DATA VIRTUALITY MASTERCLASS

Topic: Receiving Webhooks

Welcome to the DV Masterclass! Agenda

Day 1

2:00pm - 3:00pm: Analytical storage considerations

3:00pm - 3:15pm: Break

3:15pm - 4:00pm: Support stories

4:00pm - 4:15pm: Break

4:15pm - 5:00pm: Logging with log4j

Day 2

2:00pm - 3:00pm: Sending formatted reports from Data Virtuality

3:00pm - 3:15pm: Break

3:15pm - 4:00pm: Updating Data Virtuality Server

4:00pm - 4:15pm: Break

4:15pm - 4:45pm: Receiving webhooks

What to expect from this session?

In this track, we will receive a webhook with JSON payload and call a stored procedure from it

- Problem statement of the use case
- REST API procedure structure including passing of parameters
- Calling it from Postman
- Parsing the desired information from JSON
- Acting on the result
- Possible use cases

Problem statement of the use case

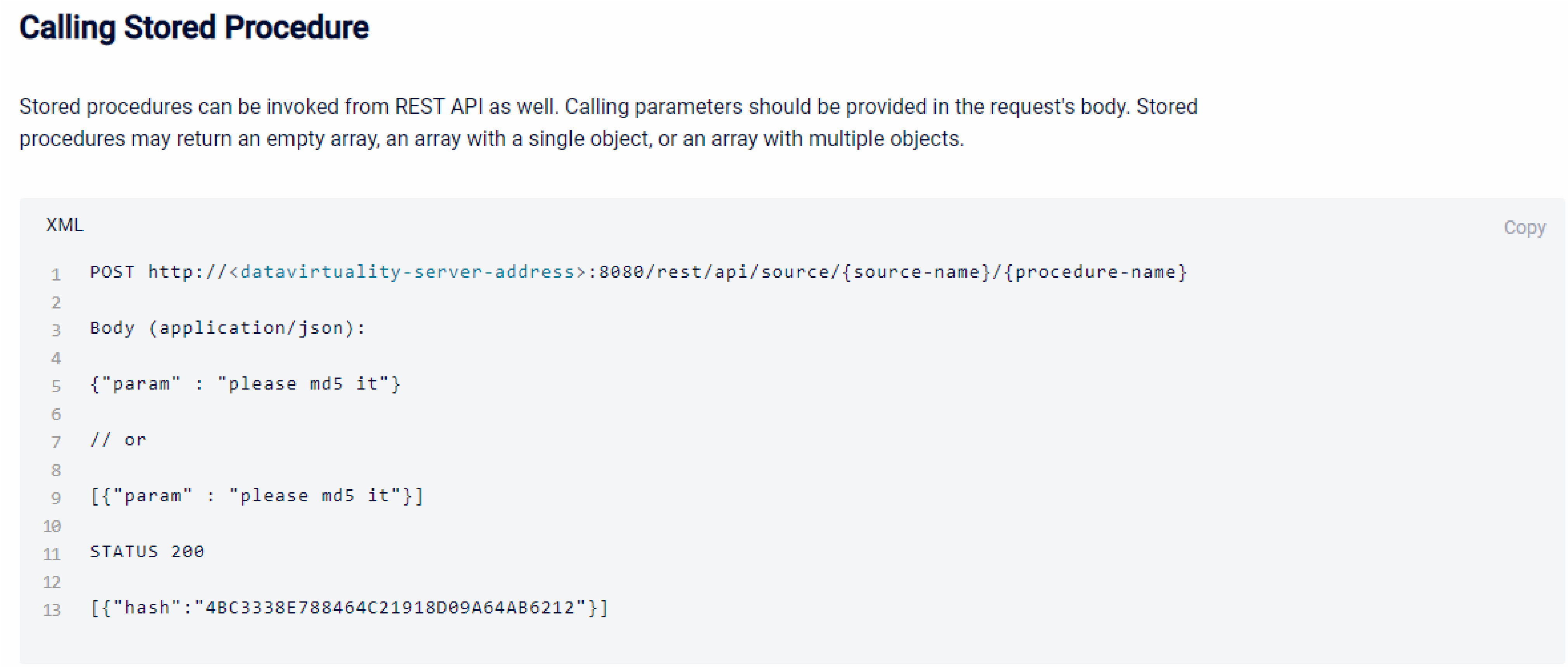
Problem statement of the use case

- Our client is using stripe
- Stripe has a new function where it can send a link to the result csv file to a webhook
- This process is faster than the API (100 items per page, with forced waiting time)

REST API procedure structure including passing of parameters

REST API procedure structure including passing of parameters

- API docs available in https://documentation.datavirtuality.com/24/user-guide/connecting-to-data-virtuality-server/rest-api/data-sources



Calling Stored Procedure

Stored procedures can be invoked from REST API as well. Calling parameters should be provided in the request's body. Stored procedures may return an empty array, an array with a single object, or an array with multiple objects.

