Welcome to our webinar!

- This webinar starts in a moment - please stay tuned
- This webinar will be recorded
- You will get the slides, recording and SQL snippets
- During the webinar, you may ask questions using the Q&A button - you may ask questions anonymously
- You can hop on and off on all the sessions

Day 2	15:00 - 16:00 Synchronization Methods 16:15 - 17:15 Working with Connectors
Day 3	15:00 - 16:00 Self-service Troubleshooting 16:15 - 17:15 Job Management Outlook

DATA VIRTUALITY MASTERCLASS

Topic: Working With Connectors

What to expect from this session?

This track will tell you how to manage data sources based on the new modular connectors approach. The following topics will be covered:

- difference between SQL- and modular connectors
- Deploying connectors, creating connections
- using the target table parameter
- use case presentation with the facebook connector
- custom requests without templates and the without target table parameter

data virtuality

Motivation for modular connectors

Challenges with SQL connectors

data virtuality

- Full implementation imported on the client side via SQL script
- Customers often modify this code
- Update can also be provided as an SQL connector
- No issues when unmodified: drop virtual schema, execute SQL script
- Big challenge with modification:
  - DV support agent does not know what was changed
  - above process would remove custom implementation
  - merge hell

Solution: modular connectors

- Modular connectors have the same SQL implementation
- BUT they are compiled into a .JAR (Java ARchive)
- When installed, will behave like a data source with tables and procedures
- Clients can’t modify the code
- But clients can call the provided procedures from their own views/procedures
- Please try not to use _internal procedures, you have been warned
- Advantages: easier update (replace the jar file) and better usability - instead of ws data source + virtual schema

Modular connector deployment

Workflow: deploying a modular connector

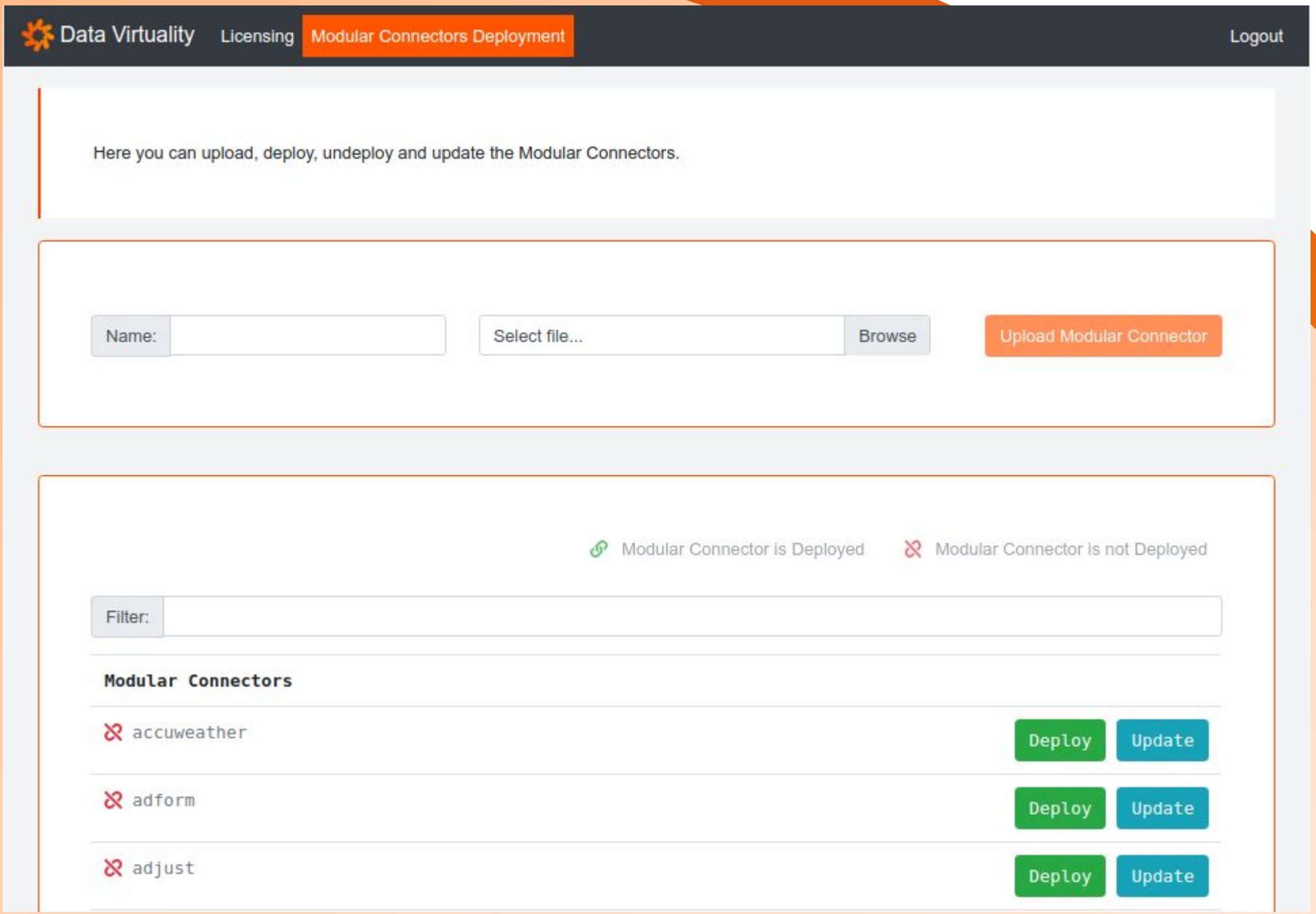
- v2.1
  - all shipped connectors are already deployed
  - copy new jars to .JAR to dvserver/standalone/deployments,
  - get name via select * from "SYSADMIN.DVConnectors";
- v2.3
  - copy .JAR to dvserver/standalone/deployments (all shipped connectors are already there but undeployed)
  - get the name of the connector and deploy it

```
call "SYSADMIN.deployModularConnector"(
    "name" => 'accuweather',
    "deploy" => true
);;
```

