitorable.

Two-way communication connection makes i.a. that the client can monitor whether the connection to UniLock is intact and not lost. If the connection is lost, the client can thus react to this by, for example, advising its own operators about this.

For two-way communication connection, UniLock uses SignalR, which is a Microsoft ASP.NET software library that can be used to create real-time functionality via a Web connection (WebSocket) between clients and server. SignalR allows UniLock to send data to clients without having to constantly query data.

When requesting UniLock WebAPI, the path used is: ”https://[IP-adresse]:[Port]/”.

5.1 SignalR Hubs

SignalR Hubs are connection points that enable the client and server to call methods from each other. Hubs also make it possible to call methods with strong type parameters and enable model binding.

Below is the information about the different hubs and what you can do with them.

5.1.1 SetupHub

With SetupHub, clients can use SignalR to connect to UniLock and thereby be notified every time object settings are changed (notifications). A notification is an object with ID, object type and event type.

Eventtype can be:

1. Created
2. Changed
3. Deleted

Connection information:

- Name: SetupHub
- Required operator rights in UniLock: [WebAPI]
- Event: ObjectUpdated, ObjectDepartmentSharingRemoved

Subscribing:

Clients subscribe using its *SignalR.HubProxy* by calling *invoke*, with one of the method names below as an argument.

Eg: *HubProxy.invoke(”SubscribeChangesControlPoints”).*

Object type:	**Subscribe**
C-point	”SubscribeChangesControlPoints”
DAS Group	”SubscribeChangesDASGroups”
Gateway	”SubscribeChangesGateways”

Time table	”SubscribeChangesTimeTables”
Security level time table	”SubscribeChangesSecurityLevelTimeTables”
Shared exception	”SubscribeChangesSharedExceptions”
Persons	”SubscribeChangesPersons”
Person groups	”SubscribeChangesPersonGroups”
Operator	”SubscribeChangesOperators”
Areas	”SubscribeChangesAreas”
OperatorGroups	”SubscribeChangesOperatorGroups”
SpecialDayCalenders	”SubscribeChangesSpecialDayCalenders”
Departments	”SubscribeChangesDepartments”

Objekttype	**Unsubscribe**
C-point	”UnsubscribeChangesControlPoints”
DAS Group	”UnsubscribeChangesDASGroups”
Gateway	”UnsubscribeChangesGateways”
Time table	”UnsubscribeChangesTimeTables”
Security level time table	”UnsubscribeChangesSecurityLevelTimeTables”
Shared exception	”UnsubscribeChangesSharedExceptions”
Persons	”UnsubscribeChangesPersons”
Person groups	”UnsubscribeChangesPersonGroups”
Operator	”UnsubscribeChangesOperators”
Areas	”UnsubscribeChangesAreas”
OperatorGroups	”UnsubscribeChangesOperatorGroups”
SpecialDayCalenders	”UnsubscribeChangesSpecialDayCalenders”
Departments	”UnsubscribeChangesDepartments”

Register events:

UniLock can send notifications to the client once the client has configured its event handler:

Eg:

Dynamic typed result:

HubProxy.On(”ObjectUpdated”, (data) => {});

Strongly typed result:

HubProxy.On<SetupEvent> (”ObjectUpdated”, (SetupEvent data) => {});

5.1.2 OperationHub

With OperationHub, clients can use a SignalR connection to connect to UniLock and thereby subscribe to the current status of the object types c-points and gateways. Each time UniLock retrieves new information from hardware (c-points and gateways), OperationHub will send notifications (status object) to all clients subscribing to that ID.

Connection information:

- Name: OperationHub
- Required operator rights in UniLock: [WebAPI]
- Event: ObjectStatusUpdated, ControlPointStatusUpdated, GatewayStatusUpdated, GatewayConnectorStatusUpdated

Subscribing:

Clients subscribe using its *SignalR.HubProxy* by calling *invoke*, with one of the method names below as an argument.

Eg:

string id = "05c11e2e-94a6-43b0-b8b0-e312de7402b9";

HubProxy.invoke("SubscribeChangesControlPoint", id);

Object type:	**Quantity**	**Subscribe**
C-point	Single	”SubscribeChangesControlPoint” Parameter: String | GUID
	Multiple	”SubscribeChangesControlPoints” Parameter: String [] | GUID []
Gateway	Single	”SubscribeChangesGateway” Parameter: String | GUID
	Multiple	”SubscribeChangesGateways” Parameter: String [] | GUID []
Gateway Connection	Single	”SubscribeChangesGatewayConnector” Parameter: String | GUID
	Multiple	”SubscribeChangesGatewayConnectors” Parameter: String [] | GUID []

Object type:	**Quantity**	**Unsubscribe**
All	Single	”UnsubscribeChange” Parameter: String | GUID
	Multiple	”UnsubscribeChanges” Parameter: String [] | GUID []
	All	“Unsubscribe”

Register events:

GUID sent as String or as System.GUID data type.