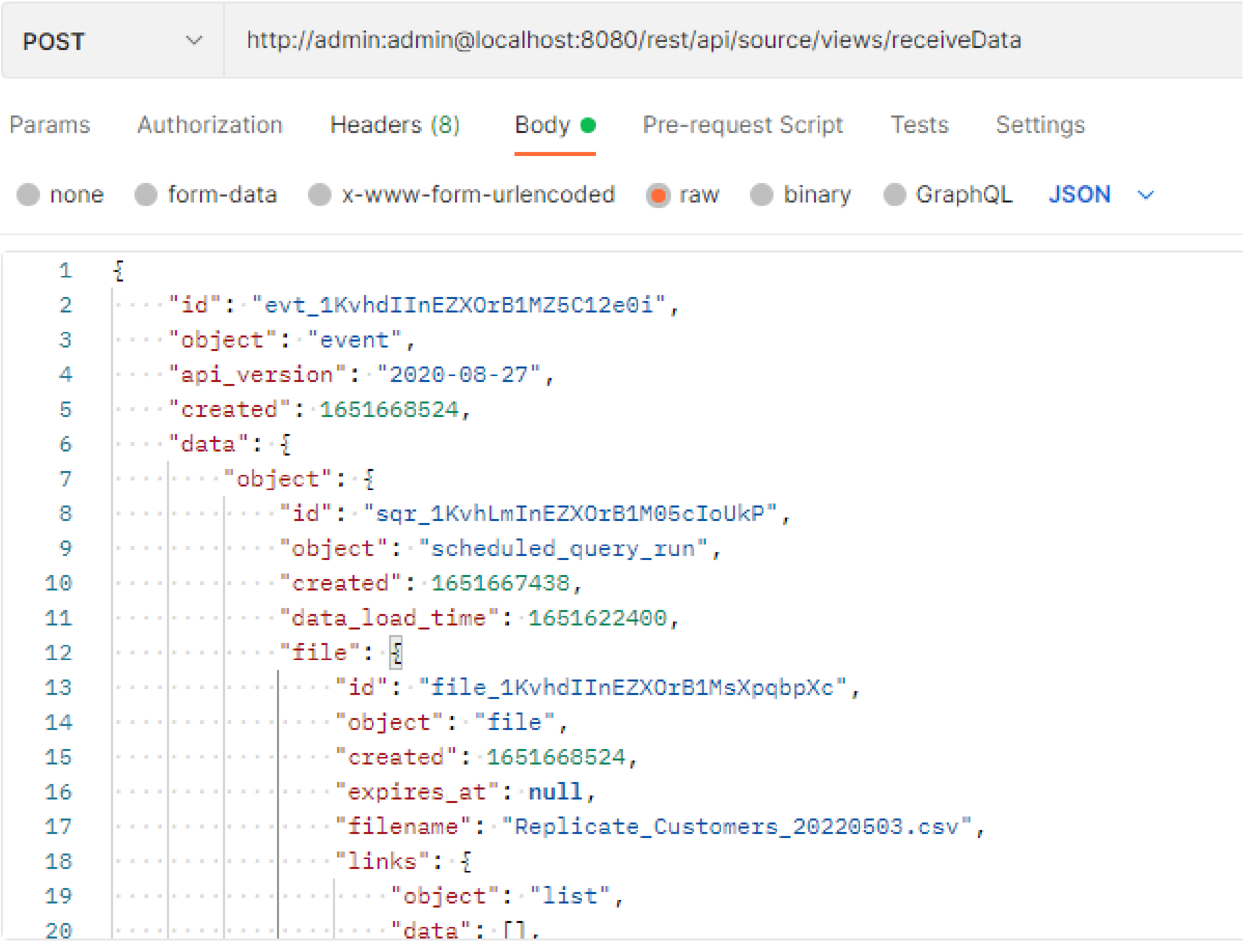
XML

Copy

```
POST http://<datavirtuality-server-address>:8080/rest/api/source/{source-name}/{procedure-name}

Body (application/json):

{"param" : "please md5 it"}

// or

[{"param" : "please md5 it"}]

STATUS 200

[{"hash":"4BC3338E788464C21918D09A64AB6212"}]
```

Calling it from Postman

Calling it from postman



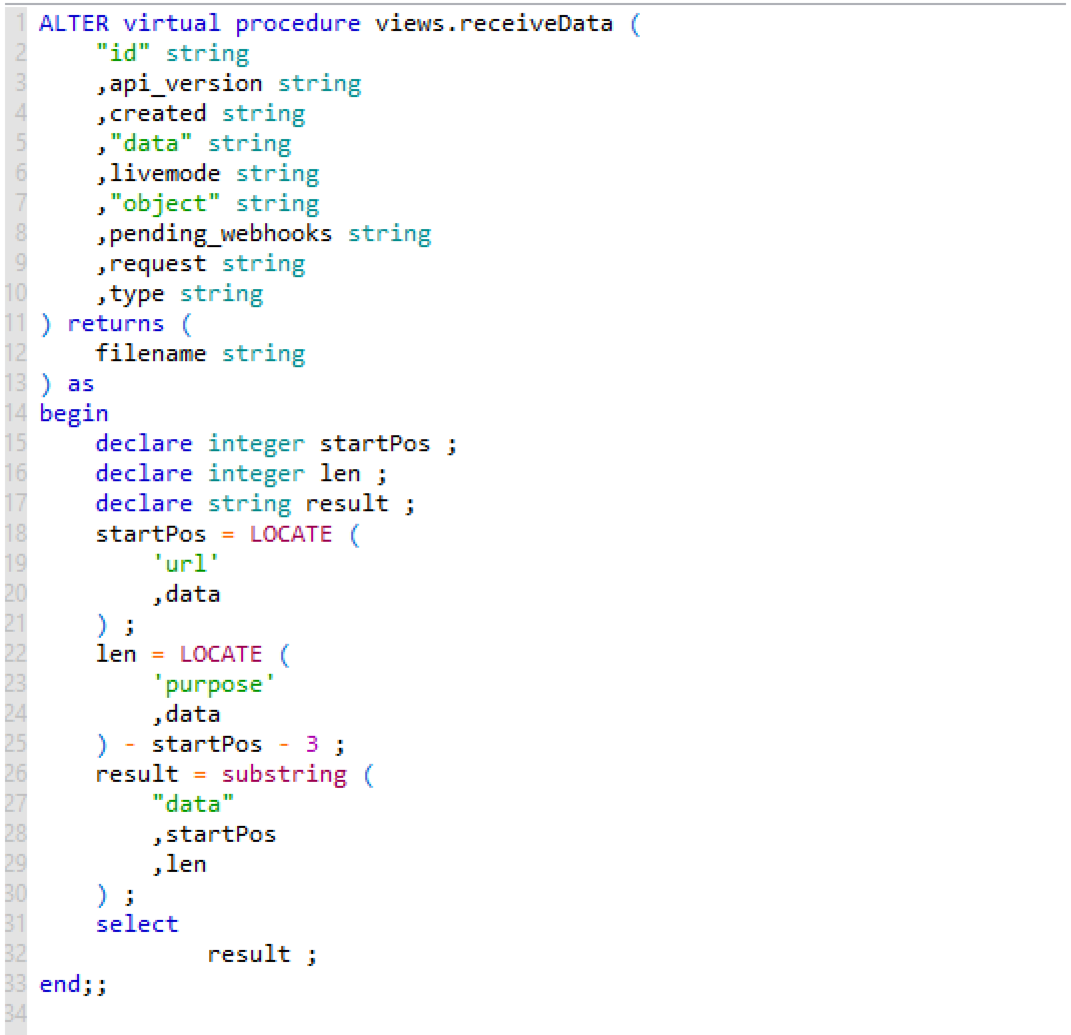
POST http://admin:admin@localhost:8080/rest/api/source/views/receiveData

○ none ○ form-data ○ x-www-form-urlencoded ● raw ○ binary ○ GraphQL JSON

```
{
    "id": "evt_1KvhdIInEZXOrB1MZ5C12e0i",
    "object": "event",
    "api_version": "2020-08-27",
    "created": 1651668524,
    "data": {
        "object": {
            "id": "sqr_1KvhLmInEZXOrB1M05cIoUkP",
            "object": "scheduled_query_run",
            "created": 1651667438,
            "data_load_time": 1651622400,
            "file": {
                "id": "file_1KvhdIInEZXOrB1MsXpqbpXc",
                "object": "file",
                "created": 1651668524,
                "expires_at": null,
                "filename": "Replicate_Customers_20220503.csv",
                "links": {
                    "object": "list",
                    "data": [],
```

Parsing the desired information from JSON

Parsing the filename out (hack)



```
ALTER virtual procedure views.receiveData (
    "id" string
    ,api_version string
    ,created string
    ,"data" string
    ,livemode string
    ,"object" string
    ,pending_webhooks string
    ,request string
    ,type string
) returns (
    filename string
) as
begin
    declare integer startPos ;
    declare integer len ;
    declare string result ;
    startPos = LOCATE (
        'url'
        ,data
    ) ;
    len = LOCATE (
        'purpose'
        ,data
    ) - startPos - 3 ;
    result = substring (
        "data"
        ,startPos
        ,len
    ) ;
    select
            result ;
end;;

```

Demo: Webhook call / Further use cases

Any feedback / questions?

Thank you!

Please feel free to contact us at:
presales@datavirtuality.com

or

visit us at:
datavirtuality.com