ODBC SimApi User Guide

October 24, 2024

Contents

1 Introduction

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

A SimApi is the connection between the Umetrics® Suite and external data sources.

This SimApi connects to an ODBC (Open Database Connectivity) data source such as a relational database. The data source must be structured as described in this document.

To use a SimApi in SIMCA-online which is used for real-time monitoring, it is important that the data source behaves as a good process data historian: There must be no data acquisition delays in the data source: current data in the data source should reflect the current state of the process to work well in SIMCA-online. The data source must also respond quickly to requests for data so that the server will be able to keep up with real-time execution of projects.

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 SIMCA or SIMCA-online or other software that can use SimApis.

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

1.1 Features

- Supports SQL dialects for Microsoft SQL Server, Oracle, Microsoft Access, IBM db2, Denodo, MySQL, PostgreSQL and standard SQL.
- ODBC connection string authentication with a provided username and password, Windows authentication or the credentials specified in the ODBC connection in Windows ODBC Data Sources.
- 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, defining the lifetime of batches. Contains 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 a narrow table with only two columns (tag name and value).
- Discrete data support: reading discrete data one or more nodes, that may have different column names.
- Write back from SIMCA-online: historical process data and batch data.
- Works with numerical data or text (qualitative) data.
- Synthetic process batch id in the batch node, which can be used in SIMCA-online phase execution conditions, if the process data doesn't have a batch id tag.
- 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 it to show only batches in a part of the system (such as batches running in a single unit, or a batch of a specific material or type).
- Synthetic batch data instance tags handle multiple measurements of batch data per batch. Useful if you want to 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.
- Connection resiliency – the SimApi automatically reconnects to a data source after it has been disconnected (for example after a network glitch).
- Thread-safe concurrent access. The SimApi works with the Concurrent SimApi access feature introduced with the SIMCA-online 18 server which can improve performance and responsiveness.
- 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 all features are optional. You need not configure features unless you need them.

1.1.1 Synthetic process batch id

In a batch project, the process data must have a batch id tag (column) that is specified on the **Execution conditions** page in 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 **Execution conditions** page 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.

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 acts like a for 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, you configure the **Batch identifier** tag in the **Execution 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 SimApi is started (the synthetic tags are created at startup). Tip: you can prepopulate a table with one dummy batch in the batch node for each unit id that will be required.
- Batch ids must be unique in the batch node and there can be only one row for each batch id. Therefore, 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 are the preferred solution.

1.1.3 Batch node with filtering support

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

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

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, called BatchNode in the database, with a Unit column that can be used to filter batches.

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 values for 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.

You can configure many different batch data views.

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 more columns are needed.

Important for SIMCA-online's 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 batch[1].

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, 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 like 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:

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 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 a 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.

Learn more on this in the SimApi Guide and SIMCA-online Technical Guide.

[1] For more information about the different data retrieval modes, of which batch data is one, see the SimApi Guide.pdf.

1.1.5.1 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 be 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.

1.1.6 Concurrent SimApi Access

Concurrent SimApi access is an optional feature introduced with the SIMCA-online 18 server which can improve performance and responsiveness of a server and clients.

The ODBC SimApi is thread-safe and uses a pool of connections to the server. The size of the pool is configurable using the DatabaseConnectionPoolSize setting in the configuration file (defaults to 10). This controls how many concurrent requests to the database are allowed. Additional concurrent requests are queued and performed as earlier requests finish.

Different database engines and different database servers may support a different number of concurrent threads. For optimal performance, the connection pool size may have to be adjusted.

Learn more considerations and how to enable this in the SIMCA-online help on Concurrent SimApi Access.

2 Prerequisites

For this SimApi to work there are requirements that need to be fulfilled, both for the PC running the SimApi and for the data source itself relating to database structure and performance.

2.1 Database structure requirements

A database can of course contain almost any data with an arbitrary structure. The ODBC SimApi is built for obtaining process data for use in SIMCