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

Synchronization Methods

What to expect from this session?

In this track, we will look at a use case to copy data from Salesforce to Zendesk as an example to synchronize data between datasources. The following topics will be covered:

- connecting to the data sources
- identifying what to read and what to write
- creating a procedure to read and write
- automating the process
- Bonus material - replication jobs

Use case statement

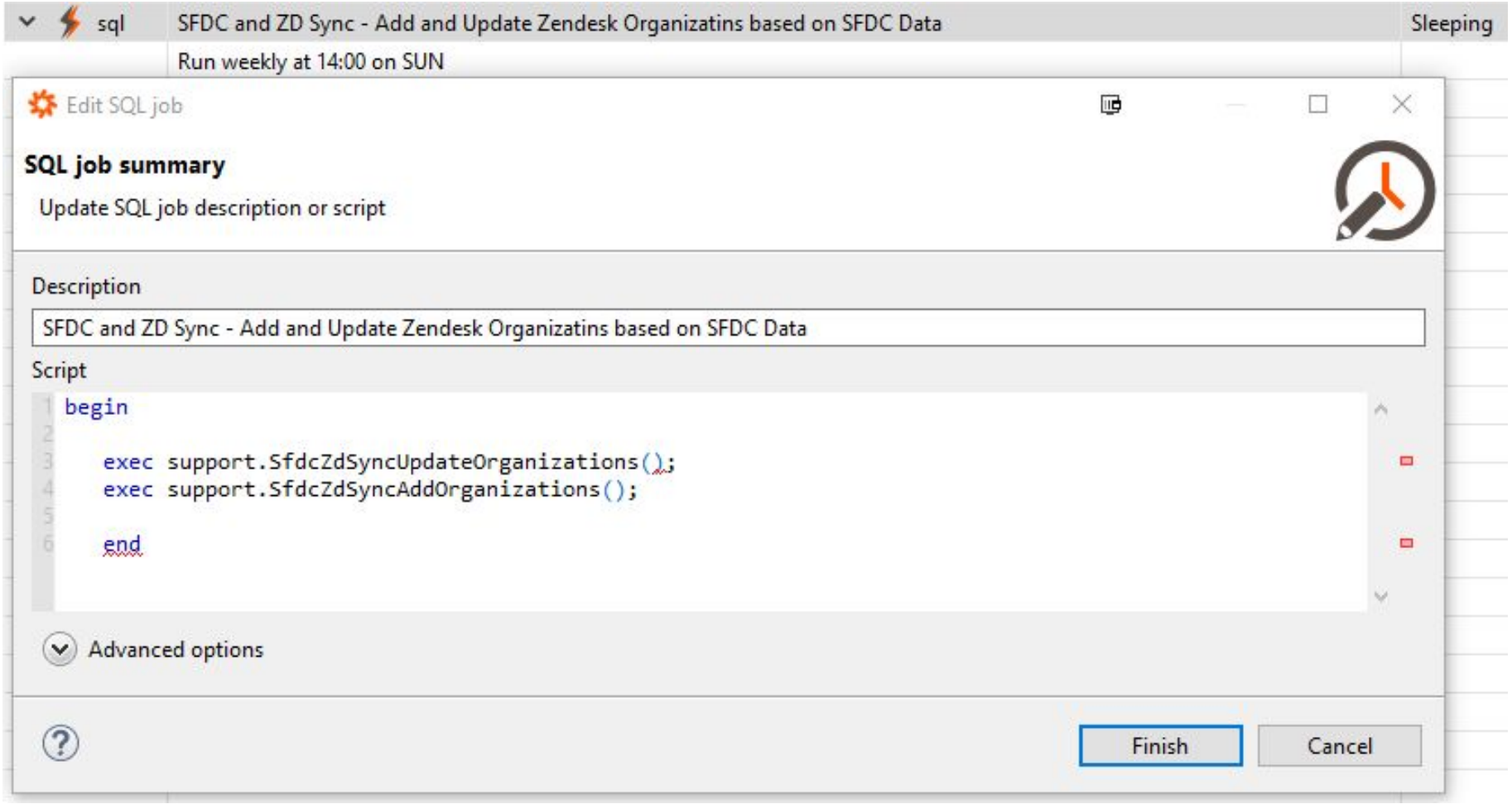
- Problem statement
  - Organizations are created by the Sales Team
  - In a POC, we will deliver support via email
  - If the prospect purchases, we would like the organization to be created in Zendesk, to deliver the support
- This is actually Enterprise Application Integration rather than reporting
- Zendesk SalesForce plugin could not be used for it

