# SQL Namespace Generator

2018-03-19

## Status

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Designer** | **Status** | **Comment** |
| 2018-03-19 | LS | Design | Initial |

## Requirement

The SQL Namespace Generator is an extension of the current SQL Data Source. It allows to specify queries to build a namespace and generate retrieval queries automatically based on the concept of place holders in subsequent queries.

The generator works on the concept of an item path and item value queries. A namespace definition file must be created to generate the namespace and parameters that can be used in subsequent queries. The definition file is a JSON file and the context will be discussed in the latter part of this document.

It allows for mass population of measures without having to manually change query parameters. If no definition file is supplied, the Data Source connector will, as per normal, browse the database to retrieve tables, views and stored procedures.

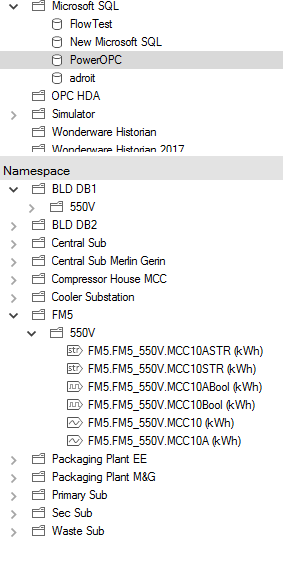
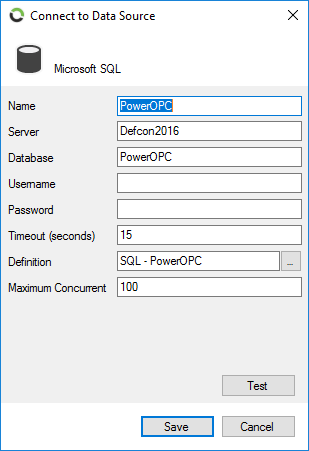


Figure 1 New Definition selection option to browse for a database definition file to auto generate a namespace

## Proposed solution

The solution will be delivered as part of the Flow Microsoft SQL data source connector.

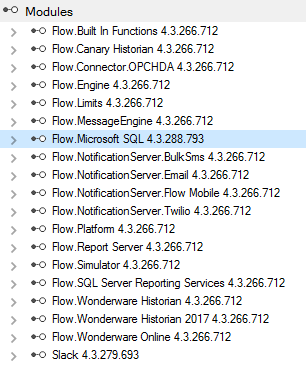


Figure 2 Microsoft SQL Module listed in the Module configuration

### SQL definition file

The SQL definition file is a JSON formatted file to specify the namespace / schema of the SQL database.

The file consists of 3 main parts:

1. Item Path query – this will generate the folder structure of your namespace
2. Item query – this creates the item in a specific folder. It will also specify items such as descriptions, unit of measure and item data types
3. Item value query – this is the query that will be used to retrieve raw or aggregated values by the Flow engine

An example of the format of such a file can be seen below:

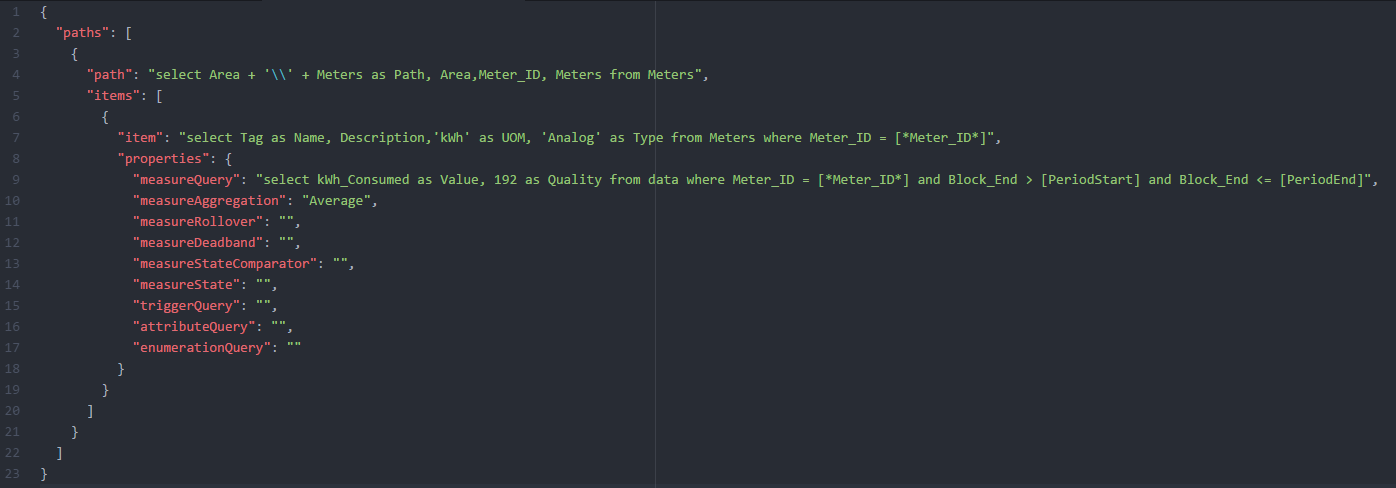


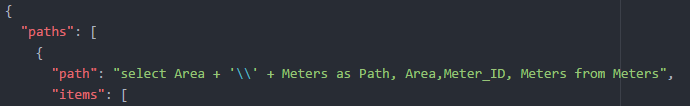
Figure 3 JSON file format for the SQL Namespace generator

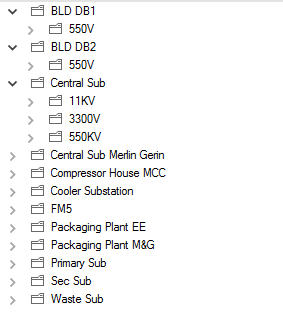
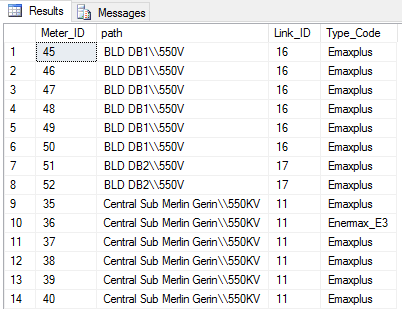
Path Query

The path query specifies the file structure of your namespace. These queries can also be defined as an array of queries in the JSON file if you need to generate more than one folder structure for different items. There are some rules that must be adhered to for the query to build a namespace successfully:

1. The query must return a column specifying the path of an item. The column must be called “path”
2. Levels off derivation in your folder structure is separated in the result of the path query by the” \\” characters.
3. The query can return other field to be used in subsequent queries
4. It must return a link to resolve your items to the folder structure you created.
5. It is good practise to order your query results by the resolving path.

See below a typical example and result set of a path query:

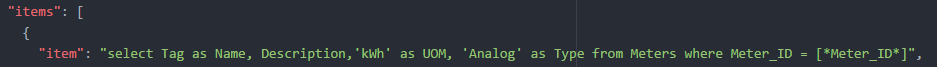




The Meter\_ID, Link\_ID and Type\_Code can be used in subsequent queries.

Item Query

The item query specifies the detail of your item and can include items such as unit of measure, description and the data type of the item. The query must resolve the item to only one path structure. These queries can also be defined as an array of queries in the JSON file if you need to generate more than one item type per path.



There are some rules that must be adhered to for the query to build add an item successfully:

1. The special characters “[\*,”,”\*]” are deemed as placeholders. The resulting row value from previous queries will be replaced within the query
2. The query must return the values in the flowing columns:
   1. Name – required. The name of the item
   2. Description – optional. The description of the item
   3. UOM – optional. The unit of measure of the item
   4. Type – optional. The data type of the item. By default, the “Analog” type will be selected if nothing is specified.
      1. Valid types include:
         1. Analog
         2. String
         3. Boolean

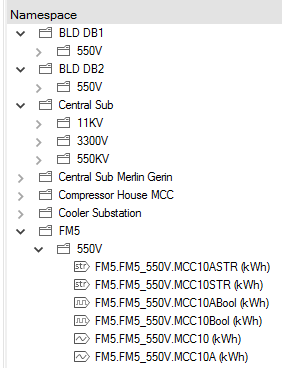


Figure 4 Resulting Items placed under their respective paths

Item Value Queries

The item value query specifies the query that will be executed by the Flow engine to return the actual values of the item for a specific time period. The values can either be aggregated by the query, or it could be the raw values and the Flow engine will do the aggregation method selected in the properties section of the file.

