



How to use Qlik NPrinting APIs inside a Qlik Sense load script

Introduction

The purpose of this document is to show how to use Qlik NPrinting APIs inside a Qlik Sense load script using the Qlik Sense REST Connector.

A little bit of context...

Triggering Qlik NPrinting APIs inside a Qlik Sense load script is one of the most wanted features in the NPrinting roadmap.

The funny part is that this is already almost entirely doable using the out-of-the-box Qlik Sense REST Connector.

This tutorial will show how to perform the following operations inside a Qlik Sense load script:

- Retrieve the list of available NPrinting Apps
- Select a specific NPrinting App
- Retrieve the NPrinting Connections related to a specific NPrinting App
- Trigger an NPrinting Reload Metadata
- Perform a polling query in order to wait until the Metadata generation is completed
- Once the Metadata generation is completed, trigger an NPrinting Publish Task

At the end everything can eventually be triggered inside a Qlik Sense Reload Task.

The concepts explained in this document are applicable to call all the available NPrinting APIs;

All the examples shown in this tutorial has been tested in a QlikView script as well.

Environment

NPrinting environment:

- Single machine containing both QlikView NPrinting 17 Server and QlikView NPrinting 17 Engine (Qlik NPrinting February 2018 TP)
- CPU: Dual 2.20 GHz Intel Xeon(R)ES-4620 (two cores)
- RAM: 8 GB
- HD: 100 GB

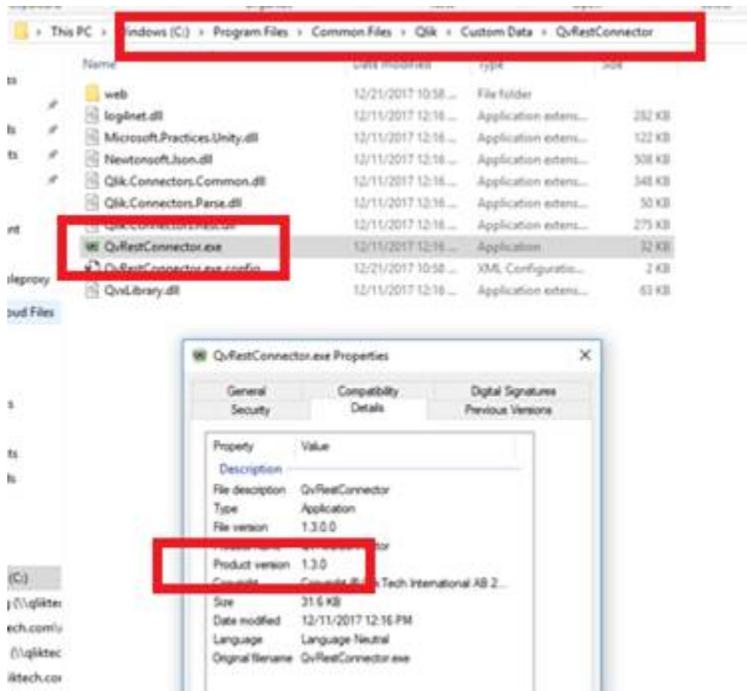
Sense environment:

- Single Qlik Sense server (Qlik Sense February 2018 TP)
- CPU: Dual 2.20 GHz Intel Xeon(R)ES-4620 (two cores)
- RAM: 8 GB
- HD: 100 GB

In Sense versions previous the February 2018 release there was a bug that prevented the concepts shown in this document to work. So make sure that you are using at least the Qlik Sense February 2018 Technical Preview.

You also need Qlik REST Connector 1.3 (ships with Qlik Sense Feb 2018) as a minimum. It is a separate download for QlikView.

To check your version of the REST CONNECTOR see the screenshot below



Reports and Documents used

As Qlik Sense document we used the usual Executive Dashboard demo application.

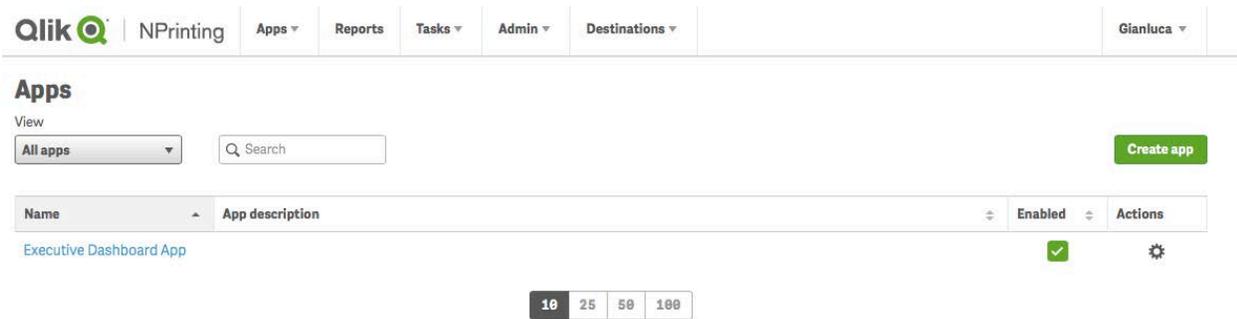
As Qlik NPrinting report we used a pretty simple one based upon the Executive Dashboard demo application.

Qlik NPrinting configuration

Before to get into the actual procedure, let's take a look at the Qlik NPrinting configuration.

Apps

We created a single NPrinting App called "Executive Dashboard App":



The screenshot shows the Qlik NPrinting configuration interface. At the top, there is a navigation bar with the Qlik logo, 'NPrinting', and several menu items: 'Apps', 'Reports', 'Tasks', 'Admin', and 'Destinations'. The user's name 'Gianluca' is displayed on the right. Below the navigation bar, the 'Apps' section is active. It features a 'View' dropdown set to 'All apps', a search input field, and a 'Create app' button. A table lists the configured apps:

Name	App description	Enabled	Actions
Executive Dashboard App		<input checked="" type="checkbox"/>	

At the bottom of the table, there is a pagination control with buttons for 10, 25, 50, and 100 items per page.

Connections

We created a single NPrinting Connection called “Executive Dashboard Connection”. This is, of course, a Connection included in the “Executive Dashboard App” NPrinting App and pointing to the Qlik Sense “Executive Dashboard” document:

The screenshot shows the configuration page for a connection in Qlik NPrinting. The breadcrumb trail is 'Connections / Executive Dashboard Connection'. The page title is 'Executive Dashboard Connection'. There are buttons for 'Cache status' (set to 'Generated'), 'Reload metadata', and a refresh icon. Below the title are tabs for 'Overview' and 'Triggers'. The main form is titled 'Connection' and contains the following fields:

- Name: Executive Dashboard Connection
- Description: (empty text area)
- App: Executive Dashboard App
- Source: QlikView, Qlik Sense
- Proxy address: https://rd-gpr-sense.rlund.qliktech.com/
- Sense app ID: c6131d9c-7ce7-4d2c-b87c-265e9df9e7bf
- Identity: QTSELDGPR
- Apply user section access for reports
- The cache will be generated by applying the configured identity: QTSELDGPR
Reports will be produced by applying the configured identity: QTSELDGPR
- Buttons: Verify connection, Run verification, Cancel, Save

Reports

We created a single report with just two images inside:

The screenshot shows the configuration page for a report in Qlik NPrinting. The breadcrumb trail is 'Reports / Executive Dashboard PPT Report'. The page title is 'Executive Dashboard PPT Report'. There are buttons for 'Export', 'Replace', 'Edit template', and a refresh icon. Below the title are tabs for 'Overview' and 'Filters'. The main form is titled 'Report' and contains the following fields:

- Title: Executive Dashboard PPT Report
- Description: (empty text area)
- Type: PowerPoint
- App: Executive Dashboard App
- Enabled
- Enable On-Demand and API report generation
- Enable cycle
- Enable dynamic naming
- Buttons: Cancel, Save

Publish Tasks

The screenshot shows the 'User' configuration page in Qlik NPrinting. The user is named 'Gianluca' with email 'gorg@qlik.com'. The 'Domain account' is 'QITSELDPR' and is checked as a 'Valid NT Domain user'. The 'Time zone' is 'Europe/Rome' and the 'Locale' is 'English'. The 'Enabled' checkbox is checked. There are 'Cancel' and 'Save' buttons at the bottom right.

We created a single Publish Task for a single recipient with Newsstand and email destination.

The screenshot shows the 'Publish task' configuration page. The task name is 'Executive Dashboard Publish Task'. The 'App' is 'Executive Dashboard App'. The 'Days to keep' and 'Reports to keep' are both set to '1'. The 'Enabled' checkbox is checked. There are 'Cancel' and 'Save' buttons at the bottom right.

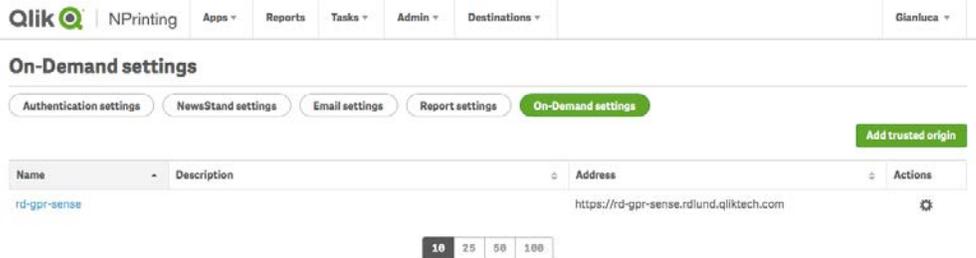
Users

In order to make the two environments (Sense and NPrinting) to talk each other, we need a user recognized by both systems. In NPrinting we need to make sure that such user has fully control over the entities that we will handle during this tutorial. The quickest way is to grant such user with the "Administrator" role.

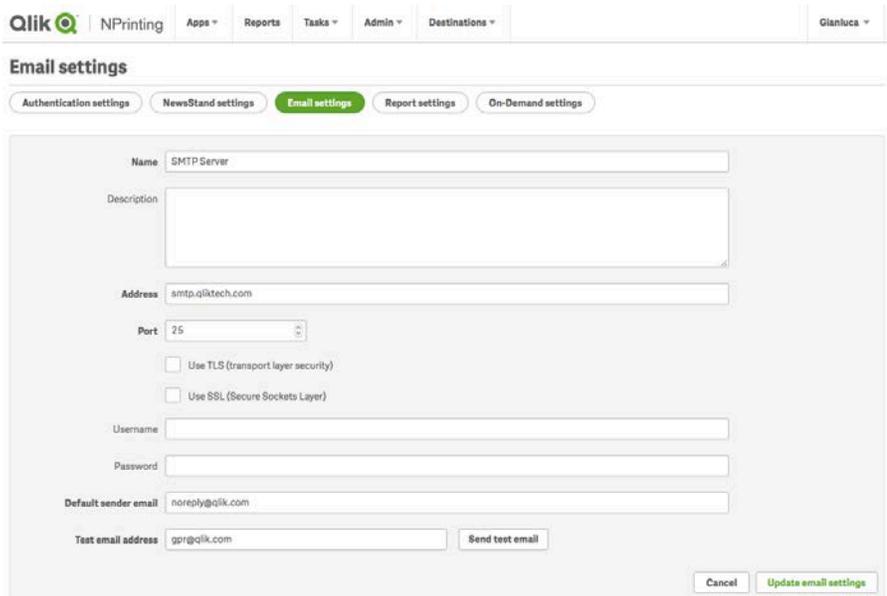
Another needed setting is the “Domain Account” that must be filled with a valid NT user. This NT user will be the one used to authenticate the Qlik Sense REST Connector (see the “NP Login” section below).

Other Settings

In order to accept POST calls from the Sense server using CORS, we need to list the Sense server's hostname among the NPrinting Trusted Origins:



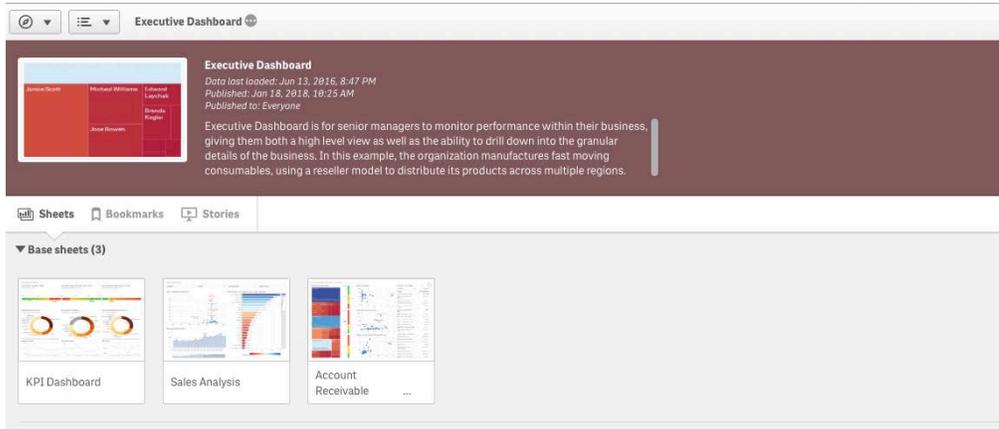
To use email destinations inside NPrinting Publish Tasks we need to configure an SMTP server:



Qlik Sense Configuration

Documents

As we said for this tutorial we used the usual “Executive Dashboard” demo document:



The REST Connectors

In order to perform different GET and POST calls towards dynamic URLs we used a simple trick, we created two different connections, using the REST Connector, to the same endpoint, one for the GET, one for the POST; we then took advantage of the “WITH CONNECTION” instruction in the Sense Load Script in order to override the desired endpoints (it will become more clear in the next sections...)

GET REST Connection

This is a screenshot of the 'Edit connection (REST)' configuration window in Qlik Sense. The 'Request' field contains the URL 'https://NPrintingServer.domain.com:4993/api/v1/login/ntlm'. The 'Timeout' is set to '30'. The 'Method' is set to 'GET'. Under 'Data options', 'Auto detect response type' is checked, and 'Key generation strategy' is set to 'Sequence ID'. Under 'Authentication', 'Authentication Schema' is 'Windows NTLM', 'User name' is 'QTSEL\GPR', and 'Name' is 'NPrinting REST Login (qtset_gpr)'. At the bottom, there are buttons for 'Test Connection', 'Cancel', and 'Save'.This is a screenshot of the 'Edit connection (REST)' configuration window in Qlik Sense, showing the 'Additional request parameters' section. The 'Skip server certificate validation' checkbox is checked. 'Use certificate' is set to 'No'. There are sections for 'Trusted locations', 'Additional request parameters', 'Query parameters', and 'Query headers', each with a table for Name and Value. The 'Name' field for the connection is 'NPrinting REST Login (qtset_gpr)'. At the bottom, there are buttons for 'Test Connection', 'Cancel', and 'Save'.

The GET REST Connection is pretty simple. In the URL input box we inserted the NPrinting login endpoint:

<https://nprintingserver.domain.com:4993/api/v1/login/ntlm>

We selected “GET” as Method.

We selected “Windows NTLM” as Authentication Schema and inserted a valid NTLM credentials. These credentials must match the ones filled during the NPrinting user configuration (see the “Users” section above).

We selected “Skip server certificate validation” since the NPrinting environment doesn’t have a valid SSL certificate installed.

POST REST Connection

The image displays two side-by-side screenshots of the 'Edit connection (REST)' dialog box in a testing tool. The left screenshot shows the 'Request' tab. The URL is 'https://NPrintingServer.domain.com:4993/api/v1/login/ntlm', the Method is 'POST', and the Request body is empty. The 'Data options' section has 'Auto detect response type' checked and 'Key generation strategy' set to 'Sequence ID'. The Name is 'NPrinting REST Login (POST) (qtset_gpr)'. The right screenshot shows the 'Data options' tab. 'Skip server certificate validation' is checked, and 'Use certificate' is set to 'No'. There are empty fields for 'Trusted locations'. Under 'Additional request parameters', there are empty fields for 'Query parameters'. Under 'Query headers', there is one header: 'Origin' with the value 'https://rd-gpr-sense.rdlund.qliktech.com'. The 'Name' field is 'NPrinting REST Login (POST) (qtset_gpr)'. Both screenshots have 'Test Connection', 'Cancel', and 'Save' buttons at the bottom.

The POST REST Connection is very similar to the GET one. In the URL input box we inserted the same NPrinting login endpoint:

<https://nprintingserver.domain.com:4993/api/v1/login/ntlm>

We selected POST as Method.

We left the Request Body text area empty.

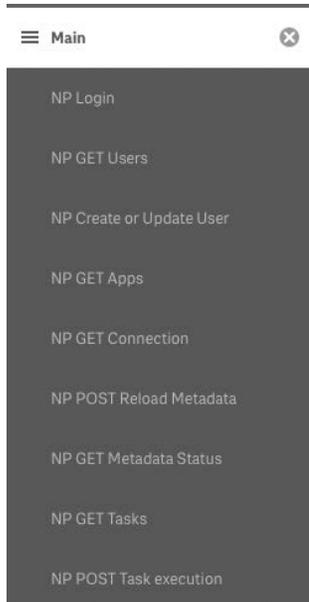
We selected “Windows NTLM” as Authentication Schema and inserted the same NTLM credentials.

We selected “Skip server certificate validation” since the NPrinting environment doesn’t have a valid SSL certificate installed.

Under the “Query headers” section we inserted “Origin” as Name and the Qlik Sense server hostname as value (the same hostname that we inserted in the NPrinting Trusted Origins); this is a requirement to make the connection work using CORS.

The Load Script

We splitted the Load Script in multiple part in order to separate the different steps:



Main

The main section of the script contains all the instructions needed to load the actual data for your Sense application.

```
SET ThousandSep=',';
SET DecimalSep='.';
SET MoneyThousandSep=',';
SET MoneyDecimalSep='.';
SET MoneyFormat='$#,##0.00;($#,##0.00)';
SET TimeFormat='h:mm:ss TT';
SET DateFormat='M/D/YYYY';
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';
SET FirstWeekDay=6;
SET BrokenWeeks=1;
SET ReferenceDay=0;
SET FirstMonthOfYear=1;
SET CollationLocale='en-US';
```

```

SET CreateSearchIndexOnReload=1;
SET MonthNames='Jan;Feb;Mar;Apr;May;Jun;Jul;Aug;Sep;Oct;Nov;Dec';
SET LongMonthNames='January;February;March;April;May;June;July;August;September;October;November;December';
SET DayNames='Mon;Tue;Wed;Thu;Fri;Sat;Sun';
SET LongDayNames='Monday;Tuesday;Wednesday;Thursday;Friday;Saturday;Sunday';

```

//...your Sense stuff goes here

NP Login

This section contains the instructions needed to perform an NPing NTLM login in order to obtain the Session Cookie that we are going to use in all the subsequent calls:

```

//Connect to NPrinting using the GET REST Connection
LIB CONNECT TO 'NPrinting REST Login (qtse_l_gpr)';

//Perform a GET call to NPrinting NTLM login API
RestConnectorMasterTable:
SQL SELECT
    "Set-Cookie",
    "__KEY__response_header"
FROM JSON "_response_header" PK "__KEY__response_header";

[_response_header]:
LOAD [Set-Cookie] AS [Set-Cookie]
RESIDENT RestConnectorMasterTable
WHERE NOT IsNull([__KEY__response_header]);

//Extracts session cookie from the API response
let vCookieRaw = Peek('Set-Cookie',0,'_response_header');
let vCookie = TextBetween('${vCookieRaw}','Secure;', 'Path=/',2);

DROP TABLE RestConnectorMasterTable;

```

Notice how we used the “LIB CONNECT” instruction to establish a “GET” connection using the first Qlik Sense REST Connector configured previously.

The “vCookie” variable now contains the Session Cookie that we are going to include in the HTTP header of the next calls.

NP GET Users

This section contains the instructions needed to check whether a specific user is present in the NPrinting users repository:

```
// Get User with "usera" email
RestUserMasterTable:
SQL SELECT
    "__KEY_data",
    (SELECT
        "id",
        "email",
        "__FK_items"
        FROM "items" FK "__FK_items")
FROM JSON (wrap off) "data" PK "__KEY_data"
WITH CONNECTION( URL "https://nprintingserver.domain.com:4993/api/v1/users", HTTPHEADER "cookie" "$(vCookie)" );

[users_items]:
LOAD [id] AS [users_userId],
    [email] AS [users_userEmail]
RESIDENT RestUserMasterTable
WHERE NOT IsNull(__FK_items) AND SubStringCount([email], 'usera@qlik.com') <> 0;

//Extracts the userId of the desired NP User
let vUserId = Peek('users_userId',0,'users_items');

DROP TABLE RestUserMasterTable;
```

Notice how we didn't need to create another connection to perform this call. We simply reused the GET connector and override the API endpoint using the "WITH CONNECTION" statement. Instead of specifying credentials, we used the Session Cookie retrieved with the Login call using the "HTTPHEADER" option.

This is the trick that we used along the entire experiment.

In the LOAD statement we included a WHERE condition in order to select only the desired NPrinting User.

The "vUserId" variable now contains the desired NPrinting userID.