- v2.3.7
  - use web admin panel (demo)

```
1 select * from "SYSADMIN.ModularConnectors";;
```

name	deployed
accuweather	false
adform-pipes	false
adform	false
adjust-pipes	false


Data Virtuality
Licensing
Modular Connectors Deployment
Logout
Here you can upload, deploy, undeploy and update the Modular Connectors.
Name:
Select file...
Browse
Upload Modular Connector
Modular Connector is Deployed
Modular Connector is not Deployed
Filter:
Modular Connectors
accuweather
Deploy
Update
adform
Deploy
Update
adjust
Deploy
Update

Demo - Modular Connector admin page

Modular connector updates

Workflow: updating a connector

- v2.1-2.3:
  - copy the JAR file to dvserver/standalone/deployments and remove old version
- v2.3.7:
  - use the admin panel
- after that, in both versions:
  - recreate the data source (!) (edit and click okay)

data virtuality

Getting the create statements

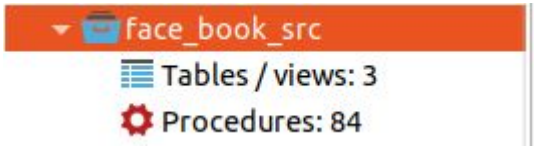
facebook: create the connection

- deploy it and get the name of the connector, then get the create statement

```
call "UTILS.getModularConnectorDatasourceCreateStatement"(
    "connectorName" => 'face_book'/* Optional */
);;
-- RESULT:
EXEC SYSADMIN.createConnection(name => 'face_book_src', jbossCLITemplateName => 'ws', connectionOrResourceAdapterProperties =>
'EndPoint=https://graph.facebook.com/v8.0,AccessToken=<AccessToken>') ;;

EXEC SYSADMIN.createDataSource(name => 'face_book_src', translator => 'face_book', modelProperties => 'cleanupMethod=DELETE,jobTimeout=0',
translatorProperties => '') ;;
```

