



sartorius stedim
biotech

User Guide

ODBC

SimApi



www.umetrics.com

© 1992-2018 Sartorius Stedim Data Analytics AB, all rights reserved

Information in this document is subject to change without notice and does not represent a commitment on part of Sartorius Stedim Data Analytics AB. The software, which includes information contained in any databases, described in this document is furnished under license agreement or nondisclosure agreement and may be used or copied only in accordance with the terms of the agreement. It is against the law to copy the software except as specifically allowed in the license or nondisclosure agreement. No part of this user guide may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording, for any purpose, without the express written permission of Sartorius Stedim Data Analytics AB.

Sartorius Stedim Data Analytics AB, patents and trademarks

US-7151976, US-7809450, US-8494798, US-8271103, US-8244498, US-9069345, US-8086327, US-8412356, US-8577480, US-8725469, US-9429939, US-9541471, US-7523384, US-7465417.

COMPUTER-IMPLEMENTED SYSTEMS AND METHODS FOR GENERATING GENERALIZED FRACTIONAL DESIGNS,
Pending

Trademark	Descriptor
Umetrics®	Suite of Data Analytics Solutions
MODDE®	Design of Experiments Solution
SIMCA®	Multivariate Data Analysis Solution
OPLS®	Method for improved regression analysis
O2PLS®	Method for data integration
O2PLS-DA®	Method for exhaustive discriminant analysis
OPLS-DA®	Method for group separation
PLS-TREE®	Top down clustering
S-PLOT®	Highlighting discriminatory variables
EZinfo®	Embedded Waters solution
VALUE FROM DATA®	We are value providers

ID #2080

Guide edition date: May 3, 2019



Sartorius Stedim Data Analytics AB

Twistevägen 48

SE-907 36 Umeå

Sweden

Phone: +46 (0)90 18 48 00

Contents

1	Introduction	4
1.1	Features	4
1.1.1	Synthetic process batch id	4
1.1.2	Generated synthetic process batch id tags; one per unique Unit ID	5
1.1.3	Batch node with filtering support.....	5
1.1.4	Batch Data Views.....	6
1.1.5	Discrete data	7
1.1.6	Synthetic batch age tags for discrete data nodes.....	7
2	Prerequisites.....	7
2.1	ODBC Drivers	7
2.2	Configure ODBC Data Sources.....	7
2.3	Database structure requirements	8
2.3.1	Data Source prerequisites	8
3	Installation and setup	8
3.1	Global connection settings	8
3.2	Direct Mode for continuous/process views.....	8
3.3	Lookup View Mode for continuous/process views	9
3.3.1	PDB views and HDB views.....	10
3.4	Batch node.....	11
3.5	Batch Data Views.....	12
3.5.1	Synthetic batch data instance tags	13
3.5.2	Synthetic process batch id tags filtered on column values.....	13
3.6	Discrete Node.....	13
3.6.1	Discrete Tag Definition View	14
3.6.2	Discrete data as seen by SIMCA-online.....	14
3.7	Some notes on SIMCA-online Write Back	15
3.8	XML Configuration File.....	15
4	Support.....	18

1 Introduction

This document is the user guide for the **ODBC SimApi** from Sartorius Stedim Data Analytics.

This SimApi connects to an ODBC (Open Database Connectivity) data source. For the SimApi to be able to interpret the data, the ODBC data source needs to be structured as described in this document (for example, there needs to be a Date/Time column).

For a detailed list of changes in different versions of this SimApi, see the **Version Info.txt** file that comes with the installation.

This SimApi can be used by Easy Analytics, SIMCA, or SIMCA-online.

For more information on available SimApis, see umetrics.com/simapi.

1.1 Features

- Handles SQL dialects for Microsoft SQL Server, Oracle, Microsoft Access, IBM db2, denodo, MySQL, PostgreSQL and standard SQL.
- ODBC connection string authentication with a user name and password or Windows authentication.
- Accessing multiple tables or views for reading continuous process data (current and historical). Two modes: Lookup view mode and Direct Mode (reading directly from a database view).
- Multiple batch nodes (with a single row for each batch with start- and end times).
- Reading batch data (batch conditions) from additional columns in batch nodes, or from **Batch Data Views** that store batch conditions in two columns (tag name and value).
- Discrete data support: reading discrete data from one node.
- Write back from SIMCA-online: historical process data and batch data to the ODBC data source.
- Supports both numerical and text data for tags.
- Works with numerical data or text (qualitative) data.
- **Synthetic process batch id** in the batch node, which can be used if the process data doesn't have a batch id tag. Can be used in phase conditions when the batch id is otherwise not available in the process data.
- **Synthetic process batch id tags** in Batch Data Views that can be filtered by other columns. Can be used in phase conditions when the batch id is otherwise not available in the process data.
- **Batch nodes with filtering support**. Filtering support is useful when you have a batch node with a master list of batches for your entire system, but want to be able to filter to show only batches in a part of the system (such as batches running in a single unit).
- **Synthetic batch data instance tags** handle multiple measurements of batch data per batch. Useful if you want be able to create batch level models that use two or more measurements of each batch condition variable per batch.
- Multiple instances of the ODBC SimApi to be configured and used from the same SIMCA-online server. This lets you connect to multiple databases on one or more database servers.
- Supports dates stored as local times or UTC in database.

Each feature (continuous, batch or discrete) can be configured and used on its own, and you don't need to configure all features if you don't intend to use them.

1.1.1 Synthetic process batch id

In a batch project, the process data has to have a batch id tag (column) that is specified on the **Phase conditions** section of the project configuration in SIMCA-online. This tag is matched against the batch node to know if a phase should execute.

If the process data doesn't have a batch id tag, the ODBC SimApi feature **synthetic process batch id** can be used. It generates the process data batch id using data from the batch node.

To use this feature, go to the **Phase conditions** section of the project configuration in the SIMCA-online client and configure the **Batch identifier tag** to be the batch id of **the batch node**. Whenever the server reads the batch id for the process data, the synthetic batch id from the batch node will be returned ensuring that the unit will execute.

Note that this feature does **not** work with concurrent (parallel) batches. Thus for any given time there must be only one batch active in the batch node.

Note: synthetic batch id tags generate an overhead in the SimApi which can lead to lower performance, especially during catch-up and repredict.

