Web hooks

WebHooks is a lightweight HTTP pattern providing a simple pub/sub model for wiring together Web APIs and SaaS services. The pattern is simple: when an event happened in a system, a HTTP POST request (the hook/notification) is sent to the registered subscribers. A web hook is an event notification that can be set up (subscribed on) by a client/user so that when an event happen in Visma.net, the client/user is automatically notified. This means that when a resource (for instance Customer) is changed, Visma.net Integrations will invoke a http POST request to the URL defined in the subscription for that event.

The main reason behind web hooks is to provide an alternative to polling. Instead of having the client/receiver polling for changes periodically from the server, web hooks provide a mechanism for getting the changes as it happens.

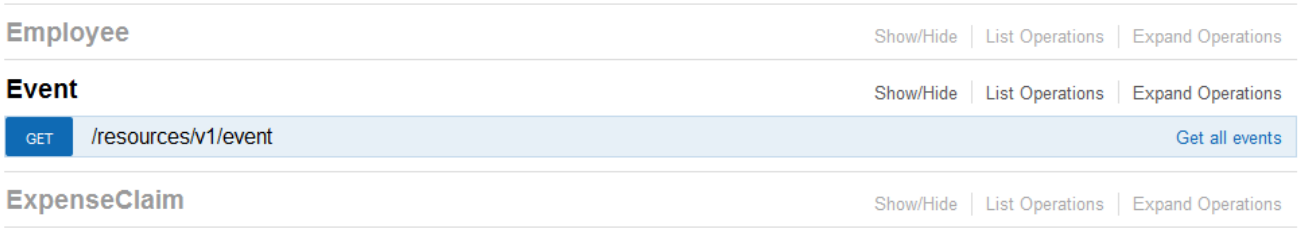
Definitions

Term	Definition
Event	Observable change that are made on the system. For instance, a new Customer was created or an Invoice was approved. The event should be seen as something that happened (past tense)
Subscription	Registered by clients interested in a particular event. Anyone with normal API access can subscribe to events. A subscription will have a URL specifying where the notification should be posted, a type for identifying the event and a set of filters that further identifies the event. For instance, a user (that uses a client) might want to be notified whenever a new Customer was created for a particular company
Notification /WebHook	Action triggered by Visma.net Integrations after the event happened ending up as a http POST request to all the registered subscriptions that match the filtering criteria.

Event discovery

The events are expose by Visma.net Integrations as a REST resource. This resource is public so anyone can get the list of available events using a simple http GET request.

This resource is also exposed in swagger ui:



Sample request

```
GET https://integration.visma.net/API/resources/v1/event HTTP/1.1
Accept: application/json
```

The response is a JSON array containing all the events that handled by Visma.net Integrations:

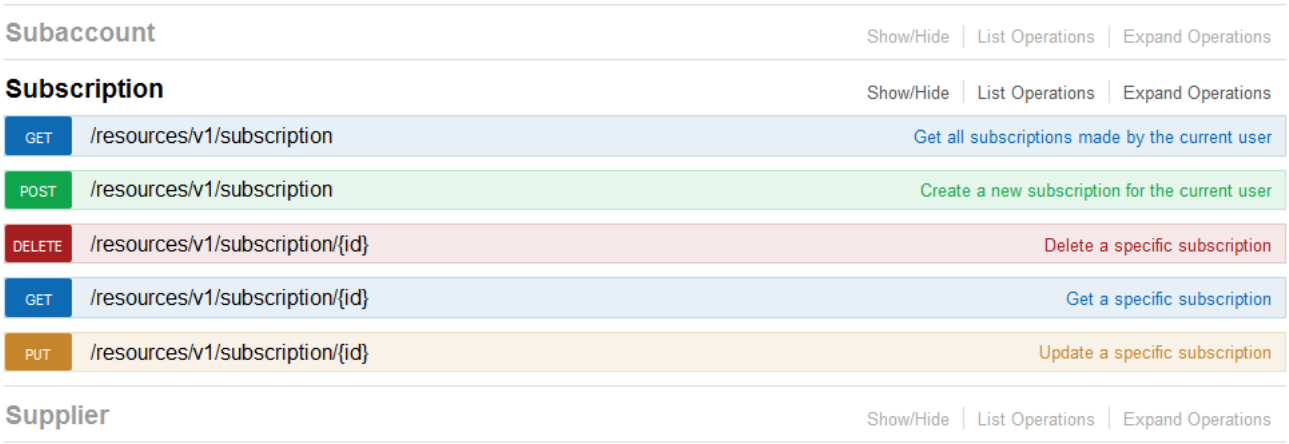
```
[
  {
    "eventType": "customer_changed",
    "name": "Customer changed",
    "description": "Triggered when a new customer was created or an existing one was updated"
  },
  {
    "eventType": "customerinvoice_changed",
    "name": "CustomerInvoice changed",
    "description": "Triggered when a new CustomerInvoice was created or an existing one was updated"
  },
  {
    "eventType": "payment_changed",
    "name": "Payment changed",
    "description": "Triggered when a new Payment was created or an existing one was updated"
  }
]
```


