SAP BW Semantically Partitioned Objects for Master Data Upload to SAP Demand Signal Management

A detailed step-by-step how-to guide
# TABLE OF CONTENTS

INTRODUCTION ............................................................................................................. 4  
SPOs: General Description .......................................................................................... 4  
Motivation ...................................................................................................................... 5  
PREREQUISITES ........................................................................................................... 6  
BUSINESS USE CASE ................................................................................................ 6  
APPROACH .................................................................................................................... 7  
IMPLEMENTATION ........................................................................................................ 7  
Partition Values for Master Data .................................................................................. 8  
Modeling ....................................................................................................................... 10  
SPOs ............................................................................................................................... 13  
Propagation layer ......................................................................................................... 21  
InfoSources and Transformations ............................................................................... 21  
Upload into InfoObjects ............................................................................................... 35  
InfoSources and Transformations and DTPs ............................................................... 38  
Process chains .............................................................................................................. 45  
Quality Validation ........................................................................................................ 50  
Overview ...................................................................................................................... 51  
Partition Values .......................................................................................................... 51  
DataSources .................................................................................................................. 52  
InfoSource & Transformations .................................................................................... 52  
SPO ................................................................................................................................. 52  
Stored procedures ........................................................................................................ 58  
Process Flow Control ................................................................................................... 63  
Customizing .................................................................................................................. 63  
Process definition ........................................................................................................ 63  
Data delivery agreement .............................................................................................. 65  
Data Set File ................................................................................................................... 67  
Steps Instantiation ....................................................................................................... 68  
ADDITIONAL LINKS .................................................................................................... 70
Applies to:
SAP Demand Signal Management (SAP DSiM) 2.0 SP01 and higher.

Summary:
This document describes the necessary steps, with all related technical details, on how to deploy SAP DSiM using semantically partitioned objects (SPOs) instead of regular DataStore objects (DSOs) to upload master data. The described prototype will cover both data upload scenarios: the pull scenario (that is by means of files) and the push scenario (that is by means of SAP Data Services).

Note:
1. Upload of transaction data for POS data is out of scope of this document (a link to the SAP Knowledge Warehouse where you can take a look at SPOs for transaction data in SAP DSiM, is provided under Additional links section at the end of this document)

2. SAP Data Services configuration will not be covered within this document and instead we will refer you to a dedicated SDN document that you can use for this purpose if needed

Author:
Amina Bounadja
amina.bounadja@sap.com
SAP Canada Inc.

Created: June 2014
INTRODUCTION

SAP Demand Signal Management (SAP DSiM) enables companies to upload, harmonize, enrich, and analyze large amounts of external demand data in order to improve supply chain, sales, marketing and other business processes. With SAP DSiM, customers can get deeper, faster, and more accurate insight on what is going on in the market (e.g. competitor or market share information), and also what happens directly at retailer/store level. This enables them to react faster to any changes in end-consumer demand.

SAP Demand Signal Management is a solution and Demand Data Foundation is a repository. It allows you to collect and integrate disparate data and store them in one consistent and centralized repository. This is in order to provide you with one data provider powered by analytics for consuming applications (e.g. SCM-APO, CRM-TPO).

The Demand Data Foundation repository is an in-memory database leveraged by the power of SAP HANA and, from a SAP BW perspective; it consists of dedicated DataStore Objects (DSOs) for each layer or processing step within the data flow.

Throughout this document, we will show that SAP DSiM can also be utilized and deployed using SPOs instead of regular DSOs. We will describe the different adjustments needed to achieve this, and provide all the technical details and steps on how to do it using the business use case as described below.

Before getting into the details, let’s have a quick look to what is an SPO? and explain why we use SPOs in SAP DSiM.

SPOs: General Description

Below are some key points that describe SPOs:

- New BW objects introduced in SAP BW 7.3
- InfoProviders that consist of several InfoCubes or DSOs with the same structure
- Semantic partitioning is a property of the InfoProvider. It divides the InfoProvider into several small, equally sized units (partitions) with the same structure.
- A SPO partition is characterized by the following partitioning criteria:
  - Only key fields are allowed for a DSO
  - Individual values and intervals can be used
  - Partitions must not overlap
Motivation
Here are some points to highlight the adding values when we use SPOs in SAP DSiM:

- **Have independent decoupled data uploads to:**
  - Allow speedier data upload through several uploads triggered in parallel and reduce the amount of time required to make data available for reporting
  - Provide better error handling and robustness of upload. Indeed, if a request of a given upload ends with an error, for example, the entire InfoProvider is unavailable for further uploads as well as for analysis and reporting. With a semantically partitioned object, the separation of the data into different partitions means that only the given partition that caused the error is unavailable.

- **Time zones support** (report availability). EDW scenarios usually involve several time zones. With a semantically partitioned object, the time zones can be separated by the partitions. Data loading and administrative tasks can therefore be scheduled independently of the time zone.

- **Allow higher data volume**, where needed, by eliminating the technical upper limits of 2 billion records per DSO.
PREREQUISITES

1. This document is not intended as a general presentation of SPOs. Therefore, we highly recommend you first get familiar with the SPO wizard. Some useful links for this purpose have been selected and gathered under the Additional links section at the end of this document.

2. A good understanding of the SAP DSiM data model and the SAP DSiM data upload process is also required because these topics are not explained in detail in this document. We will rather use SAP DSiM concepts assuming that you are already familiar with them.

BUSINESS USE CASE

Taste Inc. is a multinational consumer products company/manufacturer that produces chocolate and cake products and sells them through retailers. Taste Inc. has retailers worldwide in different regions across continents. Nevertheless, the most important ones are mainly located in South America (SA) and Europe (EU). This makes the volume of demand data, which Taste Inc. expects/receives, much higher from retailers of these two regions than the rest of the retailers which are located in regions other than Europe and SA.

Taste Inc. uses the standard SAP DSiM to upload, process, and analyze the point of sales demand data. To make SAP DSiM more suitable to Taste Inc.’s business context, we suggest deploying the solution using SPOs instead of regular DSOs. This allows for multiple data storage areas (partitions) with one for each region: “EU”, “SA” and use a separate partition (that we’ll call “Other”) for the other regions. With this set-up, Taste Inc. will ensure better balancing of data volume and can also reach a higher speed of data upload since several uploads can be triggered in parallel when using SPOs as oppose to regular DSOs.

\[\text{Note: Product and Location has each a dedicated SPO for each layer}\]
APPRAOCH

To implement the data flow above, we will make use of the context for data delivery agreement attributes (also called CDPA) and the object, newly introduced in SAP DSIM 2.0, called “partition values for master data” provided mainly to control data flow split when using SPOs for master data upload.

Below, we give a high-level overview of the approach before tackling the technical steps and details:

1. For each region we allocate a group of CDPAAs and later (in customizing) define the related data delivery agreement
2. Via dedicated Customizing, we:
   a. Define for each region one value for master data partitioning
   b. Assign the CDPAAs of each region to the one unique partition value for master data partitioning defined in point a. This is to ensure that all master data provided from countries of one given region go into one SPO partition
3. We make the partition value defined in point 2.a above available as a characteristic value of InfoObject /DDF/CTX_PART in SAP BW by executing the corresponding DTP.
4. We then use the characteristic /DDF/CTX_PART as a criterion for SPO partitioning in each layer and generate related DTPs and process chains.
5. We set up the SAP DSIM process flow control in such a way that at runtime for each region/CDPA of the handled data delivery agreement the accurate steps/process chains are instantiated/executed.

Note:
1 - The approach we describe above, and will detail in coming sections, is not the only way of realizing this business use case. There are other possible solutions that need to be evaluated in a customer project. Also, SPOs can be used differently and for other business use cases in SAP DSIM as well.

2 – In all coming sections we will refer to:
   a. Master data by the abbreviation MD
   b. Process flow control by the abbreviation PFC
   c. Data Delivery Agreement by the abbreviation DDAGR
   d. Partition value for master data by the abbreviation CTX_PART

IMPLEMENTATION

In this section, we will illustrate a step-by-step procedure to implement the above data flow design by leveraging DSiM & SPOs.

As highlighted in the data flow above, data partitioning will take place at each layer of SAP DSiM for the MD upload process. Hence, the following DSOs need to be replaced by SPOs:

<table>
<thead>
<tr>
<th>DSO</th>
<th>Technical name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>/DDF/DS21</td>
<td>Location - Acquisition</td>
</tr>
<tr>
<td></td>
<td>/DDF/DS31</td>
<td>Location - Propagation</td>
</tr>
<tr>
<td></td>
<td>/DDF/DS41</td>
<td>Product - Acquisition</td>
</tr>
<tr>
<td></td>
<td>/DDF/DS51</td>
<td>Product - Propagation</td>
</tr>
<tr>
<td></td>
<td>/DDF/DS61</td>
<td>Attribute Reference</td>
</tr>
</tbody>
</table>
Note:
1 - You can also partition (replace with SPOs) the DSOs below, by following the upcoming described steps if you want to support the name/value pair data upload:

<table>
<thead>
<tr>
<th>DSO</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/DDF/DS22</td>
<td>Location Name/Value Pairs - Acquisition</td>
</tr>
<tr>
<td>/DDF/DS32</td>
<td>Location Name/Value Pairs - Propagation</td>
</tr>
<tr>
<td>/DDF/DS43</td>
<td>Product Name/Value Pairs - Acquisition</td>
</tr>
<tr>
<td>/DDF/DS53</td>
<td>Product Name/Value Pairs - Propagation</td>
</tr>
</tbody>
</table>

2 - For your own specific use case, and hence implementation, we highly recommend you to take the time to thoroughly think about your data flow design and SPO partitioning before starting implementation. This is mainly because:

a. Any repartitioning of SPOs (and by repartitioning we mean changing data storage from one partition to another partition into the same SPO) after implementation would require additional “expensive” effort and create complexity that you can avoid beforehand. This is due to the fact that, for example, SPO partitions must not overlap and hence data previously loaded into one partition needs to be dropped before you can move/upload them into a new partition of the same SPO. However, adding brand new partitions, to upload brand new data, is of course not an issue and is fully supported by SPOs and by DSiM when using SPOs

b. Migration back to DSiM DSOs after SPOs usage, for instance, will also not be done for free and requires additional “expensive” effort and complexity that you will have to handle. As well, although this is a rare scenario and not recommended, we thought it would be good to be mentioned here to mainly highlight, that regardless of the combination (SPOs and DSOs) you want to use, it is strongly recommended to always make sure that data from one given source will always reside in one unique storage area. This will prevent you from having data redundancy and data (from the same source) spread over multiple staging areas. This will also help to reduce the complexity of your data flow and in this way reduce the implementation and maintenance effort.