UniLock can send object status to the client once the client has configured its event handler:

Eg:

Dynamic typed result:

HubProxy.On(”ObjectStatusUpdated”, (data) => {});

Strongly typed result:

HubProxy.On<OperationControlPoint> (”ControlPointStatusUpdated”, (OperationControlPoint data) => {});

5.1.3 AreaOperationHub

With AreaOperationHub, clients can use a SignalR connection to connect to UniLock and thereby subscribe to areas.
Each time a change occurs in an area, 2 types of events can be sent:

OperationAreaChanged:
If an area has been created, updated or deleted. Then AreaOperationHub sends an event to clients who subscribe to that ID, or who subscribe to all changes for areas.
The event contains information about the type of change, which area it relates to and what the area's current values are as the current number of people in the area and the last event that has taken place in the area..

OperationAreaPersonChanged:
If there has been a change in a person, or the person's relationship to an area. Then AreaOperationHub sends an event to all connected clients.
The event contains information about the type of change, which person it concerns and the person's status in all the areas in which the person is present..

Connection information:

- Name: AreaOperationHub
- Required operator rights in UniLock: [WebAPI] and [Areas]
- Event: AreaChanged, AreaPersonChanged

Subscribing:

Clients subscribe using its *SignalR.HubProxy* by calling *invoke*, with one of the method names below as an argument and the object [Guid] as parameter.

Eg:

string id = “05c11e2e-94a6-43b0-b8b0-e312de7402b9”;

HubProxy.invoke(“SubscribeArea”, id);

Type:	**Quantity**	**Subscribe**
Area	Single	”SubscribeArea”

		Parameter: String | GUID
	Multiple	”SubscribeAreas” Parameter: String [] | GUID []
	All	”SubscribeAllAreas”

Type:	**Quantity**	Unsubscribe
Area	Single	”UnsubscribeArea” Parameter: String | GUID
	Multiple	”UnsubscribeAreas” Parameter: String [] | GUID []
	All	”Unsubscribe”

Register events:

UniLock can send logs to the client once the client has configured its event handler:

Eg:

Dynamic typed result:

HubProxy.On(”AreaChanged”, (area) => {});

Strongly typed result:

HubProxy.On<OperationAreaChanged> (”AreaChanged”, (OperationAreaChanged area) => {});

5.1.4 LogHub

With LogHub, clients can use SignalR to connect to UniLock and thereby get all new logins that are made in the system, and get to know when alarms are set..

LogReceived:
Each time a logging is created which is sent out to the Adgangseditor, a "LogReceived" event is sent via LogHub to the clients who subscribe to the logging, either directly on the log type, or on the logging category.

The event contains information about the logging, and uses the data type "Log".

AlarmsCleared:
When one or more alarm logs are canceled in UniLock, an event will be sent out which contains a list of the ID logings Id that have been canceled.

Connection information:

- Name: LogHub
- Required operator rights in UniLock: [WebAPI] and [Poller]
- Event: LogReceived, AlarmsCleared

Subscribing:

When subscribing to loggings, the strict ones that you can send with as parameters are those that correspond to the logging [Type].

Type:	**Quantity**	**Subscribe**
Log	Single	”SubscribeLog” Parameter: String
	Multiple	”SubscribeLogs” Parameter: String []
	All	“SubscribeLogs”
Alarms	All	“SubscribeAlarms”
Cleared alarms	All	“SubscribeAlarmsCleared”
Info	All	“SubscribeInfoLogs”
Operators	All	”SubscribeOperatorLogs”
System	All	”SubscribeSystemLogs”
C-Points	All	”SubscribeControlPointLogs”

Type:	**Quantity**	**Unsubscribe**
Log	Single	”UnsubscribeLog” Parameter: String
	Multiple	”UnsubscribeLogs” Parameter: String []
	All	“Unsubscribe”
Alarms	All	“UnsubscribeAlarms”
Cleared alarms	All	“UnsubscribeAlarmsCleared”
Info	All	“UnsubscribeInfoLogs”
Operators	All	”UnsubscribeOperatorLogs”
System	All	”UnsubscribeSystemLogs”
C-Points	All	”UnsubscribeControlPointLogs”

Register events:

UniLock can send logs to the client once the client has configured its event handler:

Eg:

Dynamic typed result:

HubProxy.On(”LogReceived”, (data) => {});

Strongly typed result:

HubProxy.On<Log> (”LogReceived”, (Log data) => {});

5.2 Authentication

Clients must log in to retrieve data from UniLock WebAPI. There are three ways to log in to WebAPI:

1. Authentication header:

   The first option is to add an authentication header to its SignalR connection using the same procedure as described in section 6 Authentication.

2. Cookie:

   The second option is to add a cookie to its SignalR connection, where the cookie's name: "Authorization" and value must be with the same procedure as described in section 6 Authentication.

3. Query:
   The third option is to place your login in query string, with key: "Authorization", where value must be with the same procedure as described in section 6 Authentication.

The client's login must be created as an operator in UniLock and as a minimum have operator rights to make API calls [WebAPI]. The operator used must also have operator rights to the object types that can be used via WebAPI.