- Put in your access token and run the create* procedures
- Result:

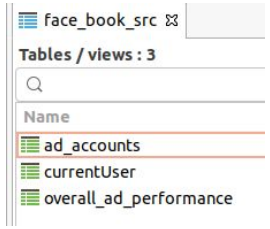


Demo - getting the modular connector create statement

Investigating the connector features

facebook: Using the views

- Simply select from the views
- They will make a live request to the facebook API




```
SELECT "account_id", "id", "name" FROM "face_book_src.ad_accounts" LIMIT 500;;
```

account_id	id	name
45	act_45	Matthias Korn
64	act_64	Data Virtuality

- Cool LDW use case: overall_ad_performance, live status of facebook ads, can be used in a live dashboard

```
SELECT * FROM "face_book_src.overall_ad_performance";;
```

k	cpc	cpm	cpp	ctr	date_start	date_stop	estimated_ad_recall_rate	estimated_ad_recallers	frequency	impressions	inline_link_click_ctr	inline_link_clicks	inline_post_engag
	5.31	25.652174	25.902439	0.48	2014-10-02	2020-09-09	<null>	<null>	1.009756	207	0.483092	1	1
	<null>	28.928571	28.928571	0	2014-10-02	2020-09-09	<null>	<null>	1	56	<null>	0	0
	5.12	23.167421	23.813953	0.45	2014-10-02	2020-09-09	<null>	<null>	1.027907	221	0.452489	1	1
	<null>	53.833333	53.833333	0	2014-10-02	2020-09-09	<null>	<null>	1	60	<null>	0	0
	<null>	15.833333	16.285714	0	2014-10-02	2020-09-09	<null>	<null>	1.028571	72	<null>	0	0
	<null>	26.41791	26.41791	0	2014-10-02	2020-09-09	<null>	<null>	1	67	<null>	0	0
	<null>	19.690722	20.105263	0	2014-10-02	2020-09-09	<null>	<null>	1.021053	97	<null>	0	0
	<null>	30.526316	30.992366	0	2014-10-02	2020-09-09	<null>	<null>	1.015267	133	<null>	0	0
	<null>	15.05618	15.05618	0	2014-10-02	2020-09-09	<null>	<null>	1	89	<null>	0	0
	3.652	17.097718	19.48	0.46	2014-10-02	2020-09-09	<null>	<null>	1.139333	1709	0.117028	2	2
	5.26	21.825726	22.288136	0.41	2014-10-02	2020-09-09	<null>	<null>	1.021186	241	<null>	0	0
	<null>	24.935065	25.263158	0	2014-10-02	2020-09-09	<null>	<null>	1.013158	77	<null>	0	0
	<null>	31.184211	32.465753	0	2014-10-02	2020-09-09	<null>	<null>	1.041096	76	<null>	0	0
	<null>	26.129032	26.129032	0	2014-10-02	2020-09-09	<null>	<null>	1	62	<null>	0	0
	6	22.140221	23.255814	0.36	2014-10-02	2020-09-09	<null>	<null>	1.050388	271	0.369004	1	1
	<null>	18.25	19.210526	0	2014-10-02	2020-09-09	<null>	<null>	1.052632	80	<null>	0	0
	<null>	24.492754	24.492754	0	2014-10-02	2020-09-09	<null>	<null>	1	69	<null>	0	0
	<null>	13.75	14.444444	0	2014-10-02	2020-09-09	<null>	<null>	1.050505	104	<null>	0	0
	<null>	14.029851	14.6875	0	2014-10-02	2020-09-09	<null>	<null>	1.046875	67	<null>	0	0
	<null>	18.607595	18.846154	0	2014-10-02	2020-09-09	<null>	<null>	1.012821	79	<null>	0	0
	<null>	12.903226	13.114754	0	2014-10-02	2020-09-09	<null>	<null>	1.016393	124	<null>	0	0
	2.576	18.679701	22.650506	0.72	2014-10-02	2020-09-09	<null>	<null>	1.212573	4552	0.373462	17	18
	<null>	20.985915	21.594203	0	2014-10-02	2020-09-09	<null>	<null>	1.028986	71	<null>	0	0