Of course we don't foresee all the possible issues you might run into and hence don't aim to make a list of all of them, nevertheless we wanted to bring these aspects to your attention. Ideally, if possible, we suggest that you first try a “dry run” following what we describe in this document and after a successful implementation to adjust it to your own business use case. This will give you a good overview beforehand on how to design and proceed with your implementation and allow you to make more accurate decisions.

Partition Values for Master Data

Based on the data flow above and considering the DDAGR, we can create the following:

1. As many regions, and hence as many CTX_PART, as we want (99 is the maximum number of partitions that an SPO can have)
2. Several CDPAs for each region as many as we need
3. For each CDPA, we can define as many DDAGRs as we need

Nevertheless, to make our example concise and easy to understand, we chose to go with following set-up:

<table>
<thead>
<tr>
<th>Region</th>
<th>CDPA</th>
<th>Data Delivery Agreement</th>
<th>Partition Values for Master Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>SA</td>
<td>TASTE_SA_1</td>
<td>TASTE_BR</td>
<td>TASTE_SA</td>
</tr>
<tr>
<td></td>
<td>TASTE_SA_2</td>
<td>TASTE_AR</td>
<td>Taste Inc. SA - Argentina</td>
</tr>
<tr>
<td>EU</td>
<td>TASTE_EU_1</td>
<td>TASTE_SP</td>
<td>Taste Inc. EU - Spain</td>
</tr>
<tr>
<td></td>
<td>TASTE_OTR1</td>
<td>TASTE_BI</td>
<td>Taste Inc. AS - Bahrain</td>
</tr>
<tr>
<td>Other</td>
<td></td>
<td></td>
<td>TASTE_OTHER</td>
</tr>
</tbody>
</table>
Also, we will illustrate the complete SAP DSiM PFC Customizing needed, including all necessary details, under the appropriate section below; nevertheless, we first need to define different CTX_PART before SPOs modeling can start, as these are prerequisites since partitioning is based on them. Steps to be followed are highlighted below:

1. Create the Partition values

2. Execute DTP to make the created partition values available in BW
Modeling
In this section we will describe the SPO modeling and creation. As previously mentioned the assumption is that you are already familiar with SPOs as the section below is not intended as a “how to” for SPOs, but rather focuses on key points on how to scale MD DSOs in SAP DSiM with SPOs.

Note:
1 - In the coming sections we will describe how to proceed to implement the given business use case, when you aim to keep/reuse standard SAP DSiM business logic and hence SAP DSiM transformations and InfoSources. Where for some given (or all) branches in the data flow (targeting given SPO partitions) you would like to implement your own specific business logic, feel free to plug in your custom InfoSources/transformations where appropriate. You can also skip the SAP DSiM business logic and hence SAP DSiM transformations and InfoSources in order to use your own specific ones.

2 - If you plan to enhance the Product/Location data model with additional attributes, you have to consider that partitions of one SPO are identical and must have the exact same structure. This leaves you with the following two alternatives:

   a. Create more than one Product/Location SPO and dedicate each to a given Product/Location data model you want to have

   b. Keep one Product/Location SPO that encapsulates all the additional attributes you plan to have

3 - If you choose to have more than one Product/Location SPO for the reasons mentioned above or for any other reasons, it is important to evaluate in advance if you will use same InfoSources and transformations across the different Product/Location SPOs or have dedicated InfoSources and transformations per Product/Location SPO. This is necessary to be considered in advance for the following reasons:

   a. Using the same InfoSources and transformations across the different Product/Location SPOs leads to the fact that any changes made to these InfoSources and transformations for one data flow will impact and deactivate the rest of the data flows. As a consequence, all connections to the related SPOs will be deactivated and hence temporary unusable until you perform the necessary adjustments and activations.

   b. Having dedicated InfoSources and transformations per Product/Location SPO is obviously more effort than using the same. However this way ensures decoupled data flows to the different SPOs, and changes to one data flow won’t impact or temporarily disable the rest of the data flows. This also allows parallel/simultaneous development/adjustments to the different data flows when needed.

No recommendations are provided in regard to any points mentioned above. Indeed, each of them depends on your use case, project setup and the effort you can/want to invest. You can use what is mentioned above to evaluate the pros and cons and make the right decisions that fit your use case and project. It is strongly recommended to consider these aspects in your design phase before starting the implementation. This will ensure that you have the most accurate design and avoid any unnecessary effort.
**Acquisition Layer**

The DSOs in focus for this section are the ones listed below:

<table>
<thead>
<tr>
<th>DSO</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/DDF/DS21</td>
<td>Location - Acquisition</td>
</tr>
<tr>
<td>/DDF/DS41</td>
<td>Product - Acquisition</td>
</tr>
</tbody>
</table>

To avoid redundancy, we will describe a step-by-step “how to” only for the **product** and you can follow the same process for the location. Also, since we will cover both data upload scenarios (PUSH and PULL), we will highlight what needs to be done for each of them in a dedicated section. You can follow the steps for the scenario you want to implement and you can also follow both steps if you intend to implement both data upload scenarios.

**Data Upload - PULL**

**DataSources**

There are mainly 3 options on how to proceed here:

1. Use the standard DataSource with related standard InfoPackage
2. Use the standard DataSource with InfoPackage, as many as the number of partitions we have
3. Use several DataSources, as many as the number of partitions we have

To determine our preferred option for implementation, let’s evaluate pro and cons for each option:

<table>
<thead>
<tr>
<th>Option</th>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard DataSource with related standard InfoPackage (1)</td>
<td>Zero effort</td>
<td><strong>Full parallelization on acquisition layer NOT possible.</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Hence, if process A and B are running, upload in acquisition layer of process B can only start when upload in acquisition layer of process A is complete, even though both processes have distinct targets. This is due to the InfoPackage that is common to process A and B and that cannot run simultaneously for both processes.</td>
</tr>
<tr>
<td>Standard DataSource with several InfoPackages (2)</td>
<td><strong>Full parallelization on acquisition layer possible.</strong> Hence, process A and B can run simultaneously since each has a dedicated InfoPackage</td>
<td>Because the DTPs for the acquisition layer have <em>Delta</em> as extraction mode, requests of a given partition will always be handled by the system for an upload attempt into all SPO partitions. Of course, no upload takes place except into the dedicated target; nevertheless time for processing is consumed.</td>
</tr>
<tr>
<td>Several DataSources each with dedicated InfoPackages (3)</td>
<td><strong>Full parallelization on acquisition layer possible.</strong> Hence, process A and B can run simultaneously since each has a dedicated InfoPackage</td>
<td>Higher implementation effort than options 1 and 2</td>
</tr>
</tbody>
</table>
From the above we can see that the best option to adopt is option 2 and that’s the one we will implement following the steps below:

1 – For the DataSources below create as many InfoPackages as SPO partitions you have. In our case, this is 3 partitions:

For each new InfoPackage in the *Extraction* tab, enter the code below for the file detection routine:

```abap
* File detection routine to get Filename during PFC Run Time
DATA lo_bi_api TYPE REF TO /ddf/cl_adu_bi_api.
DATA lv_filename TYPE /ddf/filename.
DATA lr_cx_adu_bi TYPE REF TO /ddf/cx_adu_bi.
DATA ls_msg TYPE rspc_s_msg.
CREATE OBJECT lo_bi_api.
TRY.
  CALL METHOD lo_bi_api->get_filename_by_request
  EXPORTING
    iv_request_id = p_request
  RECEIVING
    rv_filename = lv_filename.
  p_filename = lv_filename.
  p_subrc = 0.
CATCH /ddf/cx_adu_bi INTO lr_cx_adu_bi.
  CALL METHOD lr_cx_adu_bi->convert
  CHANGING
    xs_msg = ls_msg.
    sy-msgno = ls_msg-msgno.
    sy-msgid = ls_msg-msgid.
    sy-msgty = ls_msg-msgty.
    sy-msgv1 = ls_msg-msgv1.
    sy-msgv2 = ls_msg-msgv2.
    sy-msgv3 = ls_msg-msgv3.
    sy-msgv4 = ls_msg-msgv4.
    p_subrc = 4.
  EXIT.
ENDTRY.
```
After repeating these steps for location as well, you will end up with 6 InfoPackages as shown below:

![Image showing InfoPackages]

**InfoSources and Transformations**

Since we are using the standard DataSource, we will use Standard InfoSources and transformations to bind to SPOs. Hence, no is effort required in this section.