5.3 Authorization

Clients can receive notifications when:

1. The client is authenticated as described in section 5.2 Authentication.
2. The UniLock operator login used has a privilege level of "limited privilege" or higher for the desired objects.

If the client eg wants:

- Listen to notifications regarding C-points, then operator rights with a minimum right level are required to read / see c-points etc..
- Use logs, then operator rights for Log Search and Poller are required.

6. Authentication

To be able to authenticate against WebAPI, it requires that the operator you use, have a password and have the necessary rights in UniLock.

6.1 Basic authentication

UniLock WebAPI uses Basic authentication, for example by default HTTP requests.

To make valid request for WebAPI, HTTP request must contain [Authorization] header, with contentt *”Basic {Login information}”*.

The login information must be a text string with the format *”{username}:{password}”* be Base64Encoded with the character set ”UTF-8”.

Eg:

Username = ”test user”

Password = 123456

Header = name: ”Authorization”, value: ”Basic dGVzdCB1c2VyOjEyMzQ1Ng==”

6.2 Bearer authentication: OpenID Connect

UniLock WebAPI uses a certificeret OpenID Connect provider library (IdentityServer3[1]), which allows UniLock to act as an identity provider for integrating clients.

To authenticate to WebAPI, there are a number of certified client libraries at OpenID Connect[2].

The Web API authority server is located at the address “https://[IP-adresse]:[Port]/auth/v1/”.
The address is used in OpenID clients, who then find the necessary information themselves.

6.2.1 Supported authentication flows

WebAPI currently supports the following OpenID authentication flow[3]

- Implicit
- Authorization Code
- Hybrid

In addition, is also supported PKCE (rfc7636) flow ”Authorization Code” and ”Hybrid”, which is configured when registering the client.

You can only register one flow per client, due to the limitations of the used library.

6.2.2 Client registration

Use of OpenID requires clients to be authorized / trusted, which is achieved by:

[1] https://github.com/IdentityServer/IdentityServer3
[2] https://openid.net/developers/certified/
[3] https://openid.net/specs/openid-connect-core-1_0.html#Authentication

1. In UniLock Web Access, an administrator operator logs in and opens [Menu], [Settings], [Create New Client] which triggers a registration token. The token only fits the UniLock installation in question and has a lifespan of 2 hours.
2. Clients register with the generated token.
3. Clients will in future use the returned client_id and client_secret in communication with WebAPI, and use the returned registration_access_token for later configuration of the client.

Client registration follows:
- OpenID Connect Dynamic Client Registration 1.0
- OAuth 2.0 Dynamic Client Registration Protocol (rfc7591)
- OAuth 2.0 Dynamic Client Registration Management Protocol (rfc7592)

The registration takes place at the address “https://[IP-adresse]:[Port]/auth/v1/connect/register”, but can also be found through the discovery document as the field “registration_endpoint”.

6.2.2.1 Client informations

UniLock's representation of a client can be found in the json forms. But in general, it follows much of it from the openID document, with a few additions.

Special fields:
In addition to some of the standard fields from openID, the client information objects also contain some additional parameters:
- ”enabled”:
  Describes whether the client is turned on and can be used actively in the system or not. Clients cannot turn it on or off themselves, only UniLock itself can.
- ”require_proof_key”:
  Used to determine if the PKCE variant of the authentication flow is to be used (except implicitly).

Mapping to flow:
To select which authentication flow you would like to use with your client, make sure that the client's data contains so that it matches the following scheme.

grant_types:		**response_types:**			**Authentication flow:**
implicit:	**authentication_code:**	**code:**	**token:**	**id_token:**	
true	false	default			Implicit
false	true	true	false	False	Authorization code
		true	true/false	true/false	Hybrid
		default			Authorization code
default		default			Authorization code

6.2.2.2 Create

To create a client, the client must send a ”POST: …/auth/v1/connect/register”.

Header:
- Authorization: ”Bearer {registration token}”:
  Registration token is the token issued by UniLock Web Acces, for registration of a new client (Remember the limited lifespan of 2 hours).

Minimum information for creating a client is "client_name" and a valid "redirect_uri".

In the response from UniLock, will be the full client information object that UniLock has registered and stored.

6.2.2.3 Get

Clients can retrieve their registered information with”GET: …/auth/v1/connect/register/:id”,
where the ID is the client's own "client_id" (it can’t be used to retrieve other clients' information).

Header:

- Authorization: ”Bearer {client token}”:
  client token is the token “registration_access_token” which was returned along with “client_secret” and “client_id” in response to the creation.

In the response from UniLock, will be the full client information object, except fields like “registration_access_token” and “registration_client_uri”.

6.2.2.4 Update

Clients can update their information with “PUT: .../auth/v1/connect/register/:id”,
where the ID is the client's own “client_id”.

Header:

- Authorization: ”Bearer {client token}”:
  client token is the token “registration_access_token” which was returned along with “client_secret” and “client_id” in response to the creation.

In the response from UniLock, will be the full client information object that UniLock has registered and stored.

