



VDP Naming Conventions

Revision 20210513

NOTE

This document is confidential and proprietary of **Denodo Technologies**.
No part of this document may be reproduced in any form by any means without prior written authorization of **Denodo Technologies**.

Copyright © 2022
Denodo Technologies Proprietary and Confidential

CONTENTS

1 GOAL.....	4
2 GENERAL RECOMMENDATIONS.....	4
3 VDP PROJECT STRUCTURE.....	5
3.1 PROJECT LAYERS.....	5
3.2 FOLDER STRUCTURE.....	5
4 VDP NAMING CONVENTIONS.....	8
4.1 DATA SOURCES.....	8
4.2 BASE VIEWS.....	8
4.3 INTEGRATION VIEWS.....	8
4.4 INTERFACES.....	9
4.5 FINAL VIEWS.....	9
4.6 STORED PROCEDURES.....	10
4.7 ASSOCIATIONS.....	10
4.8 WEB SERVICES/WIDGETS.....	10
4.9 JMS LISTENERS.....	11
4.10 SUMMARIES.....	11
4.11 VIEW ATTRIBUTES.....	12

1 GOAL

This document enumerates the proposed naming conventions for all the elements that can be created as part of a Virtual DataPort database.

2 GENERAL RECOMMENDATIONS

- Defining a naming conventions document is strongly recommended in the organizations to reduce the effort of reading and understanding the elements created in Virtual DataPort.
- Name Views/Attributes in lowercase and words separated by "_".
- Name Views using singular (customer, contract, ...).
- Use English, if the final consumer applications require a different language, use that language for the 'final' views and/or stored procedures that are directly accessed by these applications.
- Use the Metadata > Description option to document the elements in a database.
- Use the attribute's description field to document every attribute.
- Do not include as part of the name something that depends on the environment where the element is being created.

3 VDP PROJECT STRUCTURE

3.1 PROJECT LAYERS

Different layers are identified when creating elements in a Data Virtualization project:

- **Connectivity:** Related to the physical systems. Data sources and base views are part of this level.
- **Integration:** Includes the combinations and transformations for the next layers. Not directly consumed views at this level.
- **Business Entities:** Canonical business entities exposed to all users.
- **Report Views:** Pre-built reports and analysis frequently consumed by users.
- **Data Services:** Web services for publishing views from other levels. Can contain views needed for data formatting and manipulation.

3.2 FOLDER STRUCTURE

It is important to use an organized Folder structure, specially for complex projects, as it will help to locate elements and have a clear understanding of the different elements of the project.

Based on the project structure described above, these are the recommended folders:

- One folder per layer.
- Additional folders for other elements, for example: associations, jms listeners, ...
- In complex projects, additional subfolders can be added to group related elements.

This is an example of the recommended root folders:

1 - Connectivity

This folder stores the data sources and base views of the project. Additional subfolders can be added at this level:

- 1 - data sources
- 2 - base views

2 - Integration

The integration views created for combining the different base views are stored in this folder. Additional subfolders can be created for grouping the related integration views.

3 - Business entities

The interfaces representing the business entities will be stored at this level. Additional subfolders can be added. The implementation views, since they are not going to be directly accessed by the client application, can remain in the integration level.

4 - Report views

Interfaces representing the pre-built reports published to the final application will be stored in this folder.

5 - Data services

This folder stores data services:

- Web services:
 - SOAP
 - REST
- Widgets

6 - Associations

This folder stores associations.

7 - JMS listeners

This folder stores JMS listeners

8 - Stored procedures

This folder stores custom stored procedures developed using the VDP API.

9 - Summaries

This folder stores Summaries.

Below you can see a sample VQL that will create the recommended folder structure when executed in a new database:

```
CREATE OR REPLACE FOLDER '/01 - Connectivity' DESCRIPTION 'This folder
stores the data sources and base views of the project' ;

CREATE OR REPLACE FOLDER '/01 - Connectivity/01 - Data Sources' DESCRIPTION
'This folder stores the data sources of the project' ;

CREATE OR REPLACE FOLDER '/01 - Connectivity/02 - Base Views' DESCRIPTION
'This folder stores the base views of the project' ;

CREATE OR REPLACE FOLDER '/02 - Integration' DESCRIPTION 'The integration
views created for combining the different base views are stored in this
folder' ;
```

```
CREATE OR REPLACE FOLDER '/03 - Business Entities' DESCRIPTION 'The
interfaces representing the business entities will be stored at this level.
Additional subfolders can be added. The implementation views, since they are
not going to be directly accessed by the client application, can remain in
the integration level.' ;

CREATE OR REPLACE FOLDER '/04 - Report Views' DESCRIPTION 'Interfaces
representing the pre-built reports published to final application will be
stored in this folder.' ;

CREATE OR REPLACE FOLDER '/05 - Data Services' DESCRIPTION 'This folder
stores data services.' ;

CREATE OR REPLACE FOLDER '/06 - Associations' DESCRIPTION 'This folder
stores associations.' ;

CREATE OR REPLACE FOLDER '/07 - JMS Listeners' DESCRIPTION 'This folder
stores JMS listeners' ;

CREATE OR REPLACE FOLDER '/08 - Stored Procedures' DESCRIPTION 'This folder
stores Stored procedures' ;

CREATE OR REPLACE FOLDER '/09 - Summaries' DESCRIPTION 'This folder stores
Summaries' ;
```