**Data Upload – PUSH**

No effort is needed here for the acquisition layer as we will be using the standard DataSources for both product and location.

Also, as previously mentioned, SAP Data Services configuration will not be covered within this document. Nevertheless, under the Additional links section, at the end of this document, you can find a link to a dedicated SDN document that you can use for this purpose if needed.

**SPOs**

1. Copy the standard Product Acquisition DSO. Give it an appropriate name and description and most importantly, select the checkbox Semantically Partitioned highlighted below:

![Image showing Create DataStore Object with Semantically Partitioned highlighted]
We also suggest you create a dedicated InfoArea for your prototype, as we did and highlighted above, in order not to mix with standard SAP DSiM objects and to also have better and cleaner visibility on your model.

2. Create the partitions for the regions we set above where /DDF/CTX_PART is the characteristic criteria and using the partition values for master data previously created as shown below:

![Partition Table]

Activate to physically create the partitions.

3. Create transformation where the target is set per default and the source is the standard InfoSource IN for product acquisition layer named /DDF/MD_PROD_41_I (and for location it is /DDF/MD_LOC_21_I). This way we inherit and reuse the standard transformation business logic from source to acquisition layer.

4. Create the DTPs, but first create a DTP template which contains the parameter configuration that corresponds to the settings that can be configured in a DTP. These settings are applied when DTPs are generated and they are the same as the standard SAP DSiM DTPs from DataSource to acquisition DSO.
Below we show how the DTP template should look like for each data upload scenario:

**PULL data upload scenario:**

- DTP Template: `ZSPO_TASTE_INC_MD_ACO_2_DS`
- Extraction Mode: Full
- Package Size: Only retrieve last request

**PUSH data upload scenario:**

- DTP Template: `ZSPO_TASTE_INC_MD_ACO_FILE`
- Extraction Mode: Delta
- Package Size: Only retrieve last request
To generate the DTPs, drag and drop the template on the related DataSource for each scenario as highlighted below:

**Note:**
For the PUSH scenario adjust the DTP filter to add the business logic below for the delivery ID:

```
* Business logic for Delivery

call method /ddf/cl_bw_dtp_filter=>filter_by_known_delivery

exporting
  ir_request = i_r_request
  i_fieldnm = i_fieldnm

importing
  e_subrc = p_subrc

changing
  ct_range = l_t_range[].
```
5. We have now reached the last step and it consists of generating the process chains: one for each DTP as shown below:

*Note:* This part requires focus and concentration. This is to make sure you drop the right DTP for the right process chain. Otherwise, using the wrong DTP for a given partition or for a given scenario will negatively affect your data upload later on and create errors that might require additional effort to detect and correct.

*Note:* It is technically possible and feasible to create one process chain to include all DTPs (for each scenario). However, to allow independent decoupled data uploads, and benefit from all the added values of SPOs we mentioned above, we highly recommend creating one process chain for each DTP. This way you ensure that there is no overlap between the different process chains and that they, therefore, can be executed in parallel. This will allow PFC to trigger uploads in all partitions in parallel, whereas with one process chain an upload for a given partition can only be triggered if no other partition/upload is already making use of the process chain.

At this stage, we are finished with the acquisition layer and now have the related SPOs. The resulting data flow should look like the following:
Note:
In the data flow above we see data sources for both data upload scenarios because we implemented both. In your case, you will of course see only the one you will implement or both if you also implement both.

6. Adjust the process chains, generated via the SPO wizard, to add:
   a. The accurate InfoPackage
   b. The SAP DSiM process type *Update Step Status*, with both variants/status values: 0070 (Status Complete) and 1400 (Status Error - Data delivery agreement Level (Subsequent Steps)):
**Hint:**

1. The process chains generated via the SPO wizard are all located under the display component shown below:

   ![Diagram](image)

   1. Unassigned Nodes
   2. Generated By HybridProvider
   3. Generated By Semantically Partitioned Object
   4. NODESNOTCONNECTED
   5. HYBR
   6. LPOA

2. We suggest you create new process chains to include/wrap the InfoPackage, the process chain generated by the SPO wizard and the SAP DSIM process type "Update step status".

The completed process chain should look like the one shown below for PULL data upload scenario:

![Diagram](image)

And like the one shown below for PUSH data upload scenario:
3. For a clean set up and better visibility, we recommend that you create your own display components and move all your process chains there. An example below shows what ours look like:

**Process chains for Acquisition layer**

```
> Demand Data Foundation
  > Master Data
    >> Taste_Inc_SPOs_for_MD_PC
      >>> Taste_Inc_SPOs_for_MD_PC_DS
        ◆ Z_SP21_LOC_DS_EU
        ◆ Z_SP21_LOC_DS_OTH
        ◆ Z_SP21_LOC_DS_SA
        ◆ Z_SP41_PROD_DS_EU
        ◆ Z_SP41_PROD_DS_OTH
        ◆ Z_SP41_PROD_DS_SA
    >> Taste_Inc_SPOs_for_MD_PC_FILE
      ◆ Z_SP21_LOC_FILE_EU
      ◆ Z_SP21_LOC_FILE_OTH
      ◆ Z_SP21_LOC_FILE_SA
      ◆ Z_SP41_PROD_FILE_EU
      ◆ Z_SP41_PROD_FILE_OTH
      ◆ Z_SP41_PROD_FILE_SA
```

**PUSH data upload scenario**

**PULL data upload scenario**

**Sub process chains generated via the SPO wizard**
Propagation layer
The DSOs in focus for this section are the ones listed below:

<table>
<thead>
<tr>
<th>DSO</th>
<th>Technical name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/DDF/DS31</td>
<td>Location</td>
<td>- Propagation</td>
</tr>
<tr>
<td>/DDF/DS51</td>
<td>Product</td>
<td>- Propagation</td>
</tr>
<tr>
<td>/DDF/DS61</td>
<td>Attribute</td>
<td>Reference</td>
</tr>
</tbody>
</table>

InfoSources and Transformations
To reuse standard SAP DSIM transformations we will connect each SPOs partitions to the IN InfoSource for the standard propagation DSO. To do so we proceed as follows:

1. Create the OUT InfoSource for the SPO acquisition layer as shown below:

![Image showing InfoSource creation process]

- **InfoSource**: Z_SPO_LOCATION_AQUISITION_OUT
- **Long Description**: Z_SPO_LOCATION_AQUISITION_OUT
- **Application Component**: Taste_SPO_Unknown

   **Copy from**
   - **Object Type**: DataStore Object
   - **Name**: Z_SP4100
2. Create necessary transformations for each InfoSource and SPO partition as shown below:

3. Create transformation between the SPO acquisition OUT InfoSource and the standard acquisition OUT InfoSource as highlighted below:
Resulting transformations are as shown below:

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Source</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Product Acquisition (Out)</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DDSO /DDF/DS41 -&gt; TRCS /DDF/MD_PROD_41</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>TRCS Z_SPO_PRODUCT_AQUISITION_OUT -&gt; TRCS /DDF/MD_PROD_41</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Z_SPO_PRODUCT_AQUISITION_OUT</td>
<td></td>
<td>Z_SPO_PRODUCT_AQUISITION_OUT</td>
</tr>
<tr>
<td><strong>Location Acquisition (Out)</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DDSO /DDF/DS21 -&gt; TRCS /DDF/MD_LOC_21_O</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>TRCS Z_SPO_LOCATION_AQUISITION_OUT -&gt; TRCS /DDF/MD_LOC_21</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Z_SPO_LOCATION_AQUISITION_OUT</td>
<td></td>
<td>Z_SPO_LOCATION</td>
</tr>
</tbody>
</table>
The following shows how the data flow, from acquisition SPOs partitions up to InfoSource IN for the propagation SPOs, will look like after repeating these steps for both product and location:

At this point we have all necessary BW objects to create the SPOs for the propagation layer.
SPOs

Note:
For this part there will be no distinction between PULL and PUSH data upload scenarios. Data upload from acquisition layer to propagation layer is similar to both scenarios.

To avoid redundancy, we will describe a step by step “how to” only for the product and you can follow the same process for the location. For the Attribute Reference DSO we will have a dedicated section since the data flow is different compared to product and location:

Product and Location DSOs

1. Copy the standard Propagation Product DSO. Give it an appropriate name and description and most important check the Semantically Partitioned flag highlighted below:

   ![Create DataStore Object](image)

2. Create the partitions for the regions we set above where /DDF/CTX_PART is the characteristic criteria and using the Partition values for master data previously created as shown below:

   ![Create DataStore Object](image)

   Activate to physically create the partitions.
3. Create and activate transformation where the target is set per default and the source is the IN InfoSource for the standard propagation DSO (/DDF/MD_PROD_51_I for product and /DDF/MD_LOC_31_I for location). This way we inherit and reuse the standard transformation business logic to the propagation layer as previously mentioned.