1.1.2 Generated synthetic process batch id tags; one per unique Unit ID

This feature builds on the synthetic process batch id described above, but uses an additional column in the batch node that contains the UnitID. The batch node is segmented into classes of batches that share the same value in the UnitID column. This can be seen as a way to filter out batches in the batch node that have a certain value for UnitID.

For each unique value in the UnitID column (looking in the entire batch node) the SimApi creates a synthetic tag in the batch node with the name BatchID_Unit_[Value].

For example: if values 1 and 2 are the two unique values in the UnitID column, it will result in two synthetic tags; BatchID_Unit_1 and BatchID_Unit_2. Reading process data from the synthetic batch id tag BatchID_Unit_1 will return only batch ids for batches whose UnitID column has the value 1. Batches with 2 in the UnitID column will be ignored.

To use this feature you configure the **Batch Identifier** tag in the **Phase Conditions** page for each unit to use the synthetic BatchID_Unit_[Value] tags.

The name of the unit id column is configured in the XML file.

Restrictions

- As for the regular synthetic batch id described above, by extension, this does not work with concurrent (parallel) batches sharing the same value of UnitID (concurrent batches with different UnitIDs works fine).
- All unit ids must be specified in the batch node before the SIMCA-online Server is started (the synthetic tags are created at startup). This could be done by adding one dummy batch in the batch node for each unit id that should be used.
- Batch ids still have to be unique in the batch node and there can be only one row for each batch id. As a consequence, each batch can only have one value of UnitID.
- The value of UnitID must not change for a specific batch during its lifetime.

Note; an alternative to using a synthetic process batch id filtered by UnitID like this is to use multiple batch nodes; one for each unique value of UnitID. In the case of multiple batch nodes, the same batch id can of course be present in many batch nodes (unlike when the UnitID filter is used) so in some cases multiple batch nodes might be the preferred solution.

1.1.3 Batch node with filtering support

A batch node is needed in SIMCA-online to execute batch projects.

Filtering support is useful when you have a batch node with a master list of batches for your entire system, but want to be able to filter to show only batches in a **part** of the system (such as batches running in a single unit).

Filtering is optional and to use it you must specify the column to filter on, and specify a list of valid values for that column. For each configured value, a separate node is created by the SimApi containing only batches with that value in the filter column.

Here is an example of a batch view with a Unit column that can be used to filter batches. The name of the view is BatchNode.

BatchID	StartTime	StopTime	Unit (Filter column)	Batch condition variable 1
Batch1	2017-01-01 01:00	2017-01-01 11:00	A	34
Batch2	2017-01-02 02:00	2017-01-02 12:00	B	4
Batch3	2017-01-03 03:00	2017-01-01 13:00	A	12
Batch4	2017-01-04 04:00	2017-01-04 14:00	A	4
Batch5	2017-01-05 05:00	2017-01-05 15:00	B	5

Configured for unit A and B this would give two batch nodes named BatchNode_A and BatchNode_B. Each of those batch nodes would only return batches that corresponds to their filter value so for BatchNode_A Batch1, Batch3 and Batch4 would be returned and for BatchNode_B Batch2 and Batch5 would be returned. The last column is an example of a batch condition variable that will be available in this batch node to read batch level data from.

Synthetic process batch ids are supported for batch nodes configured in this way.

1.1.4 Batch Data Views

A batch data view is a database view or table for storing batch conditions. It should have three columns: Batch identifier, Tag name and Value. One row in that view stores a value for a specific tag and batch. There will be many rows in this view for each batch when there are many batch condition variables.

Here's an example of a batch data view in the database with its three columns:

BatchID	Tag name	Value
Batch7	Yield	0.95
Batch7	ProductQuality	Excellent
Batch8	Yield	0.90
Batch8	ProductQuality	Poor

This batch data view will be exposed through the SimApi as two tags Yield and ProductQuality. Reading values for those tags for Batch8 would result in the values 0.90 and "Poor" respectively.

Note: If there are multiple rows for the same **BatchID** and **Tag name** combination in the database view, the SimApi will return the value from the last of those rows.

There are no batch start or batch end times columns in batch data views, so they cannot be used as batch nodes.

This SimApi supports many batch data views at once.

Tip: As an alternative to Batch Data Views, batch condition data can also be stored in a **batch node**. In that case one column is needed for each batch condition variable. Thus there will only be one row for each batch in batch nodes, but many more columns are needed.

Tip for SIMCA-online Extract functionality: To extract data from a batch data view you need to also include one tag (such as the Batch Identifier tag) from a batch node, so that SIMCA-online can know the batches to extract data for.

1.1.4.1 Synthetic batch data instance tags

Batch data always consists of a single observation per batch¹.

But what if the values of a batch data tag might change (for example because you rerun some measurement) and if you want to use **multiple** measurements per batch in a SIMCA-online batch level model?

Then you can use the optional feature **synthetic batch data instance tags**. These are tags that are added as additional tags in the batch data view, each mapping to a particular instance of the batch data for the tag.

Here's an example:

BatchID	TimeColumn	Tag name	Value
Batch7	2015-06-23 09:00	Yield	0.90
Batch7	2015-06-23 17:00	Yield	0.99

The SimApi will then expose this batch data view with two tags; **Yield_1** and **Yield_2**. When data is read for the batch Batch7 it will result in the values 0.90 for Yield_1 and 0.99 for Yield_2.

Notice that there is a new TimeColumn added in this example, with a timestamp for each row. This column is required for the synthetic batch data instance tag feature, in order for the SimApi to know how to order the values for the tags into the instance tags.

1.1.4.2 Synthetic process batch id tags filtered on column values

This feature is similar to the **Generated synthetic process batch id tags; one per unique Unit ID** with the difference that this is read from a batch data view and the batch id can be filtered on multiple columns.

Here's an example:

¹ For more information about the different data retrieval modes, of which batch data is one, see the SIMCA-online Technical Guide.

BatchID	TimeColumn	Tag name	Value	Unit	Line
Batch7	2015-06-23 09:00	Yield	0.90	A	1
Batch8	2015-06-23 09:00	Yield	0.95	B	2

If we would filter the BatchID on columns Unit and Line that would give us one synthetic batch id tag for each unique combination of the values from columns Unit and Line.

Synthetic tags created would be: BatchID_A_1, BatchID_B_2

Hence reading BatchID_A_1 for the time in the table would give the value Batch7, for the same time BatchID_B_2 would give the value Batch8.