Important:
When updating the client's information, a new “registration_access_token” and “client_secret” will be issued which must be used, as the existing one can no longer be used.

6.2.2.5 Delete

Clients can delete their information with “DELETE: .../auth/v1/connect/register/:id” ,
where the ID is the client's own "client_id".

Header:

- Authorization: ”Bearer {client token}”:
  client token is the token “registration_access_token” which was returned along with “client_secret” and “client_id” in response to the creation.

6.2.3 Client administration

To manage clients in UniLock's Web Access, you must be logged in with an operator and have chosen to use the administrator group.

To find the page for the administration, you can open the Settings window via the Adgangseditor, where you then select in the menu [Settings], [PollerService], [Webserver], [WebAPI]. and then press [Open] under the OpenID / Bearer Authentication item.

You can also just open the Web Access and press [Menu]. [Settings] at the top of the page.

When you are in the panel with the list of clients, you have the option here:

- Generate a new registration token, for creating new clients
- Enable / disable clients
- Delete clients
- Assign a new “access_token” to clients if they have lost their token / access.

6.3 Selection of specific operator group

WebAPI supports the ability to select a specific operator group for its http requests.

To select an operator group for your request, you can add a header named "OperatorGroup" and the value being the operator group guid.

It is not required to make a request, but if the operator is a member of several operator groups, it is necessary to guarantee that it always uses the operator group you expect.

6.4 Get current operator

As part of being able to be a member of several operator groups, it may be necessary to be able to see which operator groups the current login has the opportunity to use.

The information is always available by getting the current login's operator with request to”GET: …/setup/operator/current”.

In the operator object, you will find the list of memberships that you can use to select a specific operator group as described above.

7. C-points

Here, the data communication regarding the object type "Controlpoint" (c-point / door) is described.

It is possible to both create / get / update / delete c-points and to receive current status.

7.1 Setup

C-point setup is accessed via: ”https://[IPadresse]:[Port]/api/v1/setup/controlpoint/”.

Overall these headers are used:

- Accept:
  - Application/json
- Content-Type:
  - Application/json
- Authorization:
  - Basic [Base64Encoded login informations]

7.1.1 Create

Clients can create new c-points with ”POST: …/setup/controlpoint/”.

Minimum information for creating a new c-point is Name and IdNumber, where other object information is automatically set to default values.

7.1.2 Get

Clients can query at a specific c-point with”GET: …/setup/controlpoint/:id”.

Clients can query a collection of c-points with”GET: …/setup/controlpoint/”.

Parameters:

- Skip [string]
- Take [string]
- includeFullObject [true/false]

The client will receive a maximum of 50 full (Query parameter "includeFullObject = true") objects at a time, where paging is used in case the system contains several objects (the header of the answer contains "TotalRecordCount", which indicates the total number of objects in the system).

7.1.3 Update

Clients can update a c-point in part with”PATCH: …/setup/controlpoint/:id”.

Patch acts as a partial update, where only enclosed information is updated.

7.1.4 Delete

Clients can delete a c-point with ”DELETE: …/setup/controlpoint/:id”.

7.1.5 Example

Example of minimum amount of information in a POST call to create a new c-point

Body:

JSON:

```
{
    "Name": "Name",
    "Device": {
        "Identification": {
            "Polling": {
                "IdNumber": "4097"
            }
        }
    }
}
```

IdNumber is used as a hexadecimal value and is specified with the decimal value of the desired hexadecimal value, where IdNumber=4097 in WebAPI is converted to the used k-point ID number=1001 in UniLock.

7.2 Status

Status for c-points is online status, current operating parameters, level of inputs and outputs, DAS status, time of last check, etc.

C-point status is accessed via: ”https://[IP-adresse]:[Port]/api/v1/operation/controlpoint/”.

These headers are generally used:

- Accept:
  - Application/json
- Content-Type:
  - Application/json
- Authorization:
  - Basic [Base64Encoded login informations]

Parameters:

- Skip [string]
- Take [string]
- includeFullObject [true/false]

Clients can retrieve the status of a specific c-point with

”GET: …/operation/controlpoint/:id”

Clients can retrieve the status of a collection of c-points with

”GET: …/operation/controlpoint/”

The client will receive a maximum of 50 full (Query parameter "includeFullObject = true") objects at a time, where paging is used in case the system contains several objects (the header of the answer contains "TotalRecordCount", which indicates the total number of objects in the system).

7.3 Commands

Clients can send commands to c-points using the HTTP request method ”PUT”.

7.3.1 Unlock c-point

Clients can unlock the door during the door unlocking time, after which the door is automatically locked again, with the command: ”PUT: …/operation/controlpoint/:id/unlockthenlock”.

Eg:

PUT “https://[IP-adresse]:[Port]/api/v1/operation/controlpoint/05c11e2e-94a6-43b0-b8b0-e312de7402b9/unlockthenlock”

7.3.2 DAS disarm

Clients can disable DAS DAS (Decentralized Alarm Management) at a c-point with the command:

”PUT: …/operation/controlpoint/:id/dasdisarm”.