Each new event supported in Visma.net Integrations will be exposed in the response JSON array. An event is unique identified by it's type (eventType). This type will be used when subscribing.

Subscription

In order to register a web hook the client/user needs a subscription on Visma.net Integrations. The client/user can subscribe for each event that he/she is interested of. The subscription is also exposed by Visma.net Integrations as a REST resource. This resource can be programmatically used by clients to manage user's subscriptions. The subscription is tenant (company) based same as the rest of Visma.net exposed resources.

Subscription resource exposed in swagger ui:



1. Create subscription

A subscription is created by invoking an http POST request to https://integration.visma.net/API/resources/v1/subscription

Request:

```
POST https://integration.visma.net/API/resources/v1/subscription HTTP/1.1
Authorization: bearer 1f729814-1a98-4c8e-860b-76ec004742f5
ipp-company-id: 1234
Content-Type: application/json
Accept: application/json

{
  "event": "customer_changed",
  "hookUri": "https://api.example.com/callback"
}
```

When creating a subscription the following details needs to be provided:

- *"event"*, the event that we want to subscribe to. The value must match the "eventType" of one of the exposed events;
- *"hookUri"*, the URL where the client wants to receive the notifications/hooks triggered by Visma.net Integrations when the event happen;

A successful response will look like:

```
HTTP/1.1 201 Created
Location: https://integration.visma.net/API/resources/v1/subscription/18

{
    "id": 18,
    "event": "customer_changed",
    "hookUri": "https://api.example.com/callback"
}
```

2. Get all registered subscriptions for user and company

Request:

```
GET https://integration.visma.net/API/resources/v1/subscription HTTP/1.1
Authorization: bearer 1f729814-1a98-4c8e-860b-76ec004742f5
ipp-company-id: 1234
Accept: application/json
```

Response:

```
[
  {
    "id": 18,
    "event": "customer_changed",
    "hookUri": "https://api.example.com/callback"
  }
]
```

The response is a JSON array containing all the subscriptions made by the user holding the token in the context of the provided company.

3. Get a specific subscription

Request:

```
GET https://integration.visma.net/API/resources/v1/subscription/18 HTTP/1.1
Authorization: bearer 1f729814-1a98-4c8e-860b-76ec004742f5
ipp-company-id: 1234
Accept: application/json
```

Response:

```
{
  "id": 18,
  "event": "customer_changed",
  "hookUri": "https://api.example.com/callback"
}
```

4. Update a specific subscription

The only attribute that can be changed on a subscription is the "hookUri". The other attributes ("id" and "event") are just ignored if provided.

