**MQ subscription**

Manual for adding API user in Ocean system and subscribing to selected MQs.

Contents

[1 Introduction 3](#_Toc109314241)

[2 Message Queue 4](#_Toc109314242)

[3 Creating API user 5](#_Toc109314243)

[Role management 5](#_Toc109314244)

[Create API user 6](#_Toc109314245)

[Subscription to Message Queue 7](#_Toc109314246)

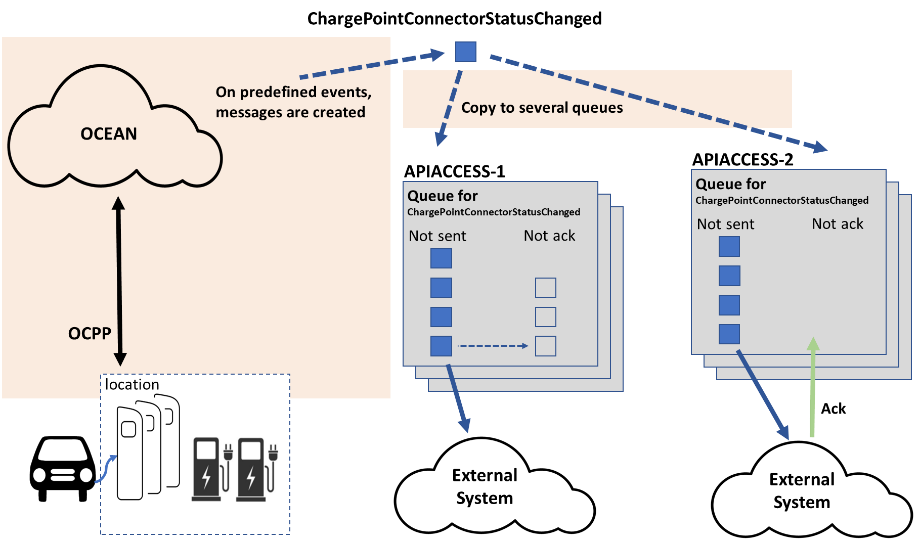
# Introduction

Ocean system integrates numerous charging points on lots of locations. To gather info about current statuses of each asset, user, session or any other data in the system, the external system can take several approaches.

One of the approaches is to request the data using the URCHIN API. Each API call involves a more or less extensive search in the Ocean’s database, which can in a large-scale environment and with frequent requests burden the whole Ocean’s system.

The preferred way of maintaining the UpToDate database is to first build the database with URCHIN API requests. With this baseline, all further changes can be learned from messages of the Ocean’s Message Queue. Those messages are asynchronous and are sent at any important change. The messages are created on the fly during the execution of some logic process, whiteout an additional database search. Therefore, they have only light impact to the system’s performance.

# Message Queue



Each API user can be set to listen to several messages, that Ocean creates and publishes them on to the Message Queue. A new and independent queue is created per each message and per each API user.

When some important event occurs the Ocean system creates a message, for example “*ChargePointConnectorStatusChanged”,* and copies it to all relevant queues. That means, that each API user gets its own copy and is responsible for it. The messages stay in the queue until the API user subscribes to the queue, reads messages and acknowledges them. If the messages are just read and not acknowledged, they stay in the queue and are again sent later upon the re-established subscription. If the API user is not subscribed to the queue at the time of the event, the message is still put into the queue and waits for the API user to subscribe to the queue.

There are some limitations (number and time) when the messages are in any case removed from the queue. API user is responsible to maintain the connection and the subscription to the Message queue to avoid losing messages.

# Creating API user

## Role management

Graphical user interface, application

Description automatically generated

With role management you can specify in detail, what an API user is able to do with URCHIN API and what messages he is able to read from the Message Queue. In the above picture a new role “Connector and Location statuses” was defined with only 2 API URLs enabled (GetConnectors and GetLocations). You can find all API possibilities on the right side and add them to this role one-by-one or bundled in a group.

## Create API user

Graphical user interface, application

Description automatically generated

In Access management press the button “Add api access”.

Graphical user interface, text, application

Description automatically generated

First, add a business partner and a name of the API user. Further, select predefined roles. In the example, we have added the above-described role “Connector and Location statuses” and the role “API MQ consumer”, which enables the subscription to Message Queue.

Next, you should define all relevant messages from Message Queue, that this user will listen to. For each of the defined message a new separate and independent queue will be created in the background.

Graphical user interface, application

Description automatically generated

When saved, the API user will see a newly generated:

* username (APIACCESS-4872) and
* password (Api key - qRBWF2xtOdTlcWrNB5hH67djN8WKNkxU)

Those 2 should be used in the external system when subscribing to the Message Queue.

## Subscription to Message Queue

Graphical user interface, text, application

Description automatically generated

This is a simple application to show the subscription to one of the queues. To do so, you must provide:

* URL of the production server
* Port 5671
* username APIACCESS-4872
* password qRBWF2xtOdTlcWrNB5hH67djN8WKNkxU
* name of the queue
  + APIACCESS-4872.\*.connector.statusChanged
  + APIACCESS-4872.\*.\*.chargepoint.deleted
  + APIACCESS-4872.\*.\*.chargepoint.inserted
  + APIACCESS-4872.\*.\*.chargepoint.updated