Notice that a time column with a timestamp for each row is required for this feature.

1.1.5 Discrete data

Discrete data is infrequently measured data which have no logical values in between measurements. Usually a sample is taken on each batch at semi-regular intervals (such as once every day). This sample is then sent to a lab which performs analysis on the sample and at a later stage returned with a report on the sample for the required variables. This is then entered in the database in the discrete data table.

1.1.6 Synthetic batch age tags for discrete data nodes

For discrete data nodes there are four synthetic tags named \$BatchAge(d), \$BatchAge(h), \$BatchAge(m), \$BatchAge(s). When reading their values they will return the batch age as a floating point number for each sample in four different magnitudes: days, hours, minutes and seconds respectively. These tags can be used as maturity in the SIMCA model, reducing the need to explicitly add and populate such tags to the discrete data tables.

2 Prerequisites

For this SimApi to work there are some requirements:

2.1 ODBC Drivers

In order for the ODBC SimApi to work with your database you must install ODBC drivers for your database. You obtain drivers from the manufacturer of the database. Drivers for Microsoft SQL Server are often installed on most Windows computers.

2.2 Configure ODBC Data Sources

The data source should be configured as a System DSN in the ODBC Data Sources control panel in Windows.

Note that there are two versions of this tool on 64-bit Windows: one for 32-bit applications and one for 64-bit².

In the ODBC configuration wizard for SQL Server you can select how to authenticate users (log in):

- **SQL Server Authentication** which means that the database is authenticating users using a user name and password. This username and password can be given in the ODBC wizard, or in the configuration file by filling in the credentials dialog as part of the configuration (see below).
- **Windows authentication.** Windows authentication means that the user account (typically in Active Directory) of the program using the SimApi is used to connect to the database. This user account needs to have appropriate rights in the database.

Windows authentication with a SIMCA-online server means that the service account of the SIMCA-online Server service is used (LocalSystem by default, but can be changed to a specific account by the administrator, see The SIMCA-online Technical Guide for more information).

Windows Authentication with SIMCA means the user account of the person that is running SIMCA will be used.

² On 64-bit Windows you can start the 32-bit ODBC Data Sources program by launching it manually from the SysWow64-folder, typically C:\Windows\SysWOW64. If you just start ODBC Data Sources from the start menu in 64-bit Windows 7 it will launch the 64-bit version.

Verify the connection to the database with the Test Data Source button at the end of the ODBC configuration wizard. As noted above, for Window Authentication this tests with your own user account, and not necessarily with the SIMCA-online service account.

2.3 Database structure requirements

A database can of course contain almost any data with an arbitrary structure. The SimApi however assumes it can read current or historical data for a limited number of tags³.

Thus, for the ODBC SimApi to work with a database, a certain predefined structure needs to be applied to the database. In many cases this can be done by creating a set of database views that re-arrange existing data into the required structure.

2.3.1 Data Source prerequisites

The ODBC SimApi requires that the database has a certain structure in order for the SimApi to be able to use it. This section and its subsections describes how the database views must look like depending on if you want to use continuous/process data, batch data or discrete data. You do not need to configure all features, only the ones you plan to use. Refer to the sections that apply.

There are two different ways to configure how continuous/process data should be accessed:

1. **Direct Mode.** In this mode you specify the view names that should be exposed through the SimApi. Each view becomes a node in the SimApi, and all columns a view become tags. Direct Mode is recommended since it is relatively simple to use.
2. **Lookup View Mode.** This mode is more complex and requires a specific view in the database called a Lookup View. This view defines the tags that should be exposed through the SimApi, but data is taken from additional related views.

Notes:

- When the term database *view* is mentioned above it refers to a database view that aggregates data from existing tables. If you like you can use database tables instead of views, but using a view in a database is a convenient way to filter or aggregate data into a form that is suitable for the SimApi.
- It is important that all views or tables used has a unique primary key and indexing is used on date/time columns so that the performance of the queries by the SimApi won't suffer.
- The SimApi supports two data types for data columns; either numerical real values (a float or other numerical datatype) or text strings (varchar in the database). Note that nvarchar is not supported. Missing values (nulls) are also allowed for data columns.

The sub sections describe how to set up the SimApi and the database views in detail.

3 Installation and setup

Refer to the **SimApi Overview User Guide** located at umetrics.com/simapi for how to install and setup this SimApi.

3.1 Global connection settings

The first section of the XML configuration file is the connection settings that describe how to connect to your database.

Required settings are:

- DSN – the data source name. It should match the name of the System DSN in Windows ODBC Data Sources that you created in 2.2.
- SQLDialect – set to match the database server you are using.

You may also need to set the LeftPunctuation and RightPunctuation settings if you use reserved SQL keywords as names of identifiers, or use spaces in identifier names of your views/tables/columns in the database.

The other connection settings are optional.

3.2 Direct Mode for continuous/process views

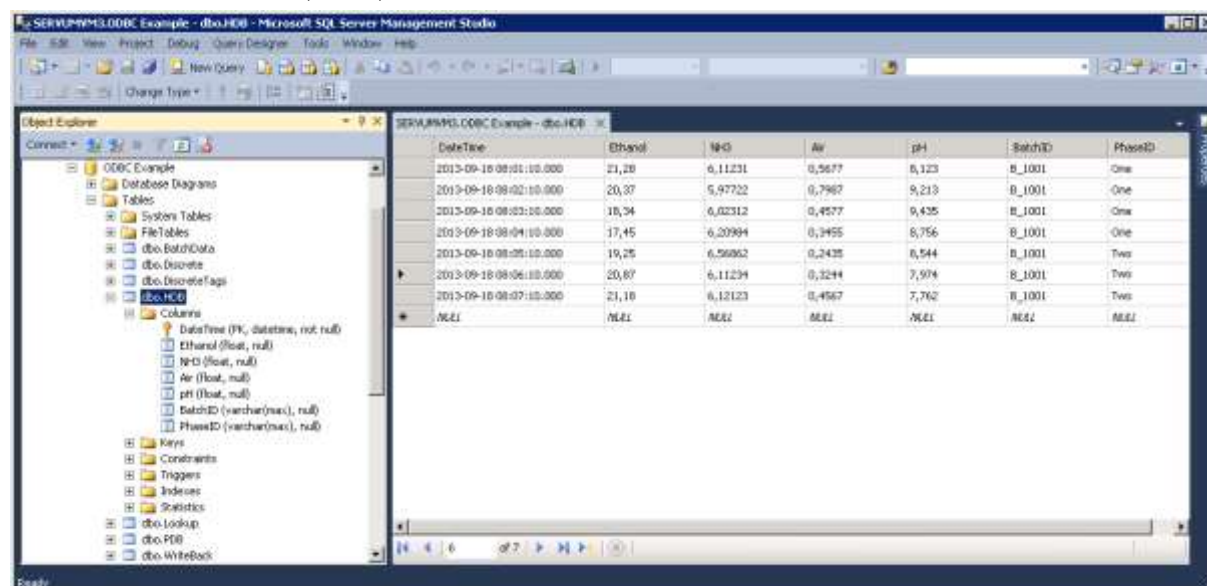
Each row in a Direct Mode view is an observation.