data virtuality

Reverse engineering the use case

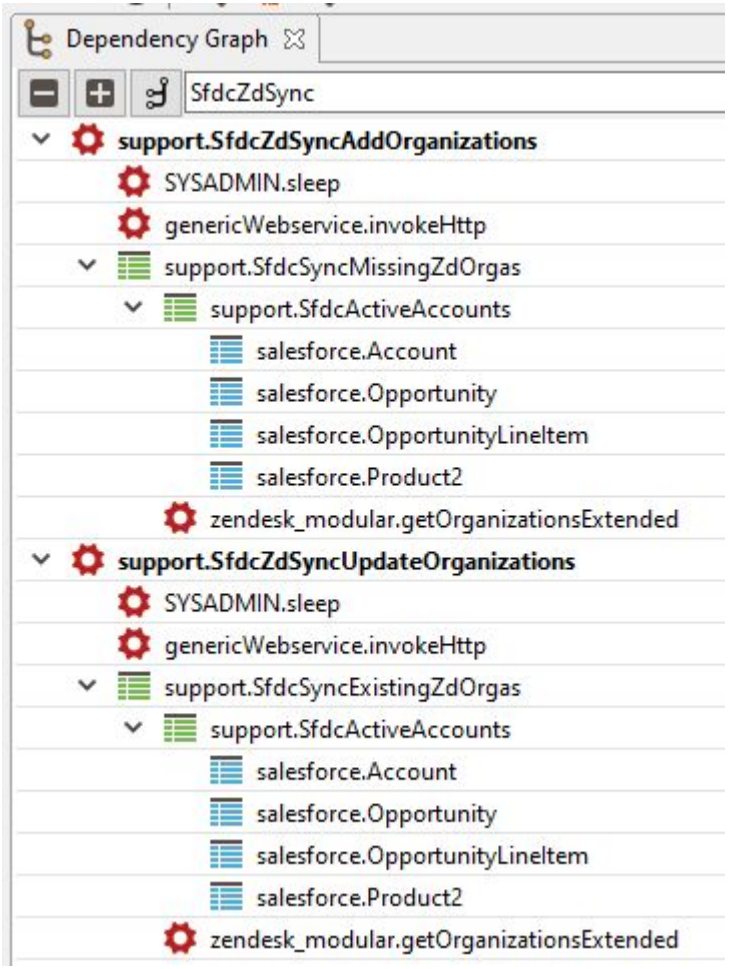
Reverse engineering the use case - part 1

- Starting with the job, I looked for the calls being made there



Reverse engineering the use case - part 2

- Using the Dependency Graph, I found all references being made
- I used DV's export functionality to move it to my server