Request:

```
PUT https://integration.visma.net/API/resources/v1/subscription/18 HTTP/1.1
Authorization: bearer 1f729814-1a98-4c8e-860b-76ec004742f5
ipp-company-id: 1234
Content-Type: application/json

{
  "hookUri": "https://api.example.com/callback/1"
}
```

Response:

```
HTTP/1.1 204 No Content
...
```

5. Delete a specific subscription

Request:

```
DELETE https://integration.visma.net/API/resources/v1/subscription/18 HTTP/1.1
Authorization: bearer 1f729814-1a98-4c8e-860b-76ec004742f5
ipp-company-id: 1234
```

Response:

```
HTTP/1.1 204 No Content
...
```

Some details about subscriptions:

- The subscription is tenant (company) based. The company identifier MUST be provided using "ipp-company-id" http header;
- The subscription is "user based". Any user that has a valid Visma.net token and access to the specified company can subscribe to any of the exposed events;
- User's subscription to an event for a company is unique. A user cannot have two subscriptions for the same event and company;
- Multiple users using the same client can subscribe to the same event and company. This means that the client will receive the **same notifications multiple times** and it's the client's responsability to handle these duplicates;

Hooks/Notifications

When an event that matches your subscription happen, Visma.net Integrations will send a notification to the registered *"hookUri"*. The notification is going to be sent as http POST request.

The payload that is sent is a JSON **array**. This is because under load, Visma.net Integrations will group the notifications together and send them only once instead of sending each one individual. For instance if there were 5 *"customerinvoice_changed"* events that happen in a short period of time, Visma.net Integrations will group these and invoke the *"hookUri"* only once instead of making 5 http requests.

Sample request performed by Visma.net Integrations on the registered hookUri:

```
POST https://api.example.com/callback HTTP/1.1
Content-Type: application/json

[
    {
        "contextId":"1234",
        "event":"customerinvoice_changed",
        "action":"CREATED",
        "resourceUri": "https://integration.visma.net/API/controller/api/v1/customerinvoice/100"
    },
    {
        "contextId":"1234",
        "event":"customerinvoice_changed",
        "action":"UPDATED",
        "resourceId": "90"
    }
]
```

In this sample, the JSON array payload contains two notifications for *"customerinvoice_changed"* event that happen on company "1234". The first one is specifying that a CustomerInvoice was created and the second one that another CostomerInvoice was updated.

Notification details:

- *"contextId"*, the tenant/company id;
- *"event"*, the event that triggered the notification;
- *"action"*, the action that was made on the resource. Currently we have only 2 possible actions:
    - *"CREATED"*, signals that a new resource was created. This is always followed by "resourceUri";
    - *"UPDATED"*, signals that a resource was updated. This is always followed by "resourceId";
- *"resourceUri"*, the URL that can be used to GET the resource. This is similar to "Location" header returned on POST requests;
- *"resourceId"*, the internal identifier of the resource that can be used to GET the resource;

How to act on these notifications is each client choice. One may choose to act as it gets them and fetch the changed resource(s) from Visma.net. Other might choose to present these notifications to the users and let the user decide what data to fetch.