The columns in the view represent variables:

³ For more information about SimApis, see the SIMCA-online Technical Guide at <http://www.umetrics.com/kb/simca-online-technical-guide>.

- There must be a single column with data/time data. This column should be the primary key, and cannot hold null values. The name of this column should match the configured TimeField setting in the configuration file. Values for this column are the time stamp for the observations.
- The remaining columns will be exposed through the SimApi as tags with the same names as the column names. Values in these columns are process data.
- Write back is supported for all tags, however the correct permissions must be set in the ODBC data source.

Note; the SimApi enumerates the columns at startup only. This means that if new columns are added to a view the SimApi must be restarted in order for the SimApi to expose them.



DateTime	Ethanol	pH	Air	BatchID	PhaseID
2013-09-18 08:01:10.000	21,28	6,11231	0,5677	8_1001	One
2013-09-18 08:02:10.000	20,37	6,97722	0,7987	8_1001	One
2013-09-18 08:03:10.000	18,34	6,02312	0,4577	8_1001	One
2013-09-18 08:04:10.000	17,45	6,20994	0,3455	8_1001	One
2013-09-18 08:05:10.000	19,25	6,56962	0,2435	8_1001	Two
2013-09-18 08:06:10.000	20,87	6,11234	0,3244	8_1001	Two
2013-09-18 08:07:10.000	21,18	6,12123	0,4567	8_1001	Two
NULL	NULL	NULL	NULL	NULL	NULL

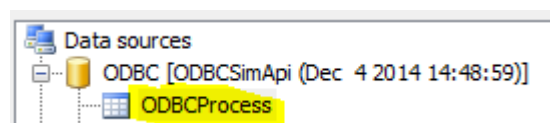
Figure 1. Direct Mode table example.

3.3 Lookup View Mode for continuous/process views

This view can be used instead of, or in addition to, using Direct Mode views.

The Lookup View is an indirect way of specifying which data columns in **other views that** should be compiled and exposed through the SimApi. The other views are either HDB sources (historical data) or PDB sources (current data) respectively. The PDB source is optional as you will see below.

The data exposed by the Lookup View is presented as a node called **ODBCProcess**:



Each row in the Lookup view defines one tag to be exposed through the SimApi and which other views to take the data from for that tag. The Lookup view thus will contain many rows. It also specifies if a tag is writeable, i.e. if SIMCA-online should be able to write values back to this tag.

There can be multiple PDB sources and multiple HDB sources in use from the Lookup View. Thus the ODBC SimApi can aggregate data from multiple views into one node with tags that are exposed through the SimApi.

The Lookup view should contain the following columns (referred to as Fields in the configuration file):

- Name – The name of a tag (Primary Key, varchar, not null).
- PDB_Source – The name of the view that contains the current data for the tag (varchar). If this column contains an empty value, the program will read all data from the HDB_Source.
- PDB_Field – The column name of the tag in the PDB Source view (varchar). If the PDB_Source value is omitted, this column will not be read.
- HDB_Source – The name of the view that contains the historical data for the tag (varchar).
- HDB_Field – The column name of the tag in the HDB Source view (varchar).
- Writeable – If SIMCA-online should be able to write data to this tag or not (bit).

Note that for each column, the above description also states which rows should be primary key and the data type for each column. The **names** of the columns can be arbitrary since the names are specified in the configuration file.

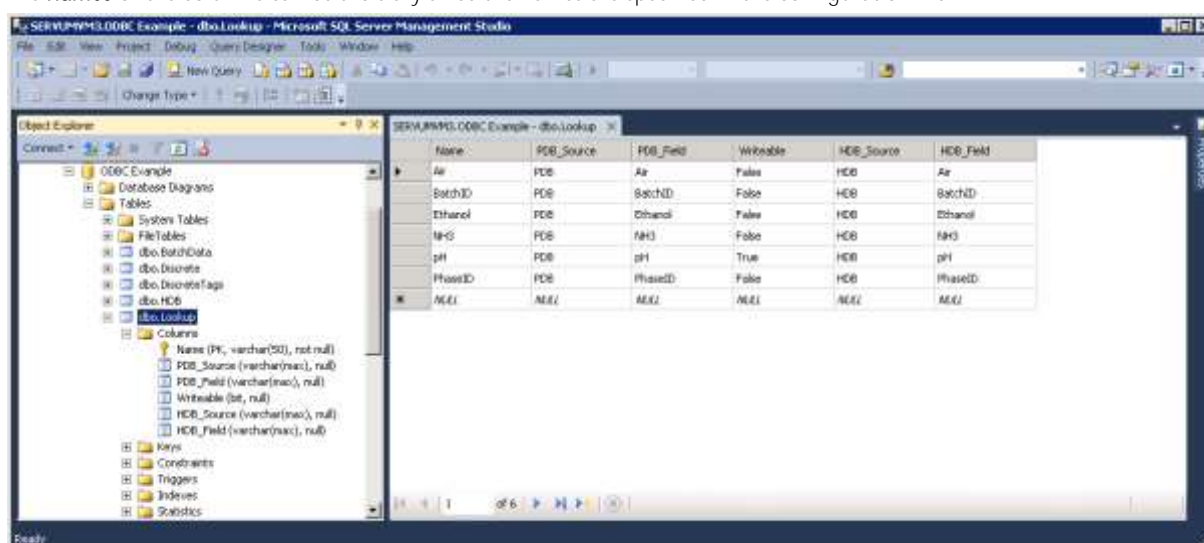


Figure 2. Example of a Lookup view (in the form of a database table). In the screenshot you'll see that one PDB source and one HDB source are used. The Name column determines the tag names that the SimApi will use, and in this case the _Field columns use the same column names. Only one tag is writable.