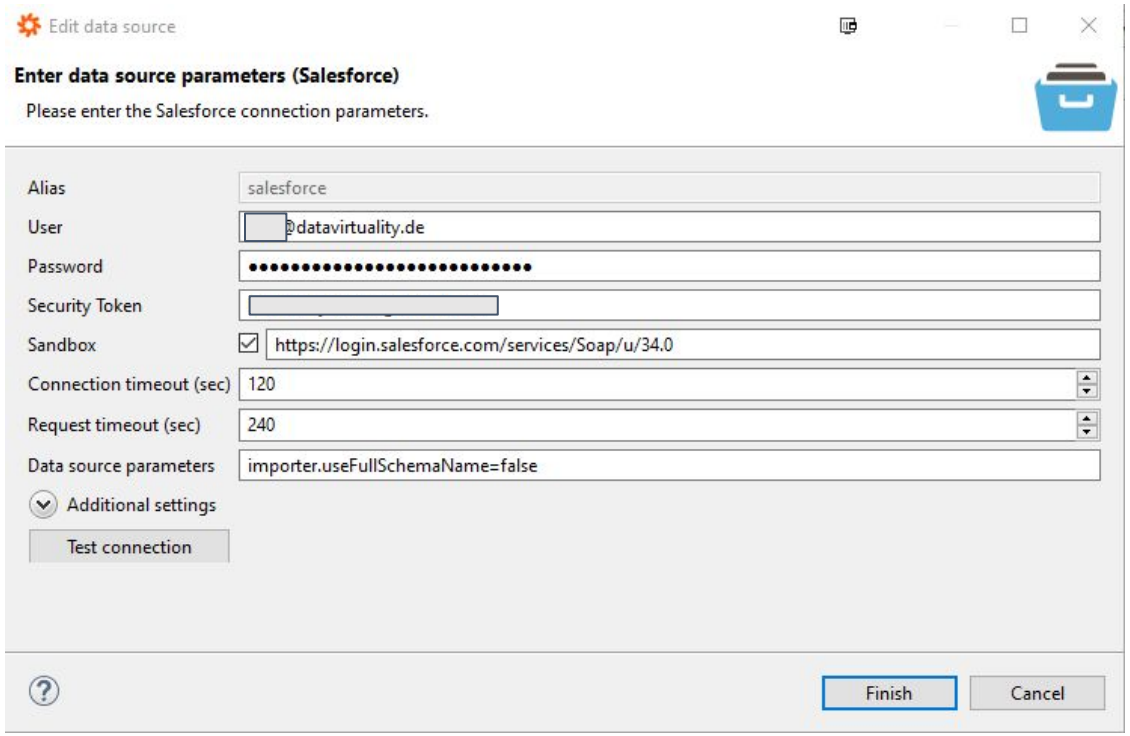


data virtuality

Data sources and affected data

Connecting the data sources

- Salesforce connection can be established via wizard



- Zendesk connection is done via modular connector

```
/* Create connection */
call SYSADMIN.createConnection(name => 'zendesk_modular', jbossCliTemplateName =>
'ws', connectionOrResourceAdapterProperties =>
'EndPoint=https://datavirtuality.zendesk.com/api/v2,decompressCompressedFiles=false
,SecurityType=HTTPBasic,AuthUserName=***@datavirtuality.de', encryptedProperties =>
'AuthPassword=***');;

/* Create data source */
call SYSADMIN.createDatasource(name => 'zendesk_modular', translator => 'zendesk',
modelProperties => 'cleanupMethod=DELETE', translatorProperties => '',
encryptedModelProperties => '', encryptedTranslatorProperties => '');;
```

Demo - Investigating the data sources

Identifying what to read and to write - overview

- We want to sync Organizations
- Let’s find the Organizations
  - existing Organizations in SF and ZD to update
  - missing Organizations that are in SF but not in ZD
- In order to be able to join, we filled the SalesForce IDs to a custom field in ZD
- We want to write the following information to Zendesk
  - Sync: SF ID, license expiration, licensed data sources, number of DV instances
  - Default values created: shared_tickets, shared_comments

Identifying existing organizations in both SF and ZD

data virtuality

```
CREATE view support.SfdcSyncExistingZdOrgas as select
        baseData.* ,instanceData.NumberOfInstances ,orga.id as organization_id
    from
        (
            select
                    AccountId,AccountName,min ("Start_of_Contract__c") as LicenseStart,max ("End_of_Contract__c") as
LicenseExpiration
                    ,string_agg (distinct "ProductName",'; ') as Products
                from
                    support.SfdcActiveAccounts
                where
                    ProductName not in ('Consulting','DataVirtuality Commercial','Discount','Instance Hosting'
,'Reseller Fee','Upfront Payment','Referral Partnership')
                group by
                    AccountId,AccountName) as baseData inner join (
            select
                    AccountId,AccountName,count (ProductCode) as NumberOfInstances from
                    support.SfdcActiveAccounts
                where
                    ProductCode in ('CONN-0024','CONN-0348') --0024 = commercial, 0348 = enterprise
                group by
                    AccountId
                    ,AccountName
                order by
                    AccountName
        ) as instanceData
        on baseData.AccountId = instanceData.AccountId inner join (
            exec "zendesk_modular.getOrganizationsExtended" ()) as orga
        on baseData.AccountId = orga.cf_id_Salesforce
```

Identifying missing organizations from ZD in SF

data virtuality

```
CREATE view support.SfdcSyncMissingZdOrgas
as
select baseData.*, instanceData.NumberOfInstances, orga.id as organization_id
from (select
                AccountId
                ,AccountName
                , min("Start_of_Contract__c") as LicenseStart
                , max("End_of_Contract__c") as LicenseExpiration
                , string_agg(distinct "ProductName",'; ') as Products
          from support.SfdcActiveAccounts
          where ProductName not in ('Consulting', 'DataVirtuality Commercial', 'Discount', 'Instance Hosting', 'Reseller
Fee', 'Upfront Payment', 'Referral Partnership')
          group by AccountId, AccountName) as baseData
     inner join
          (select    AccountId
                ,AccountName
                , count(ProductCode) as NumberOfInstances
          from support.SfdcActiveAccounts
          where ProductCode in ('CONN-0024', 'CONN-0348') --0024 = commercial, 0348 = enterprise
          group by AccountId, AccountName
          order by AccountName) as instanceData on baseData.AccountId = instanceData.AccountId
     left join (exec "zendesk_modular.getOrganizationsExtended"()) as orga on baseData.AccountId = orga.cf_id_Salesforce
where orga.cf_id_Salesforce is null
;
```