4. Create the DTPs, but first create a DTP template which is the parameters configuration that corresponds to the settings that can be configured in a DTP. These settings are applied when DTPs are generated and they are the same as the standard SAP DSiM DTP from Acquisition DSO to Propagation DSO:

![DTP Generation for Z_SP51](image)

**Note:**

It is highlighted in the screenshot above that the data source in the DTP template for the propagation layer is set as a DataStore object (instead of a Semantically Partitioned InfoProvider). This is to allow creation of transformations of type ODSO (instead of LPOA). Technically both types of transformation work. Below a small table that highlights the key pros and cons for each type to help clarify why for our prototype we have chosen transformations of type ODSO:
<table>
<thead>
<tr>
<th>Transformation type</th>
<th>Pro</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>ODSO</td>
<td><strong>Higher parallelization of the data uploads.</strong> The transformation source object holds the SPO partition. This enables PFC to identify in runtime the exact partition that is used and to allow access (in parallel) to the free ones if/when requested.</td>
<td>Several transformations are created. For each source and target partitions combination one transformation is created.</td>
</tr>
<tr>
<td>LPOA</td>
<td>One unique transformation is created for each target. It will link all source partitions (the SPO itself) to the given target.</td>
<td><strong>Coupled data uploads and NO parallelization possible.</strong> The transformation source object holds the SPO itself. This makes it not possible so far for PFC to identify in runtime the exact partition that is used. To avoid concurrent accesses to the same partition, PFC then allows only one access at a time to the given SPO.</td>
</tr>
</tbody>
</table>

To generate the DTPs drag and drop the template on the SPO partitions as highlighted below:

```plaintext
* Business logic for Delivery
call method /ddf/cl_bw_dtp_filter->filter_by_known_delivery
exporting
  ir_request = i_r_request
  i_fieldnm = i_fieldnm
importing
  e_subrc = p_subrc
changing
  ct_range = l_t_range[].
```
6. We have now reached the last step and it consists of generating the process chains: one for each DTP as shown below:

**Note:**
It is technically possible and feasible to create one process chain to include all DTPs (for each scenario). However, to allow independent decoupled data uploads, and benefit from all the added values of SPOs we mentioned above, we highly recommend creating one process chain for each DTP. This way you ensure that there is no overlap between the different process chains and that they, therefore, can be executed in parallel. This will allow PFC to trigger uploads in all partitions in parallel, whereas with one process chain an upload for a given partition can only be triggered if no other partition/upload is already making use of the process chain.
At this stage, we are finished with the propagation layer and have the related SPOs. The resulting data flow should look like the following:

**Data Flow Display for Semantically Partitioned (From DataSource to Propagation layer for Product&Location)**

- **Product Propagation (IN)**
- **Product Acquisition (OUT)**

**Standard InfoSources to reuse the main standard transformation from Acquisition to Propagation layer**

**Z DataSources with dummy transformations to connect Acquisition layer partitions to standard propagation DataSources**

**Standard InfoSources to reuse the main standard transformation from DataSources to Acquisition layer**
**Note:**
Proceeding in the same way as we did for the Acquisition layer, the next step here would be to adjust the process chains for the propagation layer previously generated via the SPO wizard. However, since these process chains also include upload into InfoObjects and DS61, we will tackle them later on as explained in the upcoming section.

**Attribute Reference DSO (DS61)**

When we check the data flow for DS61 we can see that:

1. DS61 gets data from acquisition product DSO, and that
2. The main standard transformation to it has the Product Acquisition (OUT) InfoSource as source as shown below:

Therefore, we will need to connect product acquisition SPO partitions to Product Acquisition (OUT) InfoSource to inherit/reuse the standard SAP DSiM transformation to DS61. However, this has already been done as described above under point 3 for chapter “InfoSources and Transformations” under the propagation layer and shown below:

Hence, no effort is needed here and we can start to create the Attribute Reference SPO following the steps below:
1. Copy the standard Product acquisition DSO. Give it an appropriate name and description and most important check the *Semantically Partitioned* flag highlighted below:

![Create DataStore Object](image)

2. Create the partitions for the regions we set above where `/DDF/CTX_PART` is the characteristic criteria and using the Partition values for master data previously created, as shown below:

![Create DataStore Object](image)

Activate in order to physically create the partitions.

3. Create transformation where the target is set per default and the source is the IN InfoSource for DS61 `/DDF/MD_ATTR_61_I`. This way we inherit and reuse standard transformation business logic as previously mentioned.

4. Create the DTPs, but first create a DTP template which contains the parameter configuration that corresponds to the settings that can be configured in a DTP. These settings are applied when DTPs are generated and they are the same as the standard SAP DSiM DTPs from Acquisition product DSO to DS61. These settings are similar to template `ZSPO_TASTE_INC_MD_PROPG` we previously created so you can reuse it to save effort. To generate the DTPs drag and drop the template on the related DataSource for each scenario as highlighted below:
5. Adjust the DTPs filter, the same as for the propagation layer DTPs, to add the business logic below for Delivery ID, the same as in the SAP DSiM standard DTPs for propagation layer

```
* Business logic for Delivery

call method /ddf/cl_bw_dtp_filter=>filter_by_known_delivery
  exporting
    ir_request = i_r_request
    i_fieldnm  = i_fieldnm
  importing
    e_subrc   = p_subrc
  changing
    ct_range  = l_t_range[].
```
5. We have now reached the last step and it consists of generating the process chains: one for each DTP as shown below.

**Note:**
This part requires focus and concentration. This is to make sure you drop the right DTP for the right process chain. Otherwise, using the wrong DTP for a given partition or for a given scenario will negatively affect your data upload later on and create errors that might require additional effort to detect and correct.
At this stage, we are finished with the Attribute Reference SPO; the related data flow should look like the following:
**Upload into InfoObjects**

The InfoObjects in focus for this section are the ones mentioned below:

<table>
<thead>
<tr>
<th>Related to</th>
<th>InfoObject</th>
<th>Source of data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product</td>
<td>/DDF/MANUFACTL</td>
<td>Source Manufacturer ID</td>
</tr>
<tr>
<td></td>
<td>/DDF/MATL_GRPL</td>
<td>Source Material Group ID</td>
</tr>
<tr>
<td></td>
<td>/DDF/PRODSCATL</td>
<td>Source Subcategory ID</td>
</tr>
<tr>
<td></td>
<td>/DDF/PRODCATL</td>
<td>Source Category ID</td>
</tr>
<tr>
<td></td>
<td>/DDF/BRANDL</td>
<td>Source Brand ID</td>
</tr>
<tr>
<td></td>
<td>/DDF/PRODREF</td>
<td>Source Product</td>
</tr>
<tr>
<td></td>
<td>/DDF/PRODUCT</td>
<td>Internal Source Product</td>
</tr>
<tr>
<td>Location</td>
<td>/DDF/LOCREF</td>
<td>Source Location</td>
</tr>
<tr>
<td></td>
<td>/DDF/LOCATION</td>
<td>Internal Source Location</td>
</tr>
</tbody>
</table>

**InfoSources and Transformations and DTPs**

We can divide the InfoObjects above into 3 categories based on the source of data and for each category describe what needs to be done as follows:

1. **InfoObjects reading from acquisition layer (DS21 and DS41)**

   If you check InfoObject /DDF/PRODREF, and /DDF/LOCREF, you will see they are already connected to product acquisition SPO, and respectively to location acquisition SPO, as shown below:

   - **Source Product**
     - **Product Acquisition (Out)**
       - ODSO /DDF/DS41 -> TRCS /DDF/MD_PROD_41
       - TRCS Z SPO PRODUCT ACQUISITION OUT -> TRCS /DDF/MD_PROD_41
       - Z_SPO_PRODUCT_AQUISITION_OUT -> /DDF/MD_PROD_41
   - **Location Acquisition (Out)**
     - ODSO /DDF/DS21 -> TRCS /DDF/MD_LOC_21
     - TRCS Z SPO LOCATION ACQUISITION OUT -> TRCS /DDF/MD_LOC_21
     - Z_SPO_LOCATION_AQUISITION_OUT

   This is due to connection we previously did and described above under point 3 in the chapter “InfoSources and Transformations” for the propagation layer. Hence no effort is needed here, in terms of InfoSource and Transformations, and we can now create the needed DTPs as shown below:
Note:
To avoid redundancy, we will describe a step-by-step “how to” only for the /DDF/PRODREF and you can follow the same process for the /DDF/LOCREF InfoObject.

a. Select /DDF/PRODREF and create a DTP for Attributes upload and a DTP for Text upload for all SP41 partitions as shown below:

b. Set DTP extraction mode as Full and add, for more clarity, “Attribute” to the description for DTP attribute upload and “Text” for the text upload DTP.
**Hint:**
To facilitate identifying later on which DTP is related to which SPO partition, we also suggest adding a suitable abbreviation to the DTP description. In our case, this would be as shown below:

<table>
<thead>
<tr>
<th>Data Transfer Process</th>
<th>Attribute (Z_SP4101 -&gt; /DDF/PRODREF)/SA</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>DTP_017G8W19698BYG1NH7D4D1XWN</td>
</tr>
<tr>
<td>DTP Type</td>
<td>Standard (Can Be Scheduled)</td>
</tr>
<tr>
<td>Version</td>
<td>Active</td>
</tr>
<tr>
<td></td>
<td>Saved</td>
</tr>
</tbody>
</table>

This will ease the process of building the process chains for each partition.

c. Adjust the filter on “Delivery” using the code below in the related routine:

```plaintext
* Business logic for Delivery

call method /ddf/cl_bw_dtp_filter=>filter_by_known_delivery
  exporting
    ir_request = i_r_request
    i_fieldnm = i_fieldnm
  importing
    e_subrc = p_subrc
  changing
    ct_range = l_t_range[].
```

After repeating the steps for **both InfoObjects**, with all **partitions** and for **Attribute and Text upload**, you end up with the DTPs shown below:
2. InfoObjects reading from propagation layer (DS31 and DS51)

For this section we need to create a connection/transformation between the InfoObjects and related Propagation SPOs partitions. To do so we follow the steps below:

a. The first step here is to create an InfoSource OUT for the propagations SPOs partitions as shown below:

![InfoSource creation for Propagation OUT](image-url)
b. Create necessary transformations for each InfoSource and SPO partition as shown below:

- **Z_SPO_LOCATION_AQUISITION_OUT**
  - **Z_SPO_LOCATION_PROPAGATION_OUT**
    - ODSO Z_SP3101 -> TRCS Z_SPO_LOCATION_PROPAGATION_OUT
    - ODSO Z_SP3102 -> TRCS Z_SPO_LOCATION_PROPAGATION_OUT
    - ODSO Z_SP3103 -> TRCS Z_SPO_LOCATION_PROPAGATION_OUT

- **Z_SPO_PRODUCT_AQUISITION_OUT**
  - **Z_SPO_PRODUCT_PROPAGATION_OUT**
    - ODSO Z_SP5101 -> TRCS Z_SPO_PRODUCT_PROPAGATION_OUT
    - ODSO Z_SP5102 -> TRCS Z_SPO_PRODUCT_PROPAGATION_OUT
    - ODSO Z_SP5103 -> TRCS Z_SPO_PRODUCT_PROPAGATION_OUT


c. Create transformation between the SPO propagation OUT InfoSource and the standard propagation OUT InfoSource (/DDF/MD_PROD_51 for product and /DDF/MD_LOCATION_31 for location) as highlighted below:

- **Target of the Transformation**
  - **Object Type**: InfoSource
  - **Name**: /DDF/MD_PROD_51

- **Source of the Transformation**
  - **Object Type**: InfoSource
  - **Name**: /DDF/MD_LOCATION_31
Resulting transformations should look as shown below where connection between InfoObjects and SPO partitions is highlighted:

### Product SPO partitions connection to /DDF/PRODUCT

- **Internal Source Product**
  - Internal Source Product (Attributes)
    - TRCS /DDF/MD_PROD_51 -> OBJ/PRODUCT
    - TRCS /DDF/MD_PROD_51 -> OBJ/PRODUCT
  - Data Transfer Processes
    - Product Propagation (Out)
      - ODSS /DDF/SDS1 -> TRCS /DDF/MD_PROD_51
      - TRCS Z_SPO_PRODUCT_PROPAGATION_OUT -> TRCS /DDF/MD_PROD_51
    - Z_SPO_PRODUCT_PROPAGATION_OUT

### Location SPO partitions connection to /DDF/LOCATION

- **Internal Source Location**
  - Internal Source Location (Attributes)
    - TRCS /DDF/MDLOCATION_31 -> OBJ/LOCATION
  - Location Propagation (Out)
    - ODSS /DDF/SDS1 -> TRCS /DDF/MDLOCATION_31
    - TRCS Z_SPO_LOCATION_PROPAGATION_OUT -> TRCS /DDF/MDLOCATION_31
  - Z_SPO_LOCATION_PROPAGATION_OUT

---

d. Create necessary DTPs, to upload data from SP51 partitions (and from SP31 partitions) into /DDF/PRODUCT (and into /DDF/LOCATION), proceeding in the same way as we described above for InfoObjects reading from acquisition layer (DS21 and DS41). After repeating the steps for both InfoObjects, with all partitions and for Attribute and Text upload, you end up with the DTPs shown below:
3. InfoObjects reading from DS61

a. The first step here is to create an InfoSource OUT for Z_SP61 as shown below:

b. Create necessary transformations for each SPO partition under the created InfoSource as shown below:

c. Create a transformation between the Z_SP61 OUT InfoSource and each of the InfoObjects. To do so, copy the standard SAP DSIM transformations linking the InfoObjects to DS61 as highlighted below using /DDF/MANUFACTL as an example:
The settings to copy would look as shown below:

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>/ODSO/DDF/DS61</td>
<td>/ddf/manufactrl</td>
</tr>
<tr>
<td>Texts</td>
<td>Texts</td>
</tr>
</tbody>
</table>

**Copy**
- **Transformation**: ODSO /DDF/DS61 -> IOBJ /DDF/MANUFACTRL

**Target of the Transformation**
- **Object Type**: InfoObject
- **Subtype**: Texts
- **Name**: /DDF/MANUFACTRL

**Source of the Transformation**
- **Object Type**: InfoSource
- **Name**: Z_SP061_OUT
- **With Target Object As Template**
After repeating this step for all InfoObjects of this category you should have the transformations shown below:

<table>
<thead>
<tr>
<th>Source</th>
<th>Text</th>
<th>Transformation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brand ID</td>
<td>/DDF/BRANGL</td>
<td>Z_SPO61_OUT</td>
</tr>
<tr>
<td>Category ID</td>
<td>/DDF/PRODCATL</td>
<td>Z_SPO61_OUT</td>
</tr>
<tr>
<td>Material Group ID</td>
<td>/DDF/MATL_GRPL</td>
<td>Z_SPO61_OUT</td>
</tr>
</tbody>
</table>
d. Create the necessary DTPs, to upload data from Z_SP61 partitions into dedicated InfoObjects proceeding in the same way as described above for InfoObjects reading from acquisition layer (DS21 and DS41). After repeating the steps for all InfoObjects, with all partitions you should now have DTPs similar to what is shown below:

We now have all the necessary elements to create and build the process chains to upload data into the propagation layer (SPOs partitions) and hence into the dedicated InfoObjects.
Process chains

The goal in this section is to provide process chains similar to the standard DSiM ones (/DDF/PROD_STAGE and /DDF/LOC_STAGE) to upload master data into the right SPOs partitions and InfoObjects. Therefore we will first take a look at these process chains to get an idea of how they are designed.

Product

In the screenshot below we highlight the structure of process chain /DDF/PROD_STAGE and its related sub process chains:

Following the structure shown above, we will need to create the process chains listed below:

<table>
<thead>
<tr>
<th>SAP DSiM sub process chain</th>
<th>Custom process chain for each partition</th>
</tr>
</thead>
<tbody>
<tr>
<td>/DDF/PROD2</td>
<td>ZTASTINC_PROD2_SA</td>
</tr>
<tr>
<td></td>
<td>ZTASTINC_PROD2_EU</td>
</tr>
<tr>
<td></td>
<td>ZTASTINC_PROD2_OTH</td>
</tr>
<tr>
<td>/DDF/PROD3</td>
<td>ZTASTINC_PROD3_SA</td>
</tr>
<tr>
<td></td>
<td>ZTASTINC_PROD3_EU</td>
</tr>
<tr>
<td></td>
<td>ZTASTINC_PROD3_OTH</td>
</tr>
<tr>
<td>/DDF/PROD4</td>
<td>ZTASTINC_PROD4_SA</td>
</tr>
<tr>
<td></td>
<td>ZTASTINC_PROD4_EU</td>
</tr>
<tr>
<td></td>
<td>ZTASTINC_PROD4_OTH</td>
</tr>
<tr>
<td>/DDF/PROD15</td>
<td>ZTASTINC_PROD15_SA</td>
</tr>
<tr>
<td></td>
<td>ZTASTINC_PROD15_EU</td>
</tr>
<tr>
<td></td>
<td>ZTASTINC_PROD15_OTH</td>
</tr>
<tr>
<td>/DDF/PROD_STAGE</td>
<td>ZTASTINC_PROD_STAGE_SA</td>
</tr>
<tr>
<td></td>
<td>ZTASTINC_PROD_STAGE_EU</td>
</tr>
<tr>
<td></td>
<td>ZTASTINC_PROD_STAGE_OTH</td>
</tr>
</tbody>
</table>
Below we show screenshots of how each process chain for region SA should look like and you can follow the same approach to build the process chains for the rest of the regions:
**Location**

Same as we previously did for Product, in the screenshot below we highlight the structure of process chain `/DDF/LOC_STAGE` and related sub process chains:

Following the structure shown above, we will need to create the process chains listed below:

<table>
<thead>
<tr>
<th>SAP DSiM sub process chain</th>
<th>Custom process chain for each partition</th>
</tr>
</thead>
<tbody>
<tr>
<td>/DDF/LOC2</td>
<td>ZTASTINC_LOC2_SA</td>
</tr>
<tr>
<td></td>
<td>ZTASTINC_LOC2_EU</td>
</tr>
<tr>
<td></td>
<td>ZTASTINC_LOC2_OTH</td>
</tr>
<tr>
<td>/DDF/LOC3</td>
<td>ZTASTINC_LOC3_SA</td>
</tr>
<tr>
<td></td>
<td>ZTASTINC_LOC3_EU</td>
</tr>
<tr>
<td></td>
<td>ZTASTINC_LOC3_OTH</td>
</tr>
<tr>
<td>/DDF/LOC4</td>
<td>ZTASTINC_LOC4_SA</td>
</tr>
<tr>
<td></td>
<td>ZTASTINC_LOC4_EU</td>
</tr>
<tr>
<td></td>
<td>ZTASTINC_LOC4_OTH</td>
</tr>
<tr>
<td>/DDF/LOC_STAGE</td>
<td>ZTASTINC_LOC_STAGE_SA</td>
</tr>
<tr>
<td></td>
<td>ZTASTINC_LOC_STAGE_EU</td>
</tr>
<tr>
<td></td>
<td>ZTASTINC_LOC_STAGE_OTH</td>
</tr>
</tbody>
</table>
Below we show screenshots of how each process chain for region SA should look like and you can follow the same approach to build the process chains for the rest of the regions:
At this point we are finished with the Modelling part (BI implementation) and the next step to tackle is the PFC Customizing to be able to load the data.