3.3.1 PDB views and HDB views

When you use a Lookup View you also need at least one HDB view. The views for the historical data (HDB) and the optional views for current data PDB both have the same data structure (columns).

Each row in the PDB or HDB views represents an observation with values for each tag in that PDB/HDB view as specified in the lookup view.

The differences between PDB and HDB are:

- A PDB view contains only one row of data for the tags specified in the Lookup view. It should also have a time stamp column for when the table was last updated.
- A HDB view contains several rows of data for the tags specified in the Lookup table. Each row has a time stamp containing the historical timestamp for a particular observation.

The PDB and HDB views should have the following columns,

- DateTime – For a PDB: The time when the table was last updated (Primary key, datetime, not null).
– For a HDB: The historical time for the tag values (Primary key, datetime, not null).
- [Column name] – There should be one column for each tag that was specified in the Lookup table. Contains the data for the tag in each row.

Note that for each column, the above description also states which rows should be primary key and the data type for each column.

The **names** of the columns are arbitrary in the database since the names are specified in the configuration file.

The following columns are not mandatory, but are useful to add if batches are modelled with multiple phases and there are several units in the process:

- UnitBatchID – One column per unit that contains the batch ID within a certain unit (varchar). This tag can be used in the **Batch identifier tag** field for that particular unit in the Phase conditions section of the configuration of this project in SIMCA-online.
- PhaseID – One column per unit that holds the phase info for the unit (int, float or varchar). This tag can be used in logical expression in the **Phase condition** field in the Phase conditions section of the configuration of this project in SIMCA-online.

The maximum allowed number of tags (columns) fetched is 255.

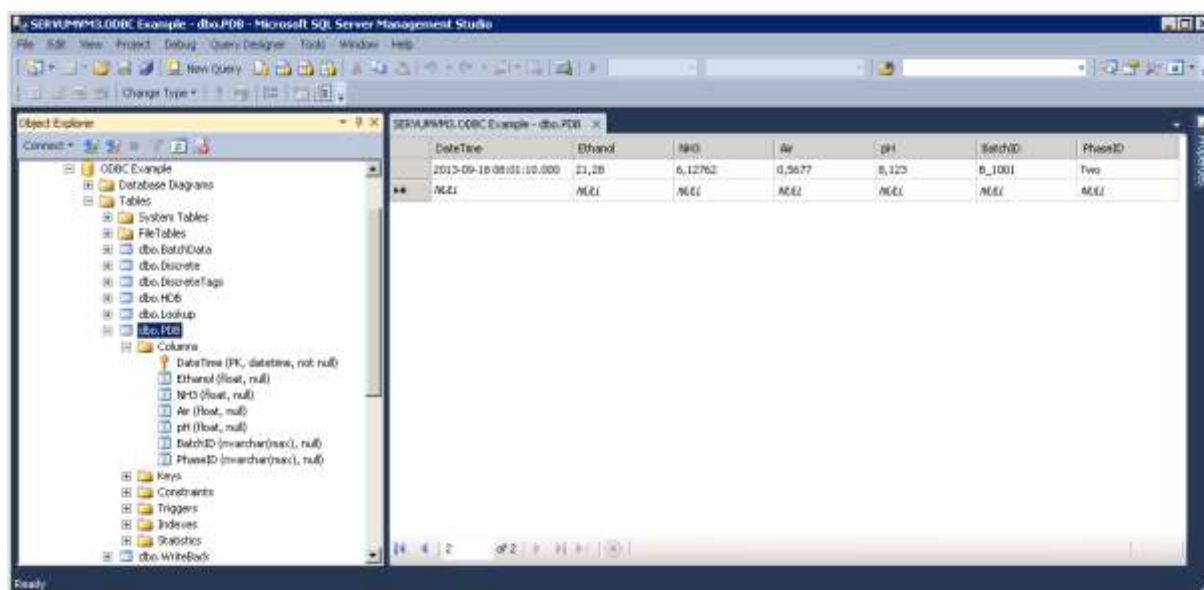


Figure 4. PDB example.

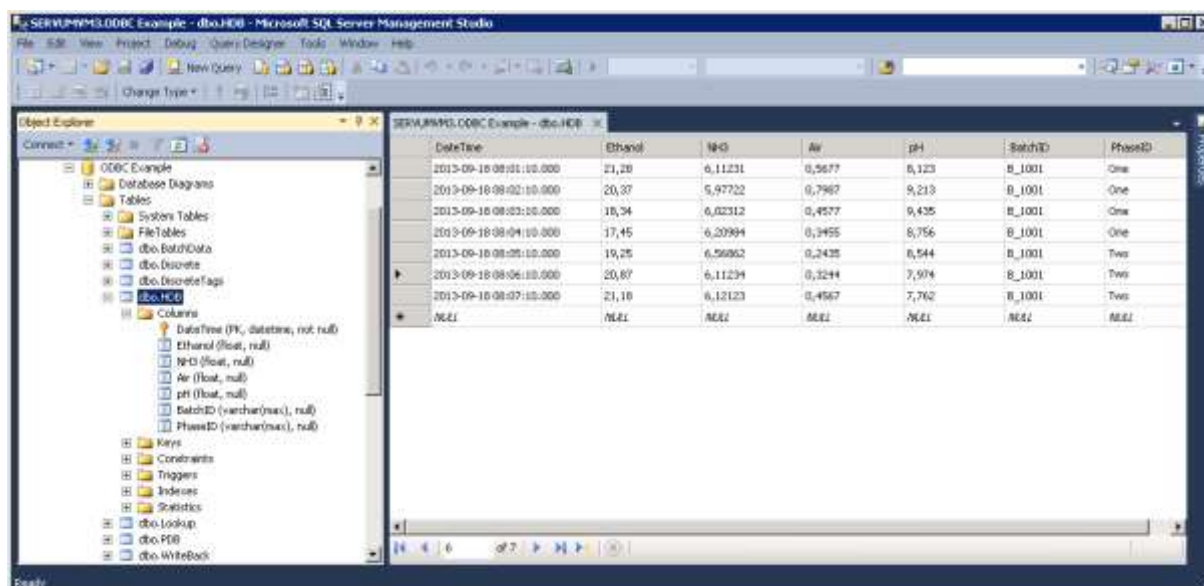


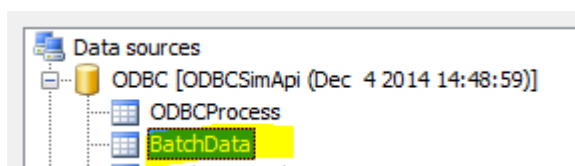
Figure 3. HDB example.