Security

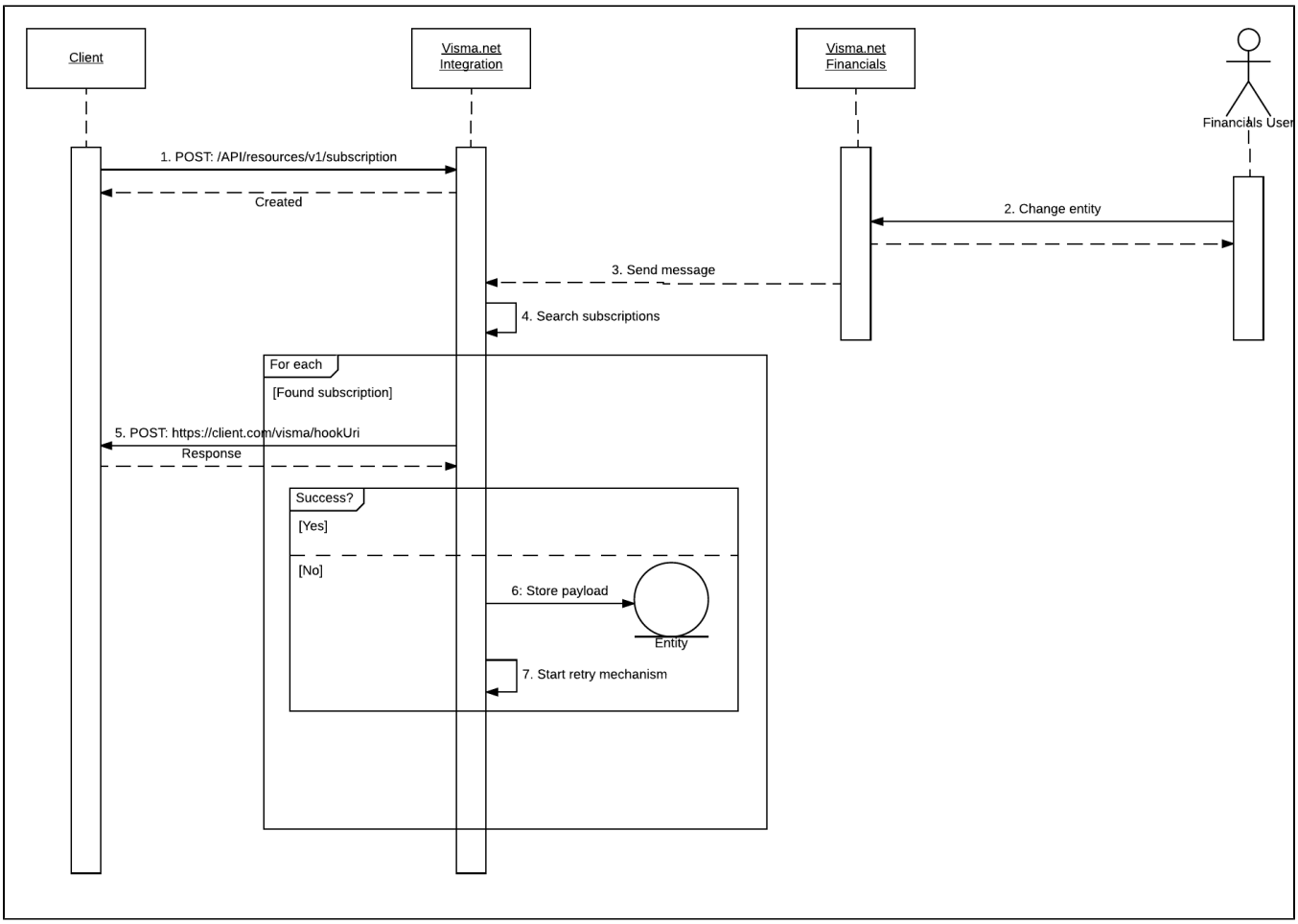
Since any user that has regular access to the API can also subscribe to the exposed events, the subscription resource will use the same authorization mechanism as the other REST resources. This means that the requests on */API/resources/v1/subscription* endpoint MUST have the `Authorization` header set with value "Bearer xxx".

We also recommend clients to use TLS (https) on the registered "hookUri".