Quality Validation

Only one DSO is in focus for this section and it is the one mentioned below:

<table>
<thead>
<tr>
<th>DSO</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/DDF/DS81</td>
<td>Quality Validation</td>
</tr>
</tbody>
</table>

Note:

1. You need to check this section only if you plan to implement Quality Validation for your MD uploads, otherwise you can skip it.
2. Quality Validation is technically part of the Acquisition Layer. For a better structure and visibility of the document, and to ease the understanding, we chose to allocate a dedicated section for it.
3. In all upcoming sections we refer to Quality Validation by the abbreviation QV and to Stored Procedure by SP.

Hint:

To ease the implementation we recommend that you first complete your MD SPOs setup (for acquisition and propagation layer) without QV, try out a data upload, and tackle the QV deployment with SPO after you manage to successfully upload MD within your SPO partitions as designed. This will help isolate any possible issues and to allow proper testing.
Overview
The QV process has two aspects regarding its deployment with SPOs, as mentioned below:

1. To evaluate the quality (or to calculate the QV KPIs) of the uploaded MD, the process needs to redirect the access of the standard DSiM MD DSOs to the newly-introduced MD SPOs.

2. Store the calculated QV KPIs into an SPO instead of the DSiM standard QV DSO (DS81).

These two aspects (or the adjustments needed for these two aspects) are independent of one another and whereas the first aspect/adjustment is mandatory the second one is optional. Indeed, the QV process will still be properly functional if we choose to store the KPIs in the standard DSiM QV DSO (DS81) rather than using an SPO instead. It is strongly recommended to use an SPO to store the QV KPIs. This is to allow independent/decoupled data uploads and benefit from all adding values mentioned at the beginning of this document.

Below we give a high-level overview of what needs to be performed and we will describe later on in details the necessary technical steps:

1. Create an SPO and assign it the QV multi provider to replace DS81.

2. Create stored procedure template instances (for some QV function types) to access the different SPO partitions. This is to check/evaluate the uploaded MD and calculate the necessary KPIs. We use one stored procedure template instance per SPO partition for each of the given function types.

3. Implement BADI “Stored Procedures” to:
   a. Call the “custom” stored procedures
   b. Derive in runtime the proper SPO partition from which to read the QV KPIs

Note:
1. From the list above, only adjustments for point 2 and 3.a are mandatory. Points 1 and 3.b are needed only if you choose to substitute DS81 by an SPO.

2. If for any reason you don’t plan to substitute DS81 by an SPO, you can skip the sections below and jump directly to the one about the stored procedures.

3. The QV process is the same for both data upload scenarios. Therefore the next steps are valid for both PULL and PUSH data upload scenarios.

Partition Values
The Quality Validation is a process that is common to the Master and Transactional data. Nevertheless, it is the InfoObject /DDF/DDA_PART (Partition Value Transactional Data) that is part of the QV DS81 key fields. Therefore, any SPO that substitutes to the QV DS81 can only be partitioned using /DDF/DDA_PART. To follow the same data partitioning model that we implemented for the MD, we need to use/define similar values for /DDF/DDA_PART as we did for Partition value for MD (CTX_PART), as shown below:
Note: Without this assignment (screenshot 3) the KPIs of the QV check will not be uploaded into the proper QV SPO partitions.

**DataSources**

For the same reasons and explanations mentioned previously under the section *DataSources* above, we will use the standard QV DataSource and create several InfoPackages following the steps below:

1. Use DataSource /DDF/DATA_QUALITY_VALIDATION and create as many InfoPackages as SPO partitions that you have. In our case, this is 3 partitions. To ease the process, you can copy the InfoPackage as highlighted below:

   ![Copy InfoPackage](image)

   The outcome for our prototype would look like the one shown below:

   ![InfoSource & Transformations](image)

   Since we are using the standard DataSource, we will use Standard InfoSource and transformations to bind to the SPO. Hence, no effort is required in this section.

**SPO**

1. Copy the standard QVD SO. Give it an appropriate name, description and most importantly, select the checkbox *Semantically Partitioned*, highlighted below:
2. Create the partitions for the regions we set above where /DDF/DDA_PART is the characteristic criteria and using the partition values previously created, as shown below:

3. Create a transformation where the target is set per default and the source is the standard InfoSource IN for QV named /DDF/DATA_QUALITY_VALIDATION. This way we inherit and reuse the standard transformation.

4. Create the DTPs, but first create a DTP template which contains the parameter configuration that corresponds to the settings that can be configured in a DTP. These settings are applied when DTPs are generated and they are the same as the standard SAP DSiM DTPs from DataSource to QV DS81:
To generate the DTPs, drag and drop the template on the SPO partitions, as highlighted below:

5. We have now reached the last step and it consists of generating the process chains: one for each DTP as shown below:

Note: As previously mentioned, it is technically possible and feasible to create one process chain to include all DTPs. However, to have independent decoupled data uploads, and benefit from all adding values we mentioned above, we highly recommend creating one process chain for each DTP. This way you ensure that there is no overlap between the different process chains and they can therefore be executed in parallel. This will allow the Process Flow Controller to trigger uploads in all partitions in parallel, whereas with one process chain an upload for a given partition can only be triggered if no other partition/upload is already making use of the process chain.

At this stage, we are finished with the Quality Validation SPO. The resulting data flow should look like the following:
6. Adjust the process chains, generated via the SPO wizard, to add:
   a. The accurate InfoPackage
   b. The SAP DSiM process type *Quality Validation Decision*
   c. The SAP DSiM process type *Update Step Status*, with both variants/status values: 0070 (Status Complete) and 1400 (Status Error - Data delivery agreement Level (Subsequent Steps))

**Hint:**

1. The process chains generated via the SPO wizard are all located under the display component shown below:

   ![Process Chain Display](image)

2. We suggest you create new process chains to include/wrap the InfoPackage, the process chain generated by the SPO wizard and the SAP DSiM process types “Update step status” and “Quality validation Decision”.
The completed process chain should look like the one shown below:

7. Assign the QV SPO to the QV Multi-Provider /DDF/MP01 and assign all the InfoObjects of the added SPO, as shown below, and activate:
8. Use the customizing below to assign the created SPO (Z_SP81) to the standard DataStore (DS81)

This is necessary, on one hand, to indicate that the standard QV DSO is now replaced by an SPO and, on the other hand, to allow the SAP DSiM process type *Quality validation Decision* to **derive on runtime the SPO** where the QV KPIs are stored.

9. To allow the SAP DSiM process type *Quality validation Decision* to **derive on runtime the exact SPO partition** where the QV KPIs are stored, we need, in addition to step 8 above, to implement the **BADI/DDF/CALLPROCEDURE** provided in the DSiM customizing shown below:
The method to implement for this purpose is \textit{GET\_DSO}. A sample code of the related necessary business logic is provided in the example implementation of this BADI. You can reuse the provided code unless you have specific handling you want to implement. To reuse the provided business logic you can choose the option “Copy sample class”, as highlighted below, when implementing the BADI. Save and activate.

\textit{Stored procedures}\n
The HANA Stored Procedures are \textbf{key elements in the QV process}. This is where the data is read and checked, based on some rules, to provide KPIs to the outside. Below is a screenshot to show where these stored procedures reside in the HANA content:
The list of the executed HANA Stored Procedures and the order or sequence of their execution is controlled by a customizing that resides under the IMG path shown below:

You can also use transaction code /DDF/QV_DEF to define a specific QV sequence and/or define specific key figure settings for your DDAGRs.

It is not in scope for this section to explain this customizing, as you have the freedom to set this up as needed to fit your own business context. Instead, the focus here is to:

1. Identify the stored procedures that need to be handled to read data from the newly introduced SPOs
2. Define the input parameters to submit for each stored procedure when creating an instance of the related template
3. Show how to call the new stored procedures instead of the standard ones

Each of these points is detailed in the steps mentioned below:

1. We first need to define the QV functions that we would like to check in order to evaluate the quality of the uploaded MD. Below is a screenshot to highlight these functions and their sequence we set for our use case:

2. For each of the QV functions defined above we need to instantiate the related stored procedure template to access the different SPO partitions. For our use case this leads to 18 stored procedure template instances to create (6 QV functions X 3 SPO partitions). Below is the list of these instances and related input parameter values for SPO partition SA, for instance.
Note:
For more information about how to work with procedure templates and their instances, see the HANA developer guide. For more information about the usage of procedure template instances with semantic partitioning, we recommend you check the IMG documentation for BADI /DDF/CALL_PROCEDURE, as well as the link provided under Additional Links section for this purpose.