7.3.3 DAS arm

Clients can connect DAS (Decentralized Alarm Management) in a c-point with the command:

”PUT: …/operation/Controlpoint/:id/dasarm”.

7.3.4 DAS Delayed arming

Clients can delay the normal arm of DAS (Decentralized Alarm Management) to a specified time in a c-point with the command:

”PUT: …/operation/Controlpoint/:id/ dasarmingtimeoverride?timeofday=HH:mm”.

8. Time tabels

Here the data communication regarding the object type ”TimeTable” is described.

It is possible to create / get / update / delete timetables.

8.1 Setup

Timetable setup is accessed via: ”https://[IPadresse]:[Port]/api/v1/setup/timetable/”.

Overall these headers are used:

- Accept:
  - Application/json
- Content-Type:
  - Application/json
- Authorization:
  - Basic [Base64Encoded login information]

8.1.1 Create

Clients can create new timetabels with ”POST: …/setup/timetable/”.

Minimum information for creating a new timetable is Name, where other object information is automatically set to default values.

8.1.2 Get

Clients can request a specific timetable with”GET: …/setup/timetable/:id”.

Clients can request a collection of timetables with ”GET: …/setup/timetable/”.

Parameters:

- Skip [string]
- Take [string]
- includeFullObject [true/false]

The client will receive a maximum of 100 full (Query parameter "includeFullObject = true") objects at a time, where paging is used in case the system contains several objects (the answer header contains "TotalRecordCount", which indicates the total number of objects in the system).

8.1.3 Update

Clients can partially update a timetable with ”PATCH: …/setup/timetable/:id”.

Patch acts as a partial update, where only enclosed information is updated.

8.1.4 Delete

Clients can delete a timetable with ”DELETE: …/setup/timetable/:id”.

9. Security level time table

Here the data communication regarding the object type "SecurityLevelTimeTable" is described.

It is possible to both create / get / update / delete security level time tables.

9.1 Setup

The setup of security level time tables is accessed via:
”https://[IPadresse]:[Port]/api/v1/setup/securityleveltimetable/”.

These headers are generally used:

- Accept:
  - Application/json
- Content-Type:
  - Application/json
- Authorization:
  - Basic [Base64Encoded login information]

9.1.1 Create

Clients can create new security level time tables with ”POST: …/setup/securityleveltimetable/”.

Minimum information for creating a new security level time table is Name, where other object information is automatically set to default values.

9.1.2 Get

Clients can query a specific security level time tables with ”GET:
…/setup/securityleveltimetable/:id”.

Clients can request a collection of security level time tables with ”GET:
…/setup/securityleveltimetable/”.

Parameters:

- Skip [string]
- Take [string]
- includeFullObject [true/false]

The client will receive a maximum of 100 full (Query parameter "includeFullObject = true") objects at a time, where paging is used in case the system contains several objects (the answer header contains "TotalRecordCount", which indicates the total number of objects in the system).

9.1.3 Update

Clients can partially update a security level time tables ”PATCH:
…/setup/securityleveltimetable/:id”.

Patch acts as a partial update, where only enclosed information is updated.

9.1.4 Delete

Clients can delete a security level time table with ”DELETE: …/setup/securityleveltimetable/:id”.

10. Shared Exceptions

Here the data communication regarding the object type "SharedException" is described.

It is possible to both create / get / update / delete shared exceptions.

10.1 Setup

Shared exception setup is accessed via: ”https://[IPadresse]:[Port]/api/v1/setup/sharedexception/”.

Overall these headers are used:

- Accept:
  - Application/json
- Content-Type:
  - Application/json
- Authorization:
  - Basic [Base64Encoded login information]

10.1.1 Create

Clients can create new shared exceptions with ”POST: …/setup/sharedexception/”.

Minimum information for creating a new shared exception is Name, where other object information is automatically set to default values.

10.1.2 Get

Clients can request a specific shared exception with ”GET: …/setup/sharedexception/:id”.

Clients can request a collection of shared exceptions with ”GET: …/setup/sharedexception/”.

Parameters:

- Skip [string]
- Take [string]
- includeFullObject [true/false]

The client will receive a maximum of 100 full (Query parameter "includeFullObject = true") objects at a time, where paging is used in case the system contains several objects (the answer header contains "TotalRecordCount", which indicates the total number of objects in the system).

10.1.3 Update

Clients can partially update a shared exception with ”PATCH: …/setup/sharedexception/:id”.

Patch acts as a partial update, where only enclosed information is updated.

10.1.4 Delete

Clients can delete a shared exception with ”DELETE: …/setup/sharedexception/:id”.

11. Gateways

Here the data communication regarding the object type "Gateway" is described.

It is possible to both create / get / update / delete gateways.

11.1 Setup

Gateway setup is accessed via: ”https://[IPadresse]:[Port]/api/v1/setup/gateway/”.