Demo - overall ad performance

data virtuality

Using the procedures

facebook: using the procedures

- Most important procedures: MarketingInsights, MarketingActions: use them over the get* procedures
- Interesting parameters:
  - target_table: result will be written to the table, if called again, will simply add incremental data
  - do not use end_date for loading incrementally
  - clean28days will remove and reload data in case it “changes in the past”

```
call "face_book_src.MarketingInsights"(
    "account_id" => long_account_id/* Optional: Ad Account ID */,
    "campaign_id" => long_campaign_id/* Optional: Ad Campaign ID (overrides Account ID) */,
    "start_date" => date_start_date/* Optional: Start Date */,
    "end_date" => date_end_date/* Optional: End Date */,
    "batchSize" => integer_batchSize/* Optional: Download data in batches, duration in days */,
    "level" => 'string_level'/* Optional: Level: Campaign, Adset or Ad */,
    "daily" => boolean_daily/* Optional: Daily precision or summary data */,
    "clean28days" => boolean_clean28days/* Optional: Remove 28 last days each time */,
    "breakdown_age" => boolean_breakdown_age/* Optional: Breakdown results by country */,
    "breakdown_country" => boolean_breakdown_country/* Optional: Breakdown results by country */,
    "breakdown_device_platform" => boolean_breakdown_device_platform/* Optional: Breakdown results by country */,
    "breakdown_gender" => boolean_breakdown_gender/* Optional: Breakdown results by country */,
    "breakdown_publisher_platform" => boolean_breakdown_publisher_platform/* Optional: Breakdown results by country */,
    "actionsOnly" => boolean_actionsOnly/* Optional: Only request actions, otherwise all ads action stats */,
    "action_attributes_mask" => 'string_action_attributes_mask'/* Optional: Override mask 101010 for 1d_click,1d_view,7d_click,7d_view,28d_click,28d_view */,
    "syncMode" => boolean_syncMode/* Optional: If true, data is requested synchronously */,
    "fieldsOverride" => 'string_fieldsOverride'/* Optional: Requests only the specified list of fields from the API */,
    "no_ads_action_stats" => boolean_no_ads_action_stats/* Optional: Do not request ad action stats, and only create the single table */,
    "customFields" => 'string_customFields'/* Optional: JSON object to define extra columns */,
    "target_table" => 'string_target_table'/* Optional: Table name to save the data to */,
    "preview" => boolean_preview/* Optional: Preview only, don't write into table */
);;
```