NP Create or Update User

This section contains the instructions needed to insert a new user if the selected user is not present, or to update it otherwise:

```
if IsNull(vUserId) then
    //create user
    LIB CONNECT TO 'NPrinting REST Login (POST) (qtset_gpr)';

    set vUserBody =
    '{"Username":"UserA","Email":"usera@qlik.com","Password":"123","Enabled":"true","Folder":"","Subfolder":"","DomainAccount":"","Timezone":"Europe/Berlin","Locale":"en"}';
    let vUserBody = replace(vUserBody,'"', chr(34)&chr(34));

    set vPostUserURL = 'https://nprintingserver.domain.com:4993/api/v1/users';

    RestNPPOSTandPUTTestTable:
    SQL SELECT
        "__KEY_data"
    FROM JSON (wrap off) "data" PK "__KEY_data"
    WITH CONNECTION( URL "$(vPostUserURL)", BODY "$(vUserBody)",
        HTTPHEADER "Origin" "https://qliksenseserver.domain.com",
        HTTPHEADER "Content-Type" "application/json",
        HTTPHEADER "cookie" "$(vCookie)");

    [post_and_put_items]:
    LOAD [__KEY_data] AS [__KEY_data]
    RESIDENT RestNPPOSTandPUTTestTable
    WHERE NOT IsNull([__KEY_data]);

    DROP TABLE RestNPPOSTandPUTTestTable;
else
    //update user
    LIB CONNECT TO 'NPrinting REST Login (POST) (qtset_gpr)';

    set vUserBody =
    '{"Username":"UserB","Email":"userb@qlik.com","Password":"123","Enabled":"true","Folder":"","Subfolder":"","DomainAccount":"","Timezone":"Europe/Berlin","Locale":"en"}';
    let vUserBody = replace(vUserBody,'"', chr(34)&chr(34));

    let vPostUserURL = 'https://nprintingserver.domain.com:4993/api/v1/users/'&'$(vUserId)';
```

```

RestNPPOSTandPUTTestTable:
SQL SELECT
    "__KEY_data"
FROM JSON (wrap off) "data" PK "__KEY_data"
WITH CONNECTION( URL "$(vPostUserURL)", BODY "$(vUserBody)",
    HTTPHEADER "Origin" "https://qliksenseserver.domain.com",
    HTTPHEADER "Content-Type" "application/json",
    HTTPHEADER "X-HTTP-Method-Override" "PUT",
    HTTPHEADER "cookie" "$(vCookie)");

[post_and_put_items]:
LOAD [__KEY_data] AS [__KEY_data]
RESIDENT RestNPPOSTandPUTTestTable
WHERE NOT IsNull([__KEY_data]);

DROP TABLE RestNPPOSTandPUTTestTable;
endif;

```

Notice how we needed to switch to the second Qlik Sense REST Connector that we configured at the beginning (see the “LIB CONNECT” instruction). This second connector uses the POST method which is needed to fire a metadata reload.

Once a “POST” connection is established we switched to the right API endpoint using the usual “WITH CONNECTION” trick.

The “IF” condition checks whether or not the user retrieved in the previous section is present in the NPrinting User repository.

In the “IF” section (so if the user has to be inserted) we simply created a JSON body and put it inside the “vUserBody” variable (the “replace” statement is required in order to escape special characters), then we used the “BODY” parameter in the “WITH CONNECTION” section to put the body inside the HTTP request. We also used several “HTTPHEADER” parameters to fill the HTTP request with all the required headers.

In the “ELSE” section (so if the user has to be updated) we dynamically composed the “vPostUserURL” variable in order to point to the specific user URL. Then, as we did for the insert, we put the body inside the HTTP request.

Notice how we changed the REST method from “POST” to “PUT” (we are performing an update of an existing user) using the “X-HTTP-Method-Override” HTTP header.

NP GET Apps

This section contains the instructions needed to get the list of all the available NPrinting Apps. We then selected only the App which name starts with “Executive Dashboard” (you can use whatever logic you want to filter the desired Apps):

```
//GET the list of the NPrinting Apps that match the desired condition
```

```
LIB CONNECT TO 'NPrinting REST Login (qtset_gpr)';
```

```
RestAppsMasterTable:
```

```
SQL SELECT
```

```
    "__KEY_data",
```

```
    (SELECT
```

```
        "id",
```

```
        "name",
```

```
        "__FK_items"
```

```
    FROM "items" FK "__FK_items")
```

```
FROM JSON (wrap off) "data" PK "__KEY_data"
```

```
WITH CONNECTION( URL "https://nprintingserver.domain.com:4993/api/v1/apps", HTTPHEADER "cookie" "$(vCookie)" );
```

```
[apps_items]:
```

```
LOAD [id] AS [apps_appld],
```

```
    [name] AS [apps_appName]
```

```
RESIDENT RestAppsMasterTable
```

```
WHERE NOT IsNull(__FK_items) AND SubStringCount([name], 'Executive Dashboard') <> 0;
```

```
//Extracts the appld of the desired NP App
```

```
let vAppld = Peek('apps_appld',0,'apps_items');
```

```
DROP TABLE RestAppsMasterTable;
```

Notice how we had to switch back to the “GET” connector using the “LIB CONNECT” statement. In the LOAD statement we included a WHERE condition in order to select only the desired NPrinting App. The “vAppld” variable now contains the desired NPrinting applID.

NP GET Connection

This section contains the instructions needed to get all the available NPrinting Connections. We then selected the right connection choosing the one related to the appId obtained in the previous section of the load script:

```
//GET the list of the connections contained in the NP App selected before
RestConnectionMasterTable:
SQL SELECT
    "__KEY_data",
    (SELECT
        "id",
        "name",
        "appId",
        "__FK_items"
    FROM "items" FK "__FK_items")
FROM JSON (wrap off) "data" PK "__KEY_data"
WITH CONNECTION( URL "https://nprintingserver.domain.com:4993/api/v1/connections", HTTPHEADER "cookie" "$(vCookie)" );

[connection_items]:
LOAD [id] AS [connection_id],
    [name] AS [connection_name],
    [appId] AS [connection_appId]
RESIDENT RestConnectionMasterTable
WHERE NOT IsNull(__FK_items) AND [appId] = '$(vAppId)';

//Extracts the desired Connection ID
let vConnectionId = Peek('connection_id',0,'connection_items');

//Compose the URL for the POST call that triggers a reload metadata
let vReloadMetadataURL = 'https://nprintingserver.domain.com:4993/api/v1/connections/'&$(vConnectionId)&'/reload';

//Compose the URL for the GET call that checks the connection status
let vConnectionStatusURL = 'https://nprintingserver.domain.com:4993/api/v1/connections/'&$(vConnectionId);

DROP TABLE RestConnectionMasterTable;
```

Notice how we used the same “WITH CONNECTION” trick to switch to the Connections API. In the LOAD statement we included a WHERE condition in order to select only the NPrinting Connection related to the previously extracted appId. The “vConnectionId” variable now contains the desired NPrinting Connection ID. We used this last variable to compose the dynamic URLs that we are going to use in the next steps.