Overall these headers are used:

- Accept:
  - Application/json
- Content-Type:
  - Application/json
- Authorization:
  - Basic [Base64Encoded login information]

11.1.1 Create

Clients can create new gateways with ”POST: …/setup/gateway/”.

The creation of a new locality requires all the object information filled in in the query.

11.1.2 Get

Clients can inquire at a specific gateway with ”GET: …/setup/gateway/:id”.

Clients can request a collection of gateways with ”GET: …/setup/gateway/”.

Parameters:

- Skip [string]
- Take [string]
- includeFullObject [true/false]

The client will receive a maximum of 100 full (Query parameter "includeFullObject = true") objects at a time, where paging is used in case the system contains several objects (the answer header contains "TotalRecordCount", which indicates the total number of objects in the system).

11.1.3 Update

Clients can partially update a gateway with”PUT: …/setup/gateway/:id”.

All object information must be enclosed.

11.1.4 Delete

Clients can delete a gateway with ”DELETE: …/setup/gateway/:id”.

11.2 Example

Example of minimum amount of information in a POST call to create a new gateway.

Body:

JSON:


```
{
    "Guid": "752e9150-b9ac-4801-b197-59a35207e109",
    "Name": "Lager Vest",
    "GatewayType": "IP",
    "Polling": {
        "Enabled": false,
        "Method": "Continuous",
        "Interval": "00:00",
        "Start": "00:00",
        "End": "00:00",
        "AllDay": true
    },
        "Device": {
            "DeviceType": "CV72V2",
            "COMSetup": null,
            "USBSetup": null,
            "IPSetup": {
                "HostName": "CV72LagerVest",
                "IPPort": 7211,
                "Password": "",
                "EncryptionKey": ""
            },
            "RemoteModem": null
        },
    "Note": "Placeret i teknikrum R215"
}
```


11.3 Status

Status for gateway is online status, current operating parameters, time of last check, etc..

Gateway status is accessed via: ”https://[IP-adresse]:[Port]/api/v1/operation/gateway/”

These headers are generally used:

- Accept:
  - Application/json
- Content-Type:
  - Application/json
- Authorization:
  - Basic [Base64Encoded login oplysninger]

Parameters:

- Skip [string]
- Take [string]
- includeFullObject [true/false]

Clients can retrieve the status of a specific gateway with ”GET: …/operation/gateway/:id”

Clients can retrieve the status of a collection of gateways with ”GET: …/operation/gateway/”

The client will receive a maximum status of 50 (Query parameter "includeFullObject = true") objects at a time, where paging is used in case the system contains more more objects (the header of the answer contains "TotalRecordCount", which indicates the total number of objects in the system).

12. Gateway connecters

Here the data communication regarding the object type "GatewayConnector" is described.

It is possible to receive the current status.

12.1 Status

The status of gateway connecters is online status, current operating parameters, etc.

Gateway connector status is accessed via: ”https://[IP-adresse]:[Port]/api/v1/operation/gatewayconnector/”

These headers are generally used:

- Accept:
  - Application/json
- Content-Type:
  - Application/json
- Authorization:
  - Basic [Base64Encoded login oplysninger]

Parameters:

- Skip [string]
- Take [string]
- includeFullObject [true/false]

Clients can retrieve the status of a specific gateway connector with ”GET: …/operation/gatewayconnector/:id”

Clients can retrieve the status of a collection of gateway connectors with ”GET: …/operation/gatewayconnector/”

The client will receive a maximum status of 50 (Query parameter "includeFullObject = true") objects at a time, where paging is used in case the system contains more more objects (the header of the answer contains "TotalRecordCount", which indicates the total number of objects in the system).

13. Persons

Here the data communication regarding the object type "Person" is described.

It is possible to both create / get / update / delete persons.

13.1 Setup

Person setup is accessed via: ”https://[IPadresse]:[Port]/api/v1/setup/person/”.

Overall these headers are used:

- Accept:
  - Application/json
- Content-Type:
  - Application/json
- Authorization:
  - Basic [Base64Encoded login information]

13.1.1 Create

Clients can create new person with ”POST: …/setup/person/”.

Minimum information for creating a new person is Name, where other object information is automatically set to default values.

13.1.2 Get

Clients can inquire about a specific person with ”GET: …/setup/person/:id”.

Clients can inquire on a collection of people with ”GET: …/setup/person/”.

Parameters:

- Skip [string]
- Take [string]
- includeFullObject [true/false]

The client will receive a maximum of 100 full (Query parameter "includeFullObject = true") objects at a time, where paging is used in case the system contains several objects (the answer header contains "TotalRecordCount", which indicates the total number of objects in the system).

13.1.3 Update

Clients can update a person partially with ”PATCH: …/setup/person /:id”.

Patch acts as a partial update, where only enclosed information is updated.

13.1.4 Delete

Clients can delete a person with ”DELETE: …/setup/person/:id”.