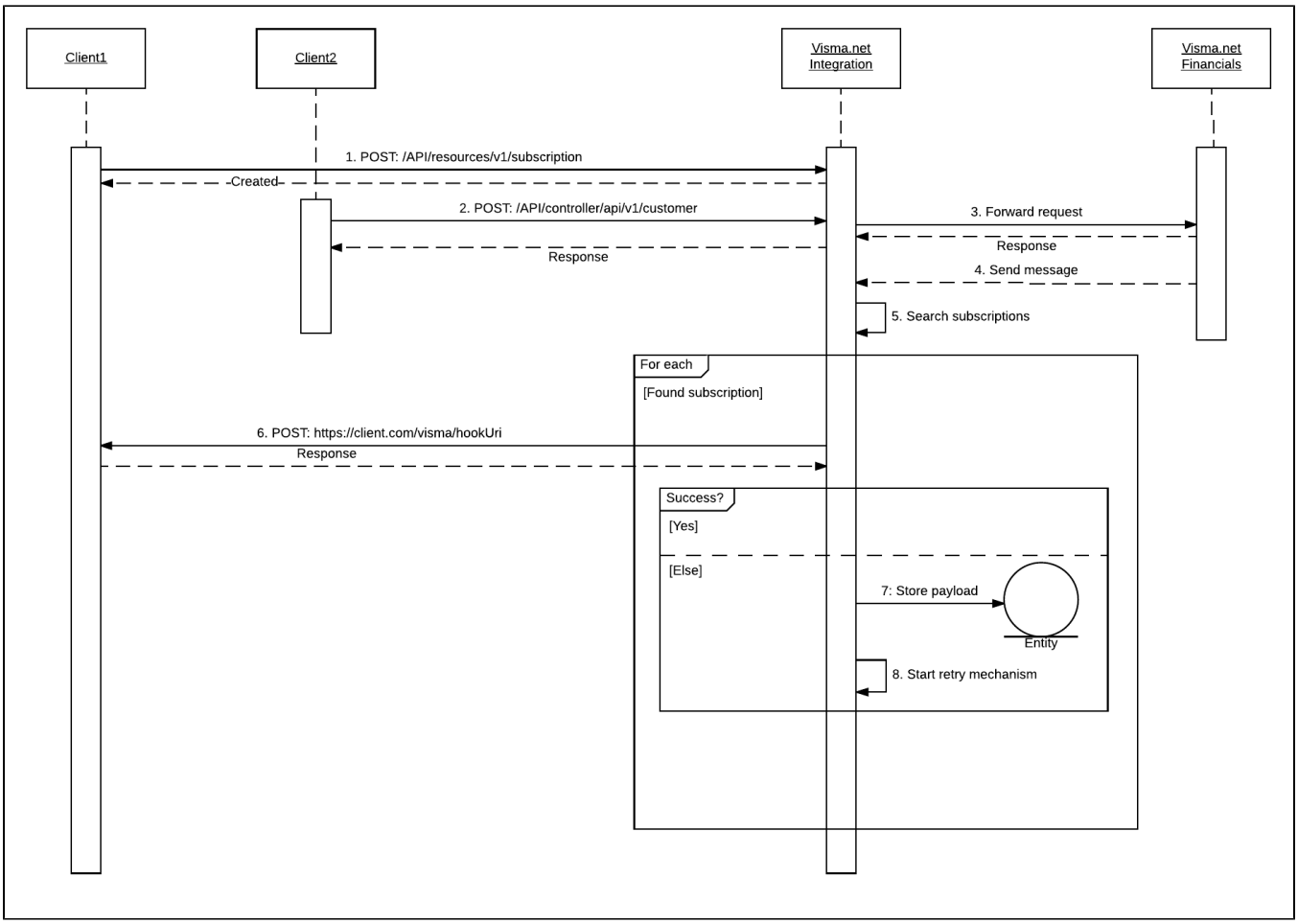
The notifications that we send are "skinny" containing only the identifiers ("resourceUri" or "resourceId") of the changed resource. The client will have to use these identifiers together with a valid token to GET the resource representation.

Flows:

Flow 1:


Client
Visma.net Integration
Visma.net Financials
Financials User
1. POST: /API/resources/v1/subscription
Created
2. Change entity
3. Send message
4. Search subscriptions
For each
[Found subscription]
5. POST: https://client.com/visma/hookUri
Response
Success?
[Yes]
[No]
6: Store payload
Entity
7. Start retry mechanism

Flow 2


Client1
Client2
Visma.net Integration
Visma.net Financials
1. POST: /API/resources/v1/subscription
Created
2. POST: /API/controller/api/v1/customer
Response
3. Forward request
Response
4. Send message
5. Search subscriptions
For each
[Found subscription]
6. POST: https://client.com/visma/hookUri
Response
Success?
[Yes]
[Else]
7: Store payload
Entity
8. Start retry mechanism