The “vReloadMetadataURL” variable now contains the URL of the API endpoint that will trigger a metadata reload.

The “vConnectionStatusURL” variable now contains the URL of the API endpoint that we’ll use to poll the connections status.

NP POST Reload Metadata

This section contains the instructions needed to trigger an NPrinting metadata reload:

```
//POST a reload metadata trigger for the desired connection
```

```
LIB CONNECT TO 'NPrinting REST Login (POST) (qtset_gpr)';
```

```
RestNPREloadMetadataTable:
```

```
SQL SELECT
```

```
    "__KEY_data"
```

```
FROM JSON (wrap off) "data" PK "__KEY_data"
```

```
WITH CONNECTION( URL "${vReloadMetadataURL}", HTTPHEADER "cookie" "${vCookie}");
```

```
[metadata_items]:
```

```
LOAD [__KEY_data] AS [__KEY_data]
```

```
RESIDENT RestNPREloadMetadataTable
```

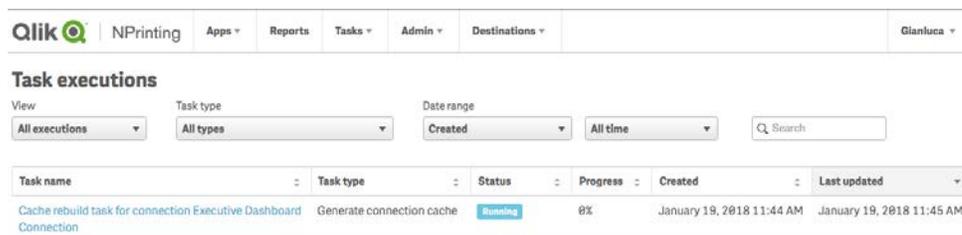
```
WHERE NOT IsNull([__KEY_data]);
```

```
DROP TABLE RestNPREloadMetadataTable;
```

Notice how we needed to switch to the second Qlik Sense REST Connector that we configured at the beginning (see the “LIB CONNECT” instruction). This second connector uses the POST method which is needed to fire a metadata reload.

Once a “POST” connection is established we switched to the right API endpoint using the usual “WITH CONNECTION” trick. Notice how we used the dynamic URL contained in the “vReloadMetadataURL” variable that we created in the previous step.

If you take a look at the NPrinting Task Execution page now you’ll see a Running metadata generation:



The screenshot shows the Qlik Sense NPrinting interface. At the top, there are navigation tabs: NPrinting, Apps, Reports, Tasks, Admin, and Destinations. The user's name, Gianluca, is visible in the top right corner. Below the navigation, the 'Task executions' section is displayed. It includes filters for 'View' (All executions), 'Task type' (All types), 'Date range' (Created), and 'All time'. A search bar is also present. Below the filters, a table lists the task executions. The table has columns for Task name, Task type, Status, Progress, Created, and Last updated. One task is listed: 'Cache rebuild task for connection Executive Dashboard Connection', which is currently 'Running' with 0% progress, created on January 19, 2018 at 11:44 AM, and last updated on January 19, 2018 at 11:45 AM.

Task name	Task type	Status	Progress	Created	Last updated
Cache rebuild task for connection Executive Dashboard Connection	Generate connection cache	Running	0%	January 19, 2018 11:44 AM	January 19, 2018 11:45 AM

NP GET Metadata Status

This section contains the instructions needed to poll the connection status. The purpose of this step is to create a “pause” in the instructions flow waiting for the metadata reload to be completed. This step is needed since we want to make sure that the metadata reload is completed before to move on with the NPrinting Publish Task triggering:

```
LIB CONNECT TO 'NPrinting REST Login (qtsel_gpt)';
let vConnectionStatus = "";

DO while (vConnectionStatus <> 'Generated')
  RestConnectionStatusTable:
  SQL SELECT
  "cacheStatus"
  FROM JSON (wrap off) "data"
  WITH CONNECTION( URL "${vConnectionStatusURL}", HTTPHEADER "cookie" "${vCookie}");

[connection_data]:
LOAD [cacheStatus] AS [connection_cacheStatus]
RESIDENT RestConnectionStatusTable;

let vConnectionStatus = Peek('connection_cacheStatus',0,'connection_data');

DROP TABLE RestConnectionStatusTable;
DROP TABLE [connection_data];
Loop
```

Notice how we needed to switch back to the first REST Connector (see the “LIB CONNECT” instruction) since we need to perform a GET call.

Notice how we used the usual “WITH CONNECTION” trick to switch to the API endpoint contained in the “vConnectionStatusURL” variable created in the “NP GET Connection” section.

We created a “Do - while” loop that checks the “cacheStatus” attribute returned by the API. The loop exits only when the status switches to “Generated”.