13.1.5 Move

Clients can move a person to another department with
”PATCH: …/setup/person/:id/MoveToDepartment”

Parameters that must be sent as Query parameter:

- departmentId [string]: the id of the department the person is being moved to.
- preserveAccess [true/false]: specifies if access rights should be preserved, after the person has been moved to the other department.
    - If true: a sharing with the department the person is being moved from, is created for the person. Any access rights the person may have, in the current department, is preserved and can be edited.
    - If false: there will not be a sharing created with the department the person is moving from. Any access rights the person may have, in the current department, will be removed.

It’s only department administrators, that has the rights to move a person to another department. Even a system administrator can’t. Further, a person can only be moved if:

1. The person is in the same department as the department administrator.
2. The person is being moved to another department than your own. However, it’s not possible to move a person to the system department.
3. The person is not created/maintained by an integration. If so, the integration must be deleted first.

14. Person groups

Here the data communication regarding the object type "PersonGroup" is described.

It is possible to both create / get / update / delete person groups.

14.1 Setup

Person group setup is accessed via: ”https://[IPadresse]:[Port]/api/v1/setup/persongroup/”.

Overall these headers are used:

- Accept:
  - Application/json
- Content-Type:
  - Application/json
- Authorization:
  - Basic [Base64Encoded login information]

14.1.1 Create

Clients can create new person groups with ”POST: …/setup/persongroup/”.

Minimum information for creating a new person group is Name, where other object information is automatically set to default values.

14.1.2 Get

Clients can inquire about a specific person group with ”GET: …/setup/persongroup/:id”.

Clients can inquire on a collection of person groups with ”GET: …/setup/persongroup/”.

Parameters:

- Skip [string]
- Take [string]
- includeFullObject [true/false]

The client will receive a maximum of 100 full (Query parameter "includeFullObject = true") objects at a time, where paging is used in case the system contains several objects (the answer header contains "TotalRecordCount", which indicates the total number of objects in the system).

14.1.3 Update

Clients can update a person group in part with ”PATCH: …/setup/persongroup/:id”.

Patch acts as a partial update, where only enclosed information is updated.

14.1.4 Delete

Clients can delete a person group with ”DELETE: …/setup/persongroup/:id”.

15. Operators

Here the data communication regarding the object type "Operators" is described.

It is possible to both create / get / update / delete operators.

15.1 Restrictions

Operators are divided into different types, each of which has its own restrictions:

BuiltIn:

Operators of this type are "readonly" and therefore can not be created, changed or deleted. The operator type is used internally to describe, for example, that the "import channel" has created a new person, etc..

Administrator:

Operators of this type can only be partially edited. If the query is made with the Administrator's username and password, then the Administrator's fields can be edited, with the exception of Name.

UniLock: (Default**)**

Operators of this type can be created, modified and deleted. In order to be able to do this, it is required that the operator used for authentication has the necessary rights to be able to change the operator accordingly in the access editor.

ActiveDirectory:

Operators of this type can be deleted and partially changed, but if Active Directory import is still enabled, it will always overwrite the operator's data with data from Active Directory.

The name, account information and operator rights cannot be changed as it is controlled by Active Directory import.

The rest of the fields can be changed if you have the necessary rights to be able to change the operator accordingly in the access editor.

15.2 Setup

The Operator setup is accessed via: ”https://[IPadresse]:[Port]/api/v1/setup/operator/”.

Overall these headers are used:

- Accept:
  - Application/json
- Content-Type:
  - Application/json
- Authorization:
  - Basic [Base64Encoded login informations]

15.2.1 Create

Clients can create new operators with ”POST: …/setup/operator/”.

Necessary information to create a new operator is Name and Username, where other object information is automatically set to default values.

15.2.2 Get

Clients can inquire their current operator with ”GET: …/setup/operator/current”.

Clients can inquire about a specific operator with ”GET: …/setup/operator/:id”.

Clients can inquire about a collection of operators with ”GET: …/setup/operator/”.

Parameters:

- Skip [string]
- Take [string]
- includeFullObject [true/false]

The client will receive a maximum of 100 full (Query parameter "includeFullObject = true") objects at a time, where paging is used in case the system contains several objects (the answer header contains "TotalRecordCount", which indicates the total number of objects in the system)

15.2.3 Update

Clients can partially update an operator with ”PATCH: …/setup/operator/:id”.

Patch acts as a partial update, where only enclosed information is updated.

15.2.4 Delete

Clients can delete an operator with ”DELETE: …/setup/operator/:id”.

16. Operator groups

Here the data communication regarding the object type "OperatorGroup" is described.

It is possible to retrieve operator groups as a list of summary objects.

16.1 Setup

The setup of the operator groups is accessed via:
”https://[IPadresse]:[Port]/api/v1/setup/operatorgroup/”.

Overall these headers are used:

- Accept:
  - Application/json
- Content-Type:
  - Application/json
- Authorization:
  - Basic [Base64Encoded login information]

16.1.1 Get

Clients can inquire about a specific operator group with ”GET: …/setup/operatorgroup/:id”.

Clients can query a collection of operator groups with ”GET: …/setup/operatorgroup/”.