Demo - MarketingInsights Procedure

data virtuality

Creating custom requests

Custom requests

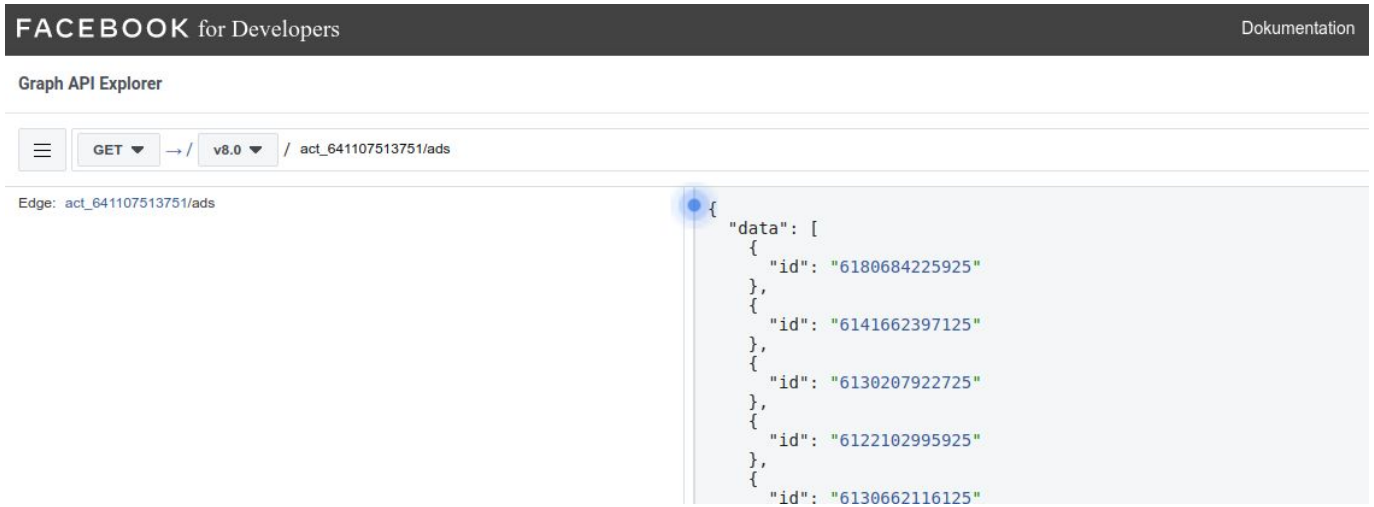
If you want to create a procedure for a custom request, use this procedure:

```
call "face_book_src.create_procedure_for_endpoint"(
    "in_request_url" => 'act_641107513751/ads'/* Optional: Request URL you get from the Graph API Explorer (omit
"https://graph.facebook.com/version/ */,

    "in_target_procedure_fully_qualified_name" => 'facebook_examples.ads'/* Optional: Fully qualified target
procedure name | The name of the procedure that will be created if should_create_procedure is set to true. Format:
schema_name.procedure_name */,

    "in_should_create_procedure" => false/* Optional: Should create procedure | If it should create the procedure
rather than output the script */,

    "in_async" => false/* Optional: Async | Whether the async API should be used */
);;
```



Custom Request - resulting procedure

```
/************************
* Auto-generated query
************************/
CREATE VIRTUAL PROCEDURE "facebook_examples.ads"()
RETURNS(
"id" STRING
) as begin

declare string next_url = 'act_641107513751%2Fads';
declare string current_url = '';
declare integer current_page = 1;
declare boolean done = false;

CREATE LOCAL TEMPORARY TABLE "#_LOCAL__FACEBOOK_ads"(
"id" STRING
, "curr_page" INTEGER
, "next_page" STRING );

WHILE (NOT done)
 begin
INSERT INTO "#_LOCAL__FACEBOOK_ads"
SELECT
"xmlTable.id"
, "current_page"
,"xmlTable.next_page"
        FROM XMLTABLE(XMLNAMESPACES( 'http://www.w3.org/2001/XMLSchema-instance' as "xsi" ), 'root/data'
                PASSING ( SELECT response FROM (EXEC "face_book_src.internal_doQuery"("action" => 'GET',"endpoint" => next_url)) AS r )  COLUMNS
 "id" STRING PATH 'id'
,"next_page" STRING PATH '../paging/next') "xmlTable"
;
current_url = next_url;
 next_url = SELECT A.next_page from "#_LOCAL__FACEBOOK_ads" A WHERE A.curr_page = current_page LIMIT 1;
IF( next_url IS NULL )
 done = true;
current_page = current_page + 1;
 end

SELECT "id"
 FROM "#_LOCAL__FACEBOOK_ads";
end;
```

Demo - Creating a custom request procedure + looking at the generator proc

data virtuality

Replicating the results

Replicating the results of the custom procedure

- Create a view around it:

```
create view facebook_examples.v_ads as
select * from
(call "facebook_examples.ads"()) a;;
```

- Create a replication job:
  - right click on the view
  - select create replication job
  - in our case, we want a full replication each time, so we can choose copyover
  - select the schedule
- If you want to do incremental load
  - let the procedure return the last 1+ days
  - use batch replication

Demo - Replicating the custom request data

Summary

- Modular connectors offer better managment features
- Use the Marketing* procedures of the facebook connector
- You can create your own procedures if we do not have the regarding template

Any feedback / questions?

Thank you!

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

or

visit us at:
datavirtuality.com