NP GET Tasks

This sections contains the instructions needed to obtain the list of all the available NPrinting Publish Tasks. We then selected only the Task related to the previously extracted applD:

RestNPTasksMasterTable:

SQL SELECT

```
"__KEY_data",  
(SELECT  
"id",  
"name",  
"appld",  
"__FK_items"
```

```
FROM "items" FK "__FK_items")
```

```
FROM JSON (wrap off) "data" PK "__KEY_data"
```

```
WITH CONNECTION( URL "https://nprintingserver.domain.com:4993/api/v1/tasks", HTTPHEADER "cookie" "$(vCookie)" );
```

[task_items]:

```
LOAD [id] AS [tasks_taskId],  
[name] AS [tasks_taskName],  
[appld] AS [tasks_appld]
```

```
RESIDENT RestNPTasksMasterTable
```

```
WHERE NOT IsNull(__FK_items) AND [appld] = '$(vAppld)';
```

```
let vTaskId = Peek("tasks_taskId",0,"task_items");
```

```
let vPublishTaskURL = 'https://nprintingserver.domain.com:4993/api/v1/tasks/'&$(vTaskId)&'/executions';
```

```
DROP TABLE RestNPTasksMasterTable;
```

Notice how we used the usual “WITH CONNECTION” trick to switch to the needed API endpoint.
Notice how we used a WHERE condition to filter the Tasks list in order to find the one related to the applD contained in the “vAppld” variable.

The “vTaskId” variable now contains the ID of the selected NPrinting Publish Task

The “vPublishTaskURL” variable now contains the URL of the API endpoint that will trigger the desired Publish Task.

NP POST Task execution

This section contains the instructions needed to trigger the desired NPrinting Publish Task:

```
LIB CONNECT TO 'NPrinting REST Login (POST) (qtsele_gpr)';
```

RestNPTaskTriggerTable:

```
SQL SELECT
```

```
    "__KEY_data"
```

```
FROM JSON (wrap off) "data" PK "__KEY_data"
```

```
WITH CONNECTION( URL "${vPublishTaskURL}", HTTPHEADER "cookie" "${vCookie}");
```

[_post_items]:

```
LOAD [__KEY_data] AS [__KEY_data]
```

```
RESIDENT RestNPTaskTriggerTable
```

```
WHERE NOT IsNull([__KEY_data]);
```

```
DROP TABLE RestNPTaskTriggerTable;
```

Notice how we needed to switch again to the “POST” REST Connector using the “LIB CONNECT” statement.

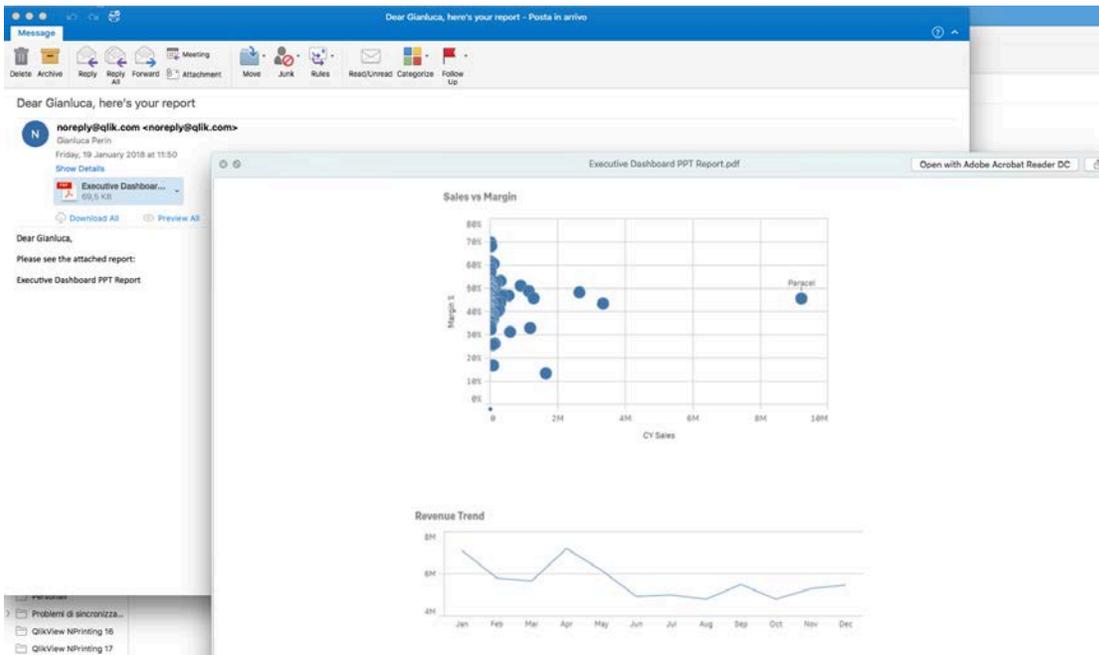
Notice how we used the usual “WITH CONNECTION” trick to switch to the API endpoint contained in the “vPublishTaskURL” variable.

If you take a look at the NPrinting Task Execution page now you’ll see a Running Publish Task:

The screenshot shows the Qlik NPrinting interface. At the top, there are navigation tabs: Apps, Reports, Tasks, Admin, and Destinations. The user's name 'Gianluca' is visible in the top right. Below the navigation is the 'Task executions' section. It includes filters for 'View' (All executions), 'Task type' (All types), 'Date range' (Created), and 'All time'. A search bar is also present. Below the filters is a table with the following data:

Task name	Task type	Status	Progress	Created	Last updated
Executive Dashboard Publish Task	Publish report	Running	0%	January 19, 2018 11:49 AM	January 19, 2018 11:50 AM

Now you can check the configured email or Newsstand destination for the final result:



Putting it all together

You don't need to reload the Qlik app in the same script that you invoke NPrinting.

This makes for some very nice task chaining possibilities and separation of events managed through the Qlik Sense QMC or QlikView QMC with publisher. See the next section for an example

Task chaining

Below is screenshot of the Qlik Sense QMC -> Tasks.

