

ProvisioningAPI

In this article

- [Handlers](#)
- [Event Payload](#)
 - [Structure](#)
- [Supported Events](#)
 - [Clients Management](#)
 - [Accounts Management](#)
 - [Subscriptions Management](#)
 - [Email Rates Manager](#)
 - [Import Manager](#)
 - [Charges](#)
 - [Payments](#)

The **Provisioning API** provides a mechanism for real-time integration with 3rd party systems, including softswitches, gateways, and CRM systems. The mechanism calls pre-defined handlers on the occurrence of specific events in the system.

While **Core API** and **Management API** sub-systems answer requests, **Provisioning API** pushes data as soon as an event occurs. For example, you can monitor the client's balance status via this functionality. Thus, when the client's balance is below zero, you can configure the system to send a notification to an external system to avoid any disruptions to your organization's current processes.

Handlers

A typical handler should be configured as is **HTTPS** or **HTTP** with an URL of the script, which will get HTTP POST requests about each event occurrence.

*Please avoid using the **Script** type of handlers, they are designed for internal usage within the system.*



Best practice example

There is an example based on **https://hostname/handler-endpoint** usage.

Open the **Provisioning section** and start creating a handler.

1. Specify the name, type, and status.
2. In the **Event** field, select **Clients Create** event from the drop-down list.
3. In the **Task** field, indicate **HTTPS** type and define the URL for the handler, for example, **my-domain.org/api**.
4. Click **Apply**.

Find an example of the **http://handler** below:

```
from flask import Flask, request
import json
app = Flask(__name__)
@app.route("/api", methods=['GET', 'POST'])
def api():
    data = json.loads(request.data)
    return json.dumps(data)
if __name__ == "__main__":
    app.run()
```

Event Payload

Structure

Within notification, the handler will be called with a JSON-formatted data payload. The payload has the following structure:

- **event**
 - **dt** - the date-time of the event in the ISO8601 format;
 - **events_id** - ID of the event occurred (helpful, when multiple events are handled by the same handler);
 - **object_id** - the entity, to which the event happened;
- **data** - data related to the specific event (check Payload Samples below)

Supported Events

Please note that some Payload Samples below contain reduced payload as they fully replicate the data structures within CoreAPI. Please refer to the respective documentation within the system interface in the **Integration/CoreAPI Docs** section for full format of the structure.

Event ID	Payload Sample
Clients Management	
clients.create The client has been created	
clients.update The client's details have been updated	
clients.archive The client has been archived	
clients.delete The client has been completely deleted <i>Not to be confused with archived</i>	
clients.balance_zero The client's balance became <=0	
clients.balance_notzero The client's balance became > 0	

clients.custom_fields.update The client's Custom Field has been updated	
Accounts Management	
clients.accounts.create The account has been created	
clients.accounts.update The account's details have been updated	
clients.accounts.delete The account has been completely deleted	
Subscriptions Management	
clients.subscriptions.assign The subscription has been assigned	

clients.subscriptions.activate The subscription has been activated	
clients.subscriptions.deactivate The subscription has been deactivated	
clients.subscriptions.renew The subscription has been renewed	
clients.subscriptions.close The subscription has been closed	
Email Rates Manager	

email_rates_manager.import

The import file has been prepared for further processing by the task specified in the Provisioning API handler settings, and the import process was stopped

email_rates_manager.rate_tables_no_match

No Rate Tables match received email

email_rates_manager.no_attachments

No attachments have been identified

email_rates_manager.file_processing_error

The file processing error

Import Manager

importd.process_success

The import attempt was successful

importd.process_failed The import attempt has failed	
Charges	

accounting.charges.create

The Transaction of Type Charge has been created

accounting.charges.delete

The Transaction of Type Charge has been removed

accounting.charges.update

The Transaction of Type Charge has been edited

Payments

accounting.payments.create

The Transaction of Type Payment has been created

accounting.payments.delete The Transaction of Type Payment has been removed	
accounting.payments.update The Transaction of Type Payment has been edited	

 Tip

- For more information about **configuring and monitoring the hooks for Provisioning API**, visit our respective article **User Guide > System > [Provisioning API](#)**.
- If you need to process some of the actions that are not listed here, contact our support for a feature request.