There are some rules that must be adhered to for the query to get item values successfully:

1. The special characters “[\*,”,”\*]” are deemed as placeholders. The resulting row value from previous queries will be replaced within the query
2. The query must return the values in the flowing columns:
   1. Value – the actual item value for a specific timestamp
   2. Quality – Quality data representing the quality data of the item. If no quality data is available, hardcode this field to 192
   3. Timestamp – Timestamp of the data. This is required if the Flow engine needs to preform an aggregation method on the data
   4. Time delimiters – [PeriodStart], PeriodEnd]. These placeholders will be replaced by the Flow engine with the correct time period based on the retrieval that must be preformed.

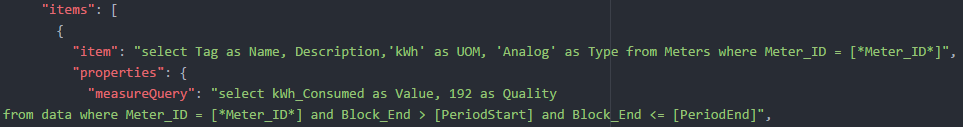


Figure 5 Item value queries

The same principle can also be applied for Trigger Queries, Attribute Queries and Enumeration Queries. They can also be defined in the JSON definition file:

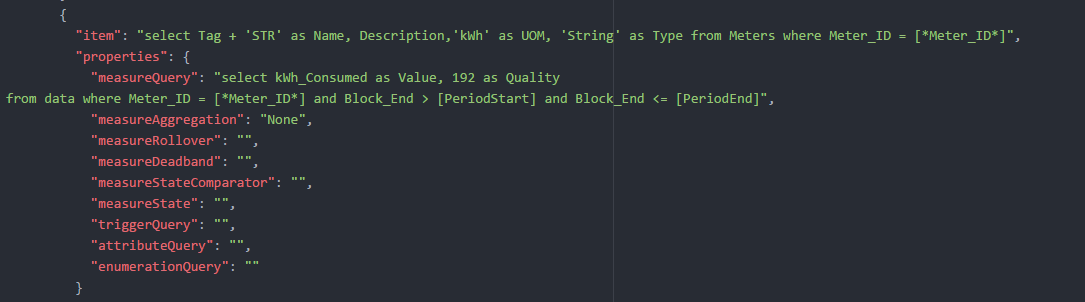
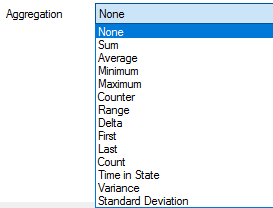


Figure 6 Trigger, Attribute and Enumeration queries

Item Properties

Other properties that can be defined, but is optional include:

1. Measure Aggregation – Aggregation method to be applied if the raw data is returned by the item value query (result must include a timestamp field).
   1. Valid values include:



1. Measure Rollover – Applicable if the Counter retrieval style is selected.
2. Measure Deadland – Applicable if the Counter retrieval style is selected.
3. Measure State Comparator – Applicable if the Time in State retrieval style is selected.
4. Measure State – Applicable if the Time in State retrieval style is selected.