4 VDP NAMING CONVENTIONS

4.1 DATA SOURCES

Data sources can be named as follows:

- **ds_{data source name}**
 - E.g.:
 - ds_internet
 - ds_soap_billing
 - ds_phone

4.2 BASE VIEWS

Base views can be named as follows:

- **bv_{data source name}_{source table / description (if sql-query view) / source ws operation / stored procedure / ... (optional)}**
 - **{data source name}** is the name of the data source over which the base view is created.
 - E.g.:
 - bv_internet_incident
 - bv_phone_incident
 - **{source table / description (if sql-query view) / source ws operation / stored procedure / ...}** is the name of the table, stored procedure, web service operation or any descriptive text to identify what it is imported from the data source. It is optional if only one base view is created for a data source. For example, to import a single XML file called "user_incidents.xml" in VDP the data source will be called "ds_user_incident" and the base view will be called "bv_user_incident".
 - E.g.:
 - bv_internet_incident

Using this naming convention the quick search will list all the views related to a specific data source when searching by the data source name.

With this approach all the base views from a specific data source will also appear together if using external JDBC/ODBC clients to access a VDP database.

4.3 INTEGRATION VIEWS

As defined in the project structure section, integration views are those derived views created for combination and transformation but they should not be directly accessed by final client applications. These integration views can be named as follows:

- **iv_{integration view descriptive name}_{order #}_{action, description} (optional)**
 - **{integration view descriptive name}** is the descriptive name for this integration view.
 - In order to get all related integration views ordered it is recommended to use the same name for all of them.
 - This descriptive name can be:
 - Similar to the base view used.
 - Similar to the final view that will use this integration view.
 - E.g.:
 - iv_data_customer_mail_03_flatten
 - iv_data_personal_05_register_address
 - **{order #}** is the numeric value used to order the views.
 - If an integration view is created from different integration views, the new integration view should use the following number to the highest number of the views that the new view depends on.
 - E.g.:
 - iv_data_customer_mail_03_flatten
 - iv_data_personal_05_register_address
 - **{action, description}** is a descriptive text about the action or meaning that identifies what the integration view does.
 - E.g.:
 - iv_data_customer_mail_03_flatten
 - iv_data_07_union_customer_info

4.4 INTERFACES

Interfaces should be named using a descriptive name that makes them easy to identify for final users:

- **{interface name}**
 - E.g.:
 - customer
 - customer_address
 - customer_email

4.5 FINAL VIEWS

The name “final views” is applied to views that contain information that is directly accessed by external systems and/or published for their consumption. Final views can be named as follows:

- **{identifier}_{by_input_parameters (optional)}**
 - **{identifier}** is the identifier for this final view.
 - E.g.:
 - customer

- `get_customer_info`
- **{by_input_parameters}**: used only if this view has mandatory input parameters.
 - Use the input parameters in the name of the view separated by “_”.
 - If the final view has a lot of input parameters this notation should not be used.
 - E.g.:
 - `customer_search_by_name_lastname`

4.6 STORED PROCEDURES

Stored procedures can be named as follows:

- **sp_{sp_name}_{by_input_parameters (optional)}**
 - **{sp_name}** is the descriptive name of the stored procedure created
 - E.g.:
 - `sp_customer_global_search`
 - **{by_input_parameters}**: used only if the stored procedure is always executed with input parameters.
 - Use the input parameters in the name of the stored procedure separated by “_”.
 - If the final view has a lot of input parameters this notation should not be used.
 - E.g.:
 - `sp_customer_search_by_name_lastname`

4.7 ASSOCIATIONS

Associations can be named as follows:

- **a_{principal entity name}_{related entity}**
 - E.g.:
 - `a_customer_contract`

4.8 WEB SERVICES/WIDGETS

The Web service/Widget name should be a business noun, a business concept, or a business process.

The naming of these elements depends on the specific customer/project naming conventions.

The Web Services operation names should be similar to the final view that implements the operation.

These are the naming conventions for published Denodo Web services:

- **SOAP:**
 - **{web service name}:**
 - Name of the Soap Web Service:
 - Should include the “service” suffix.
 - Should be in lowercase.
 - Should not use separators (“ ”, “_”...).
 - Should be singular.
 - Should not use verbs.
 - E.g.:
 - **internetincident**service
 - **userinvoicing**service
 - **{Operations}:**
 - Name of the Soap Web Service operations:
 - Should not include separators (“ ”, “_”...).
 - Should use the name of the view.
 - Should use verbs.
 - E.g.:
 - **getINTERNETINCIDENTSBySTATUS**
 - **insertINTERNETINCIDENTS**
- **REST:**
 - **{web service name}:**
 - Name of the REST Web Service:
 - Should not include the “service” suffix.
 - Should be in lowercase.
 - Can use the “_” separator.
 - Should be singular.
 - Should not use verbs.
 - E.g.:
 - **internet_incident**
 - **sale**

4.9 JMS LISTENERS

JMS listeners can be named as follows:

- **jms_{jms name}**
 - E.g.:
 - **jms_customer**

4.10 SUMMARIES

Summaries can be named as follows:

- **s_{summary name}**
 - E.g.:
 - **s_total_by_year_store**

4.11 VIEW ATTRIBUTES

- Name identifier fields (id, code, ...) as "viewname_id".
 - Example: if the view name is "bv_customer_mail" and it has an "id" field it should be named "customer_mail_id".
- Name fields from other views as "relatedviewname_id".
- Name the input parameters as "input_descriptive_name".