3.4 Batch node

Batch nodes are optional.

A batch node contains meta-information about batches such as start time, stop time, and also optionally batch conditions. A batch node is required by SIMCA-online to analyze batch data.

The name of a batch node seen from SIMCA-online or SIMCA is the original name of the view or table in the database, in this example "BatchData":



Each row in a batch node represents one batch.

A batch view needs to have the following columns:

- BatchID – The name of the batch (Primary key, varchar, not null).
- BatchStart – The start time of the batch when it first enters the entire process (not when it starts in a unit (part of) in the process) (datetime, not null).
- BatchStop – The time when the whole batch is completed in the system (not in a unit), null if not completed (datetime).

Note that the BatchID column should be the primary key.

In addition there can be optional columns as follows, for each batch condition variable:

- [Batch condition name] – One column for each batch condition. The column name will be used as tag name. (float for numerical values or varchar for text such as the configuration id).
- UnitID – The name of the unit to which the batch is associated with (varchar).

The **names** of the batch node and columns are arbitrary in the database since the names are specified in the configuration file.

The screenshot shows the Microsoft SQL Server Enterprise Manager interface. On the left, the 'Object Explorer' displays the database structure for 'ODBC Example', including tables, columns, and keys. The 'Columns' list shows: BatchID (PK, varchar(255), not null), BatchStart (datetime, not null), BatchStop (datetime, null), bc1 (float, null), bc2 (float, null), and UnitID (varchar(255), null). On the right, a query window displays the data from the 'BatchData' table. The table has columns: BatchID, BatchStart, BatchStop, bc1, bc2, and UnitID. The data is as follows:

BatchID	BatchStart	BatchStop	bc1	bc2	UnitID
B_1016	2013-12-07 11:09:35.000	2013-12-17 12:05:50.000	1.8	109	1
B_1017	2013-12-12 09:36:36.000	2013-12-22 10:01:40.000	1.8	115	2
B_1018	2013-12-17 12:08:53.000	2013-12-27 12:41:24.000	2	100	1
B_1019	2013-12-22 10:02:29.000	2014-01-01 10:59:57.000	2	109	2
B_1020	2013-12-27 12:42:24.000	2014-01-06 12:54:19.000	1.8	117	1
B_1021	2014-01-01 10:59:59.000	2014-01-11 11:05:15.000	2	114	2
B_1022	2014-01-06 12:00:10.000	2014-01-16 13:10:30.000	1.2	112	1
B_1023	2014-01-11 11:05:46.000	2014-01-23 11:59:13.000	2.1	102	2
B_1024	2014-01-16 13:10:48.000	2014-01-26 13:26:11.000	1.5	103	1
B_1025	2014-01-21 12:00:10.000	2014-01-31 12:59:51.000	1.8	101	2
B_1026	2014-01-26 13:26:19.000	2014-02-05 14:03:13.000	2	112	1
B_1027	2014-01-31 12:00:51.000	2014-02-10 13:44:33.000	1.2	119	2
B_1028	2014-02-05 14:03:30.000	2014-02-15 14:06:39.000	1.4	114	1
B_1029	2014-02-10 13:44:37.000	2014-02-20 14:06:13.000	1.7	110	2
B_1030	2014-02-15 14:06:57.000	NULL	2	NULL	1
B3	2014-10-09 12:53:49.000	2014-10-09 20:53:49.000	2	NULL	1
NULL	NULL	NULL	NULL	NULL	NULL

Figure 5. Batch node example with two batch conditions (bc1 and bc2) and a UnitID column.

3.5 Batch Data Views

Each batch data view must have these three columns (additional columns will be ignored):

- BatchID – name of the batch (varchar, not null)
- Tag name – name of the batch condition variable (varchar, not null)
- Value – value of the batch condition variable (**float** for numerical values, or **varchar** for text or float⁴).

The combination of BatchID and Tag name should be the primary key (unless you want to use the multiple batch data instance feature).

Each batch data view is exposed as a node by the SimApi. The name of the node is the view name in the database. The view name and the column names are configured in the XML configuration file using the attributes of a single BatchDataView element like this:

```
<BatchDataView ViewName="DatabaseViewOrTableName" BatchIDColumn="BatchID" TagNameColumn="Tag name" ValueColumn="Value" />
```

Note that the values used here matches the table in Figure 5.

Add multiple batch data views by adding more BatchDataView elements.

⁴ By using a varchar text column you can store text (data for qualitative variables in a SIMCA project). However you can also store numerical numbers in text format, and the SimApi will convert these to numbers. This way you can have some tags that are numerical and some that contain text.

3.5.1 Synthetic batch data instance tags

To configure the optional batch data instance tags, you need to add the attributes **NumSyntheticBatchTags** and **TimeColumn** to the **BatchDataView** element:

```
<BatchDataView ViewName="DatabaseViewOrTableName" BatchIDColumn="BatchID" TagNameColumn="Tag
name" ValueColumn="Value" TimeColumn="TimeColumn" NumSyntheticBatchTags="3" />
```

TimeColumn should be the name of the time column in your database view. This column must be provided for batch data instance tags.

Allowed value for **NumSyntheticBatchTags** are numerical values between 1 and 10. This controls how many synthetic instance tags are created per real tag. For example; for the tag "tag" new tags "tag_1", "tag_2", ... "tag_N" will be created until N= **NumSyntheticBatchTags**.