- **NO_OF_LOCS**
  - DSiM SP template: PT_KPI_NO_OF_LOCATIONS
  - SP template instance for region SA: Z_NO_OF_LOC_TASTE_SA
  - Input parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOCATION_ACQUISITION</td>
<td>&quot;SAPDIM&quot;.&quot;/BIC/AZ_SP210100&quot;</td>
</tr>
</tbody>
</table>

- **NO_OF_LOCS_WO_GLN**
  - DSiM SP template: PT_KPI_NO_LOC_WO_GLN
  - SP template instance for region SA: Z_NO_LOC_WO_GLN_TASTE_SA
  - Input parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOCATION_ACQUISITION</td>
<td>&quot;SAPDIM&quot;.&quot;/BIC/AZ_SP210100&quot;</td>
</tr>
<tr>
<td>LOCATION_SOURCE</td>
<td>&quot;SAPDIM&quot;.&quot;/IDD/PLOCREF&quot;</td>
</tr>
</tbody>
</table>

- **NO_OF_NEW_LOCS**
  - DSiM SP template: PT_KPI_NO_OF_NEW_LOCATIONS
  - SP template instance for region SA: Z_NO_NEW_LOC_TASTE_SA
  - Input parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRODUCT_ACQUISITION</td>
<td>&quot;SAPDIM&quot;.&quot;/BIC/AZ_SP410100&quot;</td>
</tr>
<tr>
<td>PRODUCT_SOURCE</td>
<td>&quot;SAPDIM&quot;.&quot;/IDD/PPRODREF&quot;</td>
</tr>
</tbody>
</table>

- **NO_OF_NEW_PRODS**
  - DSiM SP template: PT_KPI_NO_OF_NEW_PRODUCTS
  - SP template instance for region SA: Z_NO_NEW_PROD_TASTE_SA
  - Input parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRODUCT_ACQUISITION</td>
<td>&quot;SAPDIM&quot;.&quot;/BIC/AZ_SP410100&quot;</td>
</tr>
</tbody>
</table>

- **NO_OF_PRODS_WO_GTIN**
  - DSiM SP template: PT_KPI_NO_PROD_WO_GTIN
  - SP template instance for region SA: Z_NO_PROD_WO_GTIN_TASTE_SA
  - Input parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRODUCT_ACQUISITION</td>
<td>&quot;SAPDIM&quot;.&quot;/BIC/AZ_SP410100&quot;</td>
</tr>
</tbody>
</table>
Hint:

1. To easily or dynamically identify the SP to call for the given DDAGR on runtime, we followed a naming convention for the SP template instances as follows:

   \(<a\_given\_name\_based\_on\_the\_QV\_function> + \langle/DF\_DDA\_PART\_of\_the\_SPO\_partition>\)

   This way, and as explained later on, for method `GET_SP_NAME` we can hardcode the common part of the instance name and concatenate to the `/DF/ DD_A_PART` that can be derived from the DDAGR.

2. To identify the HANA DB table name of the given SPO partition to submit as input parameter, you can proceed as highlighted in the screenshots below:

   ![Screenshot 1](image1.png)
   ![Screenshot 2](image2.png)

3. To ensure that the new or custom stored procedures are called instead of the DSiM standard ones, we need to implement `BADI/DF/CALL_PROCEDURE` provided in DSiM as shown below:

   ![Screenshot 3](image3.png)
   ![Screenshot 4](image4.png)

   The method to implement for this purpose is `GET_SP_NAME`. Below is a code snapshot that we provided for our SPOs setup described in this document. The code provided below is just a proposal and it is based on the naming convention we used for our new stored procedures, described above. To be able to use it you need to follow the same naming convention for your stored procedures, as we did, otherwise you can adjust it to fit your implementation.
DATA lv_spo_sp_path TYPE string.
DATA ls_context_data TYPE /ddf/s_context_data.

IF iv_context = 'TASTE_SA_1' OR iv_context = 'TASTE_SA_2'
  OR iv_context = 'TASTE_EU_1' OR iv_context = 'TASTE_OTR1'.
  *Return the QV stored procedure for Taste INC. MD SPO
  *Assign the SPO stored procedures path within the HANA content

  CONCATENATE '"' '_SYS_BIC' '"' '.' '"' 'sap.ddf.dsimddf.qv_ext/'
  INTO lv_spo_sp_path.

  CLEAR ev_sp_name.

  CASE iv_function.
    WHEN 'NO_OF_LOCS'.
      ev_sp_name = 'Z_NO_OF_LOC'.
    WHEN 'NO_OF_LOCS_WO_GLN'.
      ev_sp_name = 'Z_NO_LOC_WO_GLN'.
    WHEN 'NO_OF_NEW_LOCS'.
      ev_sp_name = 'Z_NO_NEW_LOC'.
    WHEN 'NO_OF_NEW_PRODS'.
      ev_sp_name = 'Z_NO_NEW_PROD'.
    WHEN 'NO_OF_PRODS'.
      ev_sp_name = 'Z_NO_OF_PROD'.
    WHEN 'NO_OF_PRODS_WO_GTIN'.
      ev_sp_name = 'Z_NO_PROD_WO_GTIN'.
  END_CASE.

  *Fetch the DDAGR partition value
  IF ev_sp_name IS NOT INITIAL.
  TRY.
    CALL METHOD /ddf/cl_adu_datagrmt_cust=>read_context
      EXPORTING
        iv_cdpa = is_ddagr_attr-cdpa
      IMPORTING
        es_context_data = ls_context_data.
    CATCH /ddf/cx_adu_customizing.
      * handle exception
      ENDTRY.

  *Concatenate the stored procedure name with the DDAGR partition value
  *this is to obtain the exact stored procedure name related to the DDAGR SPO partition
  CONCATENATE lv_spo_sp_path ev_sp_name '_'- ls_context_data-partvalm INTO ev_sp_name.
  ELSE.
    *Standard QV handling
    ev_sp_name = iv_sp_name.
    ENDIF.
  ELSE.
    *Standard QV handling
    ev_sp_name = iv_sp_name.
    ENDIF.
4. Assign the new QV sequence to your DDAGRs using transaction code /DDF/QV_DEF

At this stage, we have completed the Quality Validation process deployment with SPOs for master data upload. You can now start to upload data and test the setup.

Process Flow Control

Now that we have completed all necessary BW setup and objects, we will describe how to customize the process flow control to ensure proper data upload into correct partition for each region.

Customizing

Process definition

This is the first set up to undertake. It consists of defining the required PFC steps, and related execution sequence, for the data upload. To do this we can follow one of the options below:

a. Define one process definition for each DDAGR, or
b. Define one process definition for all DDAGRs

We don’t make any recommendation on whether to go with option a or b, as there is no major difference/impact from a data upload process perspective. Therefore you are free to choose the approach with which you are more comfortable. In the following sections we show how instantiation of processing steps is done for option b since the setup of option a is more or less obvious. It might be interesting to see what needs to be done if we choose to use one process definition for all DDAGRs.

1. Go to the IMG path for Process Definition creation, highlighted below:

2. Create a process definition similar to the one shown below, where you have to make sure you properly set the reference key field to:

   a. Keep it empty for steps that have to be executed for uploads regardless of the region. In other words, the steps that are common to upload for all regions/DDAGRs (if any)
b. Assign the same key/value to uniquely identify all steps that have to be executed for upload of a given/specific region/DDAGRs

Below we highlight the “how to” for each data upload scenario:

**Data upload – PULL**

<table>
<thead>
<tr>
<th>Step No.</th>
<th>Description</th>
<th>Parallelization</th>
<th>Reference Key</th>
<th>Process Chain ID</th>
<th>Upload Stage</th>
<th>Activate</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>Product extraction to Acqu Layer - SA</td>
<td>PROP</td>
<td>TASTE_SA</td>
<td>4_SP41_PROD_FILE_SA</td>
<td>Data Acquisition</td>
<td>✓</td>
</tr>
<tr>
<td>110</td>
<td>Product extraction to Acqu Layer - EU</td>
<td>PROP</td>
<td>TASTE_EU</td>
<td>4_SP41_PROD_FILE_EU</td>
<td>Data Acquisition</td>
<td>✓</td>
</tr>
<tr>
<td>120</td>
<td>Product extraction to Acqu Layer - Other</td>
<td>PROP</td>
<td>TASTE_OTHR</td>
<td>4_SP41_PROD_FILE_OTHR</td>
<td>Data Acquisition</td>
<td>✓</td>
</tr>
<tr>
<td>140</td>
<td>Location extraction to Acqu Layer - SA</td>
<td>AQC</td>
<td>TASTE_SA</td>
<td>6_SP21_LOC_FILE_SA</td>
<td>Data Acquisition</td>
<td>✓</td>
</tr>
<tr>
<td>150</td>
<td>Location extraction to Acqu Layer - EU</td>
<td>AQC</td>
<td>TASTE_EU</td>
<td>6_SP21_LOC_FILE_EU</td>
<td>Data Acquisition</td>
<td>✓</td>
</tr>
<tr>
<td>160</td>
<td>Location extraction to Acqu Layer - Other</td>
<td>AQC</td>
<td>TASTE_OTHR</td>
<td>6_SP21_LOC_FILE_OTHR</td>
<td>Data Acquisition</td>
<td>✓</td>
</tr>
<tr>
<td>200</td>
<td>Product upload to Stage Layer - SA</td>
<td>PROP</td>
<td>TASTE_SA</td>
<td>7_BC01_PROD_STAG_SA</td>
<td>Data Propagation</td>
<td>✓</td>
</tr>
<tr>
<td>210</td>
<td>Product upload to Stage Layer - EU</td>
<td>PROP</td>
<td>TASTE_EU</td>
<td>7_BC01_PROD_STAG_EU</td>
<td>Data Propagation</td>
<td>✓</td>
</tr>
<tr>
<td>220</td>
<td>Product upload to Stage Layer - Other</td>
<td>PROP</td>
<td>TASTE_OTHR</td>
<td>7_BC01_PROD_STAG_OTHR</td>
<td>Data Propagation</td>
<td>✓</td>
</tr>
<tr>
<td>240</td>
<td>Location upload to Stage Layer - SA</td>
<td>AQC</td>
<td>TASTE_SA</td>
<td>7_BC01_LOC_STAG_SA</td>
<td>Data Propagation</td>
<td>✓</td>
</tr>
<tr>
<td>250</td>
<td>Location upload to Stage Layer - EU</td>
<td>AQC</td>
<td>TASTE_EU</td>
<td>7_BC01_LOC_STAG_EU</td>
<td>Data Propagation</td>
<td>✓</td>
</tr>
<tr>
<td>260</td>
<td>Location upload to Stage Layer - Other</td>
<td>AQC</td>
<td>TASTE_OTHR</td>
<td>7_BC01_LOC_STAG_OTHR</td>
<td>Data Propagation</td>
<td>✓</td>
</tr>
</tbody>
</table>