Two apps are used in this example:

“Reload Sales Discovery” is a task to reload a standard Qlik Sense App called “sales discovery”. It runs on a schedule every hour (or every day etc...)

“Distribute NPrinting Reports for Sales discovery” is a task with an event trigger tied to “Reload Sales Discovery”. Upon successful completion of “reload Sales discovery”, “Distribute NPrinting reports for sales discovery” will kick off.

“Distribute NPrinting reports for sales discovery” reloads a 2nd Qlik Sense app call “Publish Nprinting tasks”. “Publish Nprinting Tasks” only has a load script to login to NPrinting and initiate an existing NPrinting publish task.

With this feature, we can task chain NPrinting API calls to run Nprinting reports, update metadata, update users upon completion of a qlik sense reload task.

The Sense QMC becomes the CENTRAL schedule management tool for managing all operational qlik sense and nprinting tasks.

The same is possible with QlikView publisher as well !

The screenshot shows the Qlik Sense QMC interface. At the top, there's a 'Tasks' header with 'Total: 8', 'Showing: 2', and 'Selected: 0'. Below this is a table of tasks:

Name	Associated resource	Type	Enabled	Status	Last execution	Next execution	Tags
Reload Sales Discovery	Sales Discovery	Reload	Yes	Success	2018-02-05 10:36	2018-02-05 11:36	
Distribute NPrinting Reports for Sales Discovery	Publish NPrinting Task	Reload	Yes	Success	2018-02-05 10:36	On task event trigger	

Red arrows point from the 'Associated resource' column to the 'Edit reload task' configuration page. The configuration page shows the following details:

- IDENTIFICATION:** Name: Distribute NPrinting Reports for Sales Discovery, App: Publish NPrinting Task.
- EXECUTION:** Enabled: , Task session timeout (minutes): 1440, Max retries: 0.
- Triggers:** Name: Distribute after Successful Reload, Type: Task event, Enabled: .
- Trigger - Start on task event:** Trigger name: Distribute after Successful Reload, Enabled: .
- TIME CONSTRAINT:** Seconds: 0, Minutes: 360, Hours: 0, Days: 0.
- Tasks:** A dropdown menu shows 'Task successful' and a task named 'Reload Sales Discovery' is listed.

:ript 23

which have been

What about Qlikview ?

QlikView supports the same REST connector as Qlik Sense. The only difference with QlikView script is that data connections are defined differently.

Instead of a connection statement as follows:

```
LIB CONNECT TO 'NPrinting REST Login (qtsel_gpr)';
```

You will have a CUSTOM CONNECT statement like this:

```
SET ThousandSep=',';
SET DecimalSep='.';
SET MoneyThousandSep=',';
SET MoneyDecimalSep='.';
SET MoneyFormat='$$#,##0.00;($#,##0.00)';
SET TimeFormat='h:mm:ss TT';
SET DateFormat='M/D/YYYY';
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';
SET MonthNames='Jan;Feb;Mar;Apr;May;Jun;Jul;Aug;Sep;Oct;Nov;Dec';
SET DayNames='Mon;Tue;Wed;Thu;Fri;Sat;Sun';

CUSTOM CONNECT TO "Provider=QvRestConnector.exe;url= https://NPrintingServer.domain.com:4993/api/v1/login/ntlm ;timeout=30;method=GET;autoDetectResponseType=true;keyGeneration:
RestConnectorMasterTable:"
```

What about deleting a user ?

See the following script which rely on a "POST" connector:

```
let vDeleteUserURL = 'https://nprintingserver.domain.com:4993/api/v1/users/' & $(vUserId);
```

```
if not IsNull(vUserId) then
  RestNPDeleteUserTable:
```

```
SQL SELECT
  "__KEY_data"
FROM JSON (wrap off) "data" PK "__KEY_data"
WITH CONNECTION( URL "$(vDeleteUserURL)",
  HTTPHEADER "Origin" "https://qliksenseserver.domain.com",
  HTTPHEADER "Content-Type" "application/json",
  HTTPHEADER "X-HTTP-Method-Override" "DELETE",
  HTTPHEADER "cookie" "$(vCookie)");
```

```

[delete_user_items]:
LOAD [__KEY_data] AS [__KEY_data]
RESIDENT RestNPDeleteUserTable
WHERE NOT IsNull([__KEY_data]);

DROP TABLE RestNPDeleteUserTable;
endif;

```

Basically you can use the “X-HTTP-Method-Override” to change the REST method to “DELETE”

With HTTP ?

If you are using NPrinting without SSL then there are changes to the script, connection definitions and cookie

1. Connection:

Change both REST connections URLs to use HTTP instead of HTTPS

Change:
https://NPrintingServer.domain.com:4993/api/v1/login/ntlm
to:
http://NPrintingServer.domain.com:4993/api/v1/login/ntlm

2. Script

Change the NPrinting Server URIs to replace https with http

from
let vPublishTaskURL = 'https://nprintingserver.domain.com:4993/api/v1/tasks/' & \$(vTaskId) & '/executions';
to
let vPublishTaskURL = 'http://nprintingserver.domain.com:4993/api/v1/tasks/' & \$(vTaskId) & '/executions';

3. Cookie

Change the vCookie logic as follows, the cookie will not have the word ‘Secure’ in it because its HTTP.

```

let vCookieRaw = Peek('Set-Cookie',0,'_response_header');
let vCookie = TextBetween($(vCookieRaw),'Secure,','Path=',2);

let vCookieRaw = peek('Set-Cookie',0,'_response_header');
let vCookie=TextBetween($(vCookieRaw),'HttpOnly,','Path=',1);

```