3.5.2 Synthetic process batch id tags filtered on column values

To configure the optional batch id filter tags that can be used for continuous data retrieval mode, you need to add the attributes **FilterColumns** and **TimeColumn** to the **BatchDataView** element:

```
<BatchDataView ViewName="DatabaseViewOrTableName" BatchIDColumn="BatchID" TagNameColumn="Tag
name" ValueColumn="Value" FilterColumns="Column1|Column2" TimeColumn="TimeColumn"/>
```

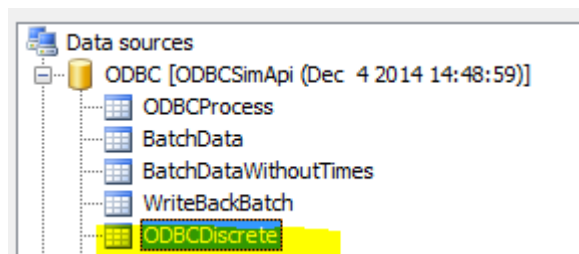
FilterColumns should be one or more column names in your database view. If several column names are used separate them with the pipe character (|).

TimeColumn should be the name of the time column in your database view. This column must be provided for batch data instance tags.

3.6 Discrete Node

The optional discrete node contains discrete data measurements. A discrete node is required by SIMCA-online if it should be able to analyze discrete data.

There can be only one discrete node. It is exposed as **ODBCDiscrete** through the SimApi:



Each row in a discrete node represents one measurement for a batch and tag at a given time.

A discrete node needs to have the following columns:

- BatchID – The name of the batch (varchar, not null).
- TagName – The name of the tag (varchar, not null).
- Time – The time when the sample was taken (datetime, not null).
- Value – The measurement value (float). Discrete data cannot be string data.

Note that the combination of BatchID+TagName+Time should be the primary key.

The **names** of the discrete node and columns are arbitrary in the database since the names are specified in the configuration file.

Figure 6 shows a screenshot of Microsoft SQL Server Management Studio. The 'Object Explorer' on the left shows the database structure, including 'dbo.Discrete'. The main window displays the data from the 'dbo.Discrete' table. The table has four columns: BatchID, TagName, Time, and Value. The data is organized by BatchID, with each batch having multiple rows for different tags and their corresponding values at specific times.

BatchID	TagName	Time	Value
B_1027	CO	2014-02-03 13:40:15.000	16,9
B_1028	pH	2014-02-06 14:28:40.000	4
B_1028	pH	2014-02-07 14:17:02.000	4,9
B_1028	pH	2014-02-08 14:50:49.000	4,2
B_1028	CO	2014-02-06 14:28:40.000	17
B_1028	CO	2014-02-07 14:17:02.000	16,2
B_1028	CO	2014-02-08 14:50:49.000	13,8
B_1029	pH	2014-02-11 14:02:14.000	4,6
B_1029	pH	2014-02-12 13:54:24.000	4,2
B_1029	pH	2014-02-13 14:05:07.000	4,1
B_1029	CO	2014-02-11 14:02:14.000	12,8
B_1029	CO	2014-02-12 13:54:24.000	10,8
B_1029	CO	2014-02-13 14:05:07.000	14,1
B_1030	pH	2014-02-16 14:44:22.000	4,5
B_1030	pH	2014-02-17 15:01:50.000	4,6
B_1030	pH	2014-02-18 14:59:49.000	4,8
NE21	NE21	NE21	NE21

Figure 6. Discrete node example with two tags sampled three times (at roughly 24 hour intervals) per batch.

3.6.1 Discrete Tag Definition View

This optional view is used by the SimApi to enumerate the discrete tags that should be available through the SimApi. This happens at SimApi startup.

You may want to use this view for performance optimizations when loading the SimApi or if you want to control what tags are exposed from the SimApi. For instance if you want to expose tags that doesn't have any measurements yet when the SimApi is started.

Figure 7 shows a screenshot of Microsoft SQL Server Management Studio. The 'Object Explorer' on the left shows the database structure, including 'dbo.DiscreteTags'. The main window displays the data from the 'dbo.DiscreteTags' table. The table has two columns: TagName and DiscreteTags. The data is organized by TagName, with each tag having a corresponding value in the DiscreteTags column.

TagName	DiscreteTags
CO	
pH	
pH2	

Figure 7. Simple discrete tag lookup view which use the discrete table.

3.6.2 Discrete data as seen by SIMCA-online

The following is how the discrete data in **Error! Reference source not found.** will look in SIMCA-online when combined with the batches in **Error! Reference source not found.**

	1	2	3	4	5	6	7	8
	BATCH	OBSERVATION	\$BatchAge(d)	\$BatchAge(h)	\$BatchAge(m)	\$BatchAge(s)	CO	pH
76	B_1024	2	3,03539	72,8494	4370,97	262258	17,9	4,7
77	B_1025	0	1,02872	24,6892	1481,35	88881	10,8	4,3
78	B_1025	1	2,03596	48,8631	2931,78	175907	11,3	4,1
79	B_1025	2	3,02333	72,56	4353,6	261216	14,3	4,6
80	B_1026	0	1,02106	24,5056	1470,33	88220	10,5	4,9
81	B_1026	1	2,02056	48,4933	2909,6	174576	16,8	4,8
82	B_1026	2	3,04142	72,9942	4379,65	262779	16	4,6
83	B_1027	0	1,04109	24,9861	1499,17	89950	13,7	4,1
84	B_1027	1	2,00284	48,0681	2884,08	173045	13,2	4,6
85	B_1027	2	3,02736	72,6567	4359,4	261564	16,9	4,2
86	B_1028	0	1,01748	24,4194	1465,17	87910	17	4
87	B_1028	1	2,0094	48,2256	2893,53	173612	16,2	4,9
88	B_1028	2	3,03286	72,7886	4367,32	262039	13,8	4,2
89	B_1029	0	1,01223	24,2936	1457,62	87457	12,8	4,6
90	B_1029	1	2,00679	48,1631	2889,78	173387	10,8	4,2
91	B_1029	2	3,01424	72,3417	4340,5	260430	14,1	4,1
92	B_1030	0	1,02598	24,6236	1477,42	88645	16,9	4,5
93	B_1030	1	2,03811	48,9147	2934,88	176093	11,1	4,6
94	B_1030	2	3,00894	72,2144	4332,87	259972	20	4,8

Figure 8. Discrete data as seen by SIMCA-online. Note that the generated batch age tags represent the age of the batch for each observation/sample.

3.7 Some notes on SIMCA-online Write Back

Write back in SIMCA-online can be used to write data from SIMCA-online into the ODBC data source. See the SIMCA-online Technical guide for more details.

Writing continuous process data (from continuous configurations, the batch evolution level or from Control Advisor) and batch data (from the batch level) are supported.