Demo - Identifying the customers and using string_agg

Creating a procedure to write the data

Creating a procedure to read and write - Update

```
create virtual procedure support.SfdcZdSyncUpdateOrganizations()
as
begin

declare string operation = 'PUT';
declare string endpoint = 'https://datavirtuality.zendesk.com/api/v2/organizations/';
declare string headers = 'Authorization: Basic ***';
declare string contenttype = 'application/json';
declare string request;

     loop on (select distinct AccountName, AccountId, LicenseExpiration, cast(Products as string) as Products,
NumberOfInstances, Organization_Id
                from  support.SfdcSyncExistingZdOrgas
                ) as cur
     begin
     request =
     '{
    "organization": {"name": "' || cur.AccountName || '"
                   , "shared_tickets": true
                   , "shared_comments": true
                   , "organization_fields":{"id_salesforce": "' || cur.AccountId || '"
                                          , "license_expiration": "' || cur.LicenseExpiration || 'T23:59:59Z"
                                          , "quellysteme": "' || cur.Products || '"
                                          , "number_of_instances": ' || cur.NumberOfInstances || '}}}';

     exec "genericWebservice.invokeHttp"("action" => operation,"endpoint" => endpoint || cur.Organization_Id || '.json',
       "requestHeaders" => headers,"requestContentType" => contenttype,"request" => request );

     exec "SYSADMIN.sleep"("millis" => 1000);
     end

end;
```

Creating a procedure to read and write - add organizations

```
create virtual procedure support.SfdcZdSyncAddOrganizations()
as
begin

declare string operation = 'POST';
declare string endpoint = 'https://datavirtuality.zendesk.com/api/v2/organizations.json';
declare string headers = 'Authorization: Basic ***=';
declare string contenttype = 'application/json';
declare string request;

     loop on (select distinct AccountName, AccountId, LicenseExpiration, cast(Products as string) as Products,
NumberOfInstances
                from  support.SfdcSyncMissingZdOrgas
                ) as cur
     begin
          request =
     '{
    "organization": {"name": "' || cur.AccountName || '"
                   , "shared_tickets": true
                   , "shared_comments": true
                   , "organization_fields":{"id_salesforce": "' || cur.AccountId || '"
                                          , "license_expiration": "' || cur.LicenseExpiration || 'T23:59:59Z"
                                          , "quellysteme": "' || cur.Products || '"
                                          , "number_of_instances": ' || cur.NumberOfInstances || '}}}';

     exec "genericWebservice.invokeHttp"("action" => operation,"endpoint" => endpoint,"requestHeaders" => headers,
       "requestContentType" => contenttype, "request" => request );

     exec "SYSADMIN.sleep"("millis" => 1000);
     end

end;
```

Automating the process and improvements

- A job was created that executes both procedures
- This is where the reverse engineering started
- Improvement: use internal_doQuery function instead of the generic webservice call

data virtuality

Other integration use cases

Other interesting integrations

- Google Analytics - help center views
- Jira + Zendesk integration
- Zendesk tickets for reporting, as the built in reporting goes back 90 days only and closed tickets are archived, Zendesk reporting in general
- Heartbeat of internal monitoring instance
- History update of changes in google sheets
- History of Salesforce changes
- History of Marketing automation system
- Salesforce Backup as csv, then moved to ftp
- Asana task reporting + Jira integration
- SalesForce reporting

data virtuality

Bonus material - Synchronizing data with replication jobs

Motivation to show replication types

- The synchronization we showed was very specific as we are writing to Zendesk
- Usually you would use replication jobs to write to a database
- Customers often find it hard to choose between the job types

data virtuality

Batch replication

Batch replication

- Appends the data to an existing table
- Use identity field to prevent duplicates
  - you could also use a date column
- Wizard forces you to use dwh table
  - but you can create a job on your own with a different database
  - importer.defaultSchema needs to be specified for the data destination

Demo - Batch replication

data virtuality

History Update Replication

History Update Replication

- Tracks changes in the selected fields
- Only the status at runtime is relevant for the changes
- If a tracked value changes, a new row will be added to the target table
    - you will have to consider this if you want to report on the target table
- Validity time frame column is added
- In the procedure call, you can use other destinations than dwh
- ID MUST be unique

Demo - History Update replication

data virtuality

Upsert Replication

Upsert Replication

- Provides Update and Insert
- Can be utilized for incremental replications
- No deletes are captured
    - as a workaround you can schedule a full reload

Demo - Upsert Replication

data virtuality

CopyOver Replication

Copy Over Replication

- Very simple solution to move data around
- Wizard supports other data destinations than DWH
- When the metadata of the source table changes, use DROP
- TRUNCATE vs DELETE
    - usually TRUNCATE is faster: no rollback, locks entire table instead of row, can’t use WHERE

Demo - CopyOver Replication

Summary

- DV can be used for more than reporting
- We hope you were inspired to create own integrations
- Having a clear understanding of the Replication Types will help in daily DV business

Any feedback / questions?

Thank you!

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

or

visit us at:
datavirtuality.com