Parameters:

- Skip [string]
- Take [string]

The client will receive a maximum of 100 full (Query parameter "includeFullObject = true") objects at a time, where paging is used in case the system contains several objects (the answer header contains "TotalRecordCount", which indicates the total number of objects in the system)

17. Specialday Calender

Here the data communication regarding the object type "SpecialDayCalender" is described.

It is possible to retrieve operator groups as a list of summary objects.

17.1 Setup

The setup of the specialdaycalender is accessed via:
”https://[IPadresse]:[Port]/api/v1/setup/specialdaycalender/”.

Overall these headers are used:

- Accept:
  - Application/json
- Content-Type:
  - Application/json
- Authorization:
  - Basic [Base64Encoded login information]

17.1.1 Get

Clients can inquire about a specific specialday calender with ”GET:
…/setup/specialdaycalender/:id”.

Clients can query a collection of specialday calenders with ”GET: …/setup/specialdaycalender/”.

Parameters:

- Skip [string]
- Take [string]

18. Departments

Here the data communication regarding the object type "Department" is described.

It is possible to retrieve departments as a list of summary objects.

18.1 Setup

The setup of the departments is accessed via: ”https://[IPadresse]:[Port]/api/v1/setup/department/”.

Overall these headers are used:

- Accept:
  - Application/json
- Content-Type:
  - Application/json
- Authorization:
  - Basic [Base64Encoded login information]

18.1.1 Get

Clients can inquire about a specific department with ”GET: …/setup/department /:id”.

Clients can query a collection of departments with ”GET: …/setup/department/”.

Parameters:

- Skip [string]
- Take [string]

19. Logs

Here, the data communication regarding the object type "Log" is described.

It is possible to get logs.

Logs are accessed via: ”https://[IPadresse]:[Port]/api/v1/logging/log/”.

Overall these headers are used:

- Accept:
  - Application/json
- Content-Type:
  - Application/json
- Authorization:
  - Basic [Base64Encoded login oplysninger]

Important:

- Elements can be added and removed from [LogType] without it being described as a "breaking change", which is why it is recommended that unknown log types be taken care of in integrating clients.
- Logs do not use summary objects and do not return a TotalRecordCount, as the other endpoints do.
- ObjectReferences is only available on logs from the installation date onwards.

19.1 Get

Clients can request a specific log with ”GET: …/logging/log/:id”.

Parameters:

- IncludeDescription [true/false]
  true: Logs must also contain text descriptions

Clients can query a collection of logs with ”GET: …/logging/log/”.

Parameters:

- Skip [int]
- Take [int]
- IncludeDescription [bool]: *true: Logs must also contain text descriptions.*
- FromDate [datetime]: *format: “yyyy-MM-dd”*
- ToDate [datetime]: *format: “yyyy-MM-dd”*
- FromTimeOfDay [datetime]: *format: “HH:mm”*
- ToTimeOfDay [datetime]: *format: “HH:mm”*
- LogType [string]: c*omma separated string.*
- ControlPointName [string]: *May contain wildcard '*' at the beginning or end of the text*
- ControlPoint [Guid]
- PersonName [string]: *May contain wildcard '*' at the beginning or end of the text*
- Person [Guid]
- FromId [long]
- logalarmresetid [long]

- fromlogalarmresetid [long]
- OrderDirection [string]: (asc/desc*) : standard is desc*

20. Alarms

This describes the data communication regarding screen alarms (logs of type Alarm) that use the object type "Log".

It is possible to retrieve and acknowledge alarm logs.

Alarms are accessed via: ”https://[IPadresse]:[Port]/api/v1/logging/alarm/”.

Overall these headers are used:

- Accept:
  - Application/json
- Content-Type:
  - Application/json
- Authorization:
  - Basic [Base64Encoded login oplysninger]

20.1 Get

Clients can query on all the active alarm logs with ”GET: …/logging/alarm/”.

Parameters:

- IncludeDescription [true/false]
  true: Logs must also contain text descriptions

20.2 Acknowledge

Clients can acknowledge active alarm logs with ”POST: …/logging/alarm”.

In the content of the query, the client must send a list of alarm log ID’s.

21. Areas

Here the data communication regarding the object type "Area" is described.

It is possible to retrieve status of areas.

21.1 Status

Status for areas is the number of people in the area, recent incident and expiration times for people in the area, etc.

Area status is accessed via: ”https://[IP-adresse]:[Port]/api/v1/operation/area/”

Overall these headers are used:

- Accept:
  - Application/json
- Content-Type:
  - Application/json
- Authorization:
  - Basic [Base64Encoded login informations]

Parametre:

- Skip [string]
- Take [string]
- includeFullObject [true/false]

Important:
Areas use a specialized summary objects. For more info, refer to the Json Schemes.

Clients can retrieve status for a specific area with ”GET: …/operation/area/:id”

Clients can retrieve status of a collection of areas with ”GET: …/operation/area/”

The client will receive a maximum status of 50 (Query parameter "includeFullObject = true") objects at a time, where paging is used in case the system contains more objects (the answer header contains "TotalRecordCount", which indicates the total number of objects in the system).