It is not recommended to write back to the same nodes that you are using to read data, because this would attempt to add duplicate rows with the same primary keys in the database views because of the primary keys we recommend on the date/time-column and batch ID columns (see above).

Instead create one or more Direct Mode views for continuous / evolution data, or batch nodes for batch data to use for write-back.

The reason for this issue is that the ODBC SimApi uses SQL INSERT statements to add a new row of data for each observation or for each batch at the batch level.

- For continuous data the time of the observation is written back together with the values from SIMCA-online.
- For batch data the batch ID is written back together with the values from SIMCA-online.

Since the time of an observation or batch id of a batch is written-back this would violate the primary key constraint in the databases if the same values already were present (as they would be if data were read from the same nodes).

Other than these primary key differences, the same database schema applies to nodes for write back (see above for more information):

- A date/time column should exist for continuous nodes, and a batch id column for batch nodes.
- Add one data column for each tag that should be available for write back. Use the float datatype for numeric data, and varchar for text data (such as when writing back the configuration id of a configuration). For example: if you plan to write back 20 different data vectors from SIMCA-online you need to add 20 data columns to the database view.

3.8 XML Configuration File

The XML configuration file is a text file, it can be edited with for example Notepad. It has the following settings:

Connection specific settings	Explanation
------------------------------	-------------

DSN	Data Source Name as set up in the Windows ODBC Administrator control panel.
SQLDialect	The SQL dialect to use. One these values: standard , postgresql , db2 , mssql , mysql , oracle , access , denodo . If left blank then standard will be used, but the default for a new XML-file is mssql.
Credentials	Stored the ODBC user name and password in an encrypted form. Use the Configure button to specify the user name and password.
QueryTimeout	The time before a query or connection to the database will time out and fail.
DBSchema	The database schema in the database (if applicable). A database schema is a way to logically group objects such as tables, views, stored procedures etc. Think of a schema as a container of objects. You can assign a user login permissions to a single schema so that the user can only access the objects they are authorized to access. Schemas can be created and altered in a database, and users can be granted access to a schema. A schema can be owned by any user, and schema ownership is transferable.
LeftPunctuation	SQL dialect specific left delimiter used to separate identifiers from other SQL commands. The default is empty which means that no left punctuation is used. You need to specify a non-empty value if the identifiers in the database use spaces or reserved SQL keywords. For SQL Server or Access you use "[", but for Oracle and other databases using the SQL standard you should set this setting to """ (this is the XML escape sequence for the double quotation mark ").
RightPunctuation	SQL dialect specific right delimiter used to separate identifiers from SQL commands. For SQL Server or Access you use "]", but for Oracle and other databases using the SQL standard you should set this setting to """ as for LeftPunctuation above.
SelectStatementTerminator	SQL dialect specific. Character to use to terminate select-statements. For some dialects and versions this need to be set to an empty string "". Default is ";".
Database specific settings	
UseLocalTime	Specifies if dates are stored as local time or UTC in database. Default is "1" meaning local time is used.
Direct mode specific settings	
Tables	The name of the views that contains continuous/process data. Multiple views can be specified by separating their names with a pipe character (). For example: Table1 Table2 Table3 View1. The TimeField name has to be identical in all views.
Loopup view specific settings	
LookupTable	The name of the lookup view or table.
TagNameField	The column name where the tag names are given.
PDBTableField	The name of a column in the lookup view. For each row this column holds a name of a PDB view. The name of a view with PDB data. Can be left blank, if so the most recent row of the HDB will be used instead for current data.
PDBTagField	The column name in the PDB table where data for the tag can be found (not used if PDBTableField is omitted).
HDBTableField	The name of a column in the lookup view. For each row this column holds a name of a HDB view.

HDBTagField	The name of a column in the lookup view. For each row this column holds a name of a tag in the HDB view.
WriteableField	The column name that tells if the tag is writeable or not.
Direct mode and HDB/PDB view specific settings	
TimeField	The name of the date/time column in the Continuous/Process View and the PDB- or HDB-views (or tables).
Batch node specific settings	
BatchTable	The name of the view or table that contains the batch data. Multiple batch view can be specified by separating their names with a pipe character (). For example: BT1 BT2 BT3. The following columns have to be identical in all views.
BatchIDField	The columns name of the batch ID in the batch node.
StartTimeField	The column name of the start time for the batch.
StopTimeField	The column name of the stop time for the batch.
BatchIDUnitField	The column name of the unit ID in the batch node. This field can be used to generate synthetic batch id process tags filtered by unit id.
Batch node with filtering support settings	
ViewName	The name of the view or table that contains the batch data.
BatchIDColumn	The column name of the batch ID of the batch.
StartTimeColumn	The column name of the start time for the batch.
StopTimeColumn	The column name of the stop time for the batch.
FilterColumn	The name of the column that contains the filter values in the batch node. This setting can be left empty if you don't need the filtering functionality.
FilterValues	<p>The values that should be used to filter batches. Multiple filter values are separated by the pipe character (). Leave empty if you don't need filtering. For example: FilterValue1 FilterValue2 FilterValue3.</p> <p>This is an example of how it could look like in the configuration file:</p> <pre><BatchNode ViewName="BatchNode" BatchIDColumn="BatchID" StartTimeColumn="StartTime" StopTimeColumn="StopTime" FilterColumn="Unit FilterValues="FilterValue1 FilterValue2" /></pre> <p>Multiple <BatchNode> elements are supported. Note that you can have different names of the columns between batch nodes.</p>
Batch data view settings	Stored in one or more <i><BatchDataView></i> elements.
Discrete view specific settings	
DiscreteTable	The name of the view/table that contains the discrete data.

DiscreteLookupTable	The name of the view/table that defines the discrete tags to use. If left blank the DiscreteTable will be analyzed at startup to enumerate all tags there.
DiscreteTimeField	The column name of the time of the measurement.
DiscreteBatchIDField	The column name of the identity of the batch that was measured.
DiscreteTagNameField	The column name of the name of the tag that was measured.
DiscreteValueField	The column name of the value of the measurement.
Log file specific settings	
LogFileSize	The maximum allowed size of the log file before the file is truncated.
LogLevel	The higher the value the more information is printed to the log file. Maximum value is 4 and minimum value is 0. (0=Critical, 1=Error, 2=Warning, 3=Information, 4=Debug).

4 Support

umetrics_support-SIMCA-online@sartorius-stedin.com