**Data upload - PUSH**

<table>
<thead>
<tr>
<th>Step No.</th>
<th>Description</th>
<th>Parallelization</th>
<th>Reference Key</th>
<th>Process Chain ID</th>
<th>Upload Stage</th>
<th>Activate</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>Product extraction to Acqu Layer - SA</td>
<td>PROP</td>
<td>TASTE_SA</td>
<td>5_BC01_PROD_DS_SA</td>
<td>Data Acquisition</td>
<td>✓</td>
</tr>
<tr>
<td>110</td>
<td>Product extraction to Acqu Layer - EU</td>
<td>PROP</td>
<td>TASTE_EU</td>
<td>5_BC01_PROD_DS_EU</td>
<td>Data Acquisition</td>
<td>✓</td>
</tr>
<tr>
<td>120</td>
<td>Product extraction to Acqu Layer - Other</td>
<td>PROP</td>
<td>TASTE_OTHR</td>
<td>5_BC01_PROD_DS_OTHR</td>
<td>Data Acquisition</td>
<td>✓</td>
</tr>
<tr>
<td>140</td>
<td>Location extraction to Acqu Layer - SA</td>
<td>AQC</td>
<td>TASTE_SA</td>
<td>5_BC01_LOC_DS_SA</td>
<td>Data Acquisition</td>
<td>✓</td>
</tr>
<tr>
<td>150</td>
<td>Location extraction to Acqu Layer - EU</td>
<td>AQC</td>
<td>TASTE_EU</td>
<td>5_BC01_LOC_DS_EU</td>
<td>Data Acquisition</td>
<td>✓</td>
</tr>
<tr>
<td>160</td>
<td>Location extraction to Acqu Layer - Other</td>
<td>AQC</td>
<td>TASTE_OTHR</td>
<td>5_BC01_LOC_DS_OTHR</td>
<td>Data Acquisition</td>
<td>✓</td>
</tr>
<tr>
<td>200</td>
<td>Product upload to Stage Layer - SA</td>
<td>PROP</td>
<td>TASTE_SA</td>
<td>1_BC01_PROD_STAG_SA</td>
<td>Data Propagation</td>
<td>✓</td>
</tr>
<tr>
<td>210</td>
<td>Product upload to Stage Layer - EU</td>
<td>PROP</td>
<td>TASTE_EU</td>
<td>1_BC01_PROD_STAG_EU</td>
<td>Data Propagation</td>
<td>✓</td>
</tr>
<tr>
<td>220</td>
<td>Product upload to Stage Layer - Other</td>
<td>PROP</td>
<td>TASTE_OTHR</td>
<td>1_BC01_PROD_STAG_OTHR</td>
<td>Data Propagation</td>
<td>✓</td>
</tr>
<tr>
<td>240</td>
<td>Location upload to Stage Layer - SA</td>
<td>AQC</td>
<td>TASTE_SA</td>
<td>1_BC01_LOC_STAG_SA</td>
<td>Data Propagation</td>
<td>✓</td>
</tr>
<tr>
<td>250</td>
<td>Location upload to Stage Layer - EU</td>
<td>AQC</td>
<td>TASTE_EU</td>
<td>1_BC01_LOC_STAG_EU</td>
<td>Data Propagation</td>
<td>✓</td>
</tr>
<tr>
<td>260</td>
<td>Location upload to Stage Layer - Other</td>
<td>AQC</td>
<td>TASTE_OTHR</td>
<td>1_BC01_LOC_STAG_OTHR</td>
<td>Data Propagation</td>
<td>✓</td>
</tr>
</tbody>
</table>

**Note:**

You may notice that as reference key, for all steps of a given region, we chose to use the partition value for Master Data of the given region. This is straightforward for our use case; however it is NOT a requirement. You can define and use any reference key value that is most suitable to the use case you might implement. What's important, as shown below, is that you are later able to derive this value during step instantiation, based on the relevant DDAGR.
Data delivery agreement
As mentioned previously, the DDAGRs we need to create for our use case are as follows:

<table>
<thead>
<tr>
<th>Region</th>
<th>CDPA</th>
<th>Data Delivery Agreement ID</th>
<th>Description</th>
<th>Partition Values for Master Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>SA</td>
<td>TASTE_SA_1</td>
<td>TASTE_BR</td>
<td>Taste Inc. SA - Brazil</td>
<td>TASTE_SA</td>
</tr>
<tr>
<td></td>
<td>TASTE_SA_2</td>
<td>TASTE_AR</td>
<td>Taste Inc. SA - Argentina</td>
<td></td>
</tr>
<tr>
<td>EU</td>
<td>TASTE_EU_1</td>
<td>TASTE_SP</td>
<td>Taste Inc. EU - Spain</td>
<td>TASTE_EU</td>
</tr>
<tr>
<td>Other</td>
<td>TASTE_OTR1</td>
<td>TASTE_BI</td>
<td>Taste Inc. AS - Bahrain</td>
<td>TASTE_OTHER</td>
</tr>
</tbody>
</table>

1. We first create the necessary CDPAs as shown below and most importantly we link them to the correct partition value for master data:

   Note: if you also have implemented the Quality Validation process and partitioned DS81 make sure to also assign the partition values to the DDAGRs as previously mentioned.

2. Below we show an example of how a DDAGR for each data upload scenario should look:

![Diagram of DDAGR settings]

Settings for PULL data upload scenario

Process definition for PULL data upload scenario
3. For each DDAGR perform the dataset assignment as shown below:
Hint:
If you choose to go with one process definition for all DDAGRs, make sure to assign the correct steps to the DDAGR. To help you do this properly, we suggest coming up with, in advance, a naming convention or other technique, as we show above, to make it easier to recognize the relevant processing steps for each DDAGR/Region.

Data Set File
1. For each DDAGR, and of course only for the PULL data upload scenario, create the file set definition as highlighted below:

   ![Image of file set definition]

   **File Set Definition**
   - Description: FS_TASTE_BR
   - **File Name Pattern**
     - TASTE_INC_...........PRO
     - TASTE_INC_...........LOC
     - TASTE_INC_...........FIN

2. Complete the definition by assigning the correct data set to each file, and then saving.
**Steps Instantiation**

This section is only relevant if you defined one process definition for all DDAGRs. If that is not the case just skip it and can ready to start testing your set up!

When we define one process definition for all DDAGRs, we need to help PFC to identify, on runtime, what are the steps to instantiate, and hence the ones NOT to instantiate, for the handled DDAGR. To do this we need to implement the BADI shown below, provided in SAP DSiM for this purpose:

<table>
<thead>
<tr>
<th>Demand Data Foundation</th>
</tr>
</thead>
<tbody>
<tr>
<td>▼ Data Model</td>
</tr>
<tr>
<td>▼ Data Upload</td>
</tr>
<tr>
<td>▼ Business Add-Ins (BAdIs)</td>
</tr>
<tr>
<td>▼ BAdI: Data Delivery Enhancements</td>
</tr>
<tr>
<td>▼ BAdI: Step Enhancements</td>
</tr>
<tr>
<td>▼ Check Instantiation</td>
</tr>
</tbody>
</table>

This BADI has only one method and the idea behind it is as follows:

a. Consider that we previously associated a “reference key” to identify steps of each region

b. On runtime based on the CDPA of the submitted DDAGR, we identify the steps to instantiate

c. Steps to instantiate are the ones that hold the reference key with a value equal to the master data partition value associated to the given CDPA. *That is true for our use case, but might be different if you chose different reference keys.*

d. Instantiated steps, contain the proper process chains to bring/load data into the correct partition
Below you see a code snapshot of a "proposal", for the set up mentioned in this document, that you could use as an example.

**GET_RELEVANT_STEPS**

```abap
METHOD /ddf/if_ex_step-get_relevant_steps.

DATA lv_ref_key TYPE /ddf/refkey.
DATA ls_context_data TYPE /ddf/s_context_data.

FIELD-SYMBOLS <fs_step_output> LIKE LINE OF ct_step_relevant.
FIELD-SYMBOLS <fs_step> LIKE LINE OF it_step.

REFRESH ct_step_relevant.
IF is_data_agreement-process_def = 'TSTINC_FIL'
  OR is_data_agreement-process_def = 'TSTINC_DS'.
  TRY.
    CALL METHOD /ddf/cl_adu_datagramnt_cust=>read_context
      EXPORTING
        lv_cdpa = is_data_agreement-cdpa
      IMPORTING
        es_context_data = ls_context_data.
      CATCH /ddf/cx_adu_Customizing .
      lv_ref_key = ls_context_data-partvalm.
  ENDTRY.
ENDIF.

LOOP AT it_step ASSIGNING <fs_step>
  WHERE refkey IS NOT INITIAL
  AND refkey <> lv_ref_key.
  LOOP AT ct_step_relevant ASSIGNING <fs_step_no>
    WHERE step_no = <fs_step>-step_no.
    CLEAR <fs_step_no>. 
    EXIT.
  ENDLOOP.
ENDLOOP.
ENDMETHOD.
```
ADDITIONAL LINKS

BW 7.30: Semantically Partitioned Objects
DSiM and SAP BusinessObjects Data Services - Configuration Manual
Master Data in DSiM: Data Model
Master Data in DSiM: Data Flow
Setting up SPOs in DSiM
HANA Stored Procedures: Accessing the Correct Partitions
HANA Developer guide