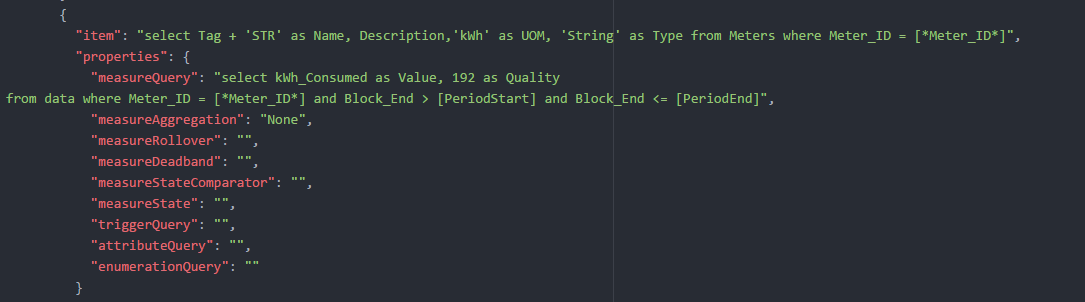


Figure Item properties to be defined

### JSON Definition file structure example

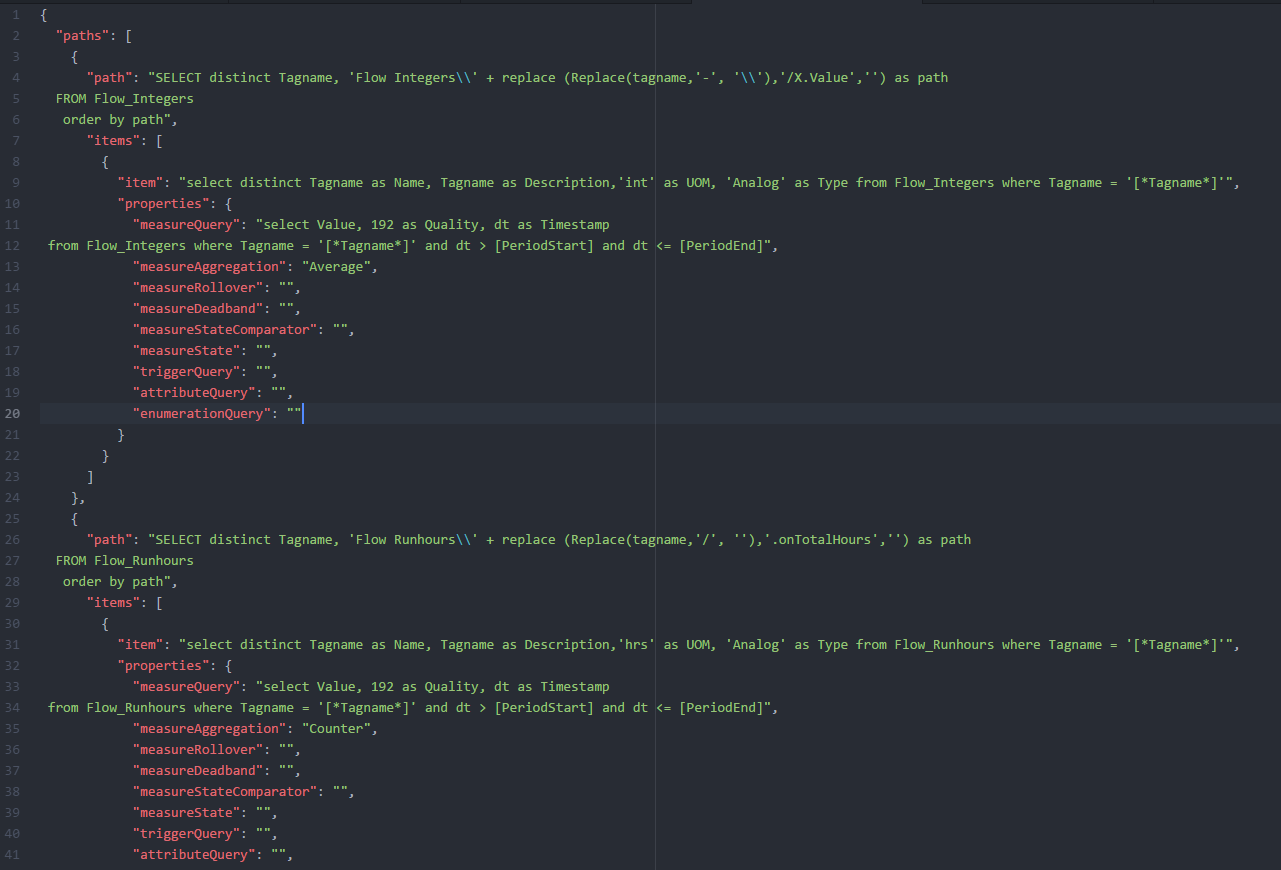


Figure SQL definition structure

#### File example

{

"paths": [

{

"path": "SELECT distinct Tagname, 'Flow Integers\\' + replace (Replace(tagname,'-', '\\'),'/X.Value','') as path

FROM Flow\_Integers

order by path",

"items": [

{

"item": "select distinct Tagname as Name, Tagname as Description,'int' as UOM, 'Analog' as Type from Flow\_Integers where Tagname = '[\*Tagname\*]'",

"properties": {

"measureQuery": "select Value, 192 as Quality, dt as Timestamp

from Flow\_Integers where Tagname = '[\*Tagname\*]' and dt > [PeriodStart] and dt <= [PeriodEnd]",

"measureAggregation": "Average",

"measureRollover": "",

"measureDeadband": "",

"measureStateComparator": "",

"measureState": "",

"triggerQuery": "",

"attributeQuery": "",

"enumerationQuery": ""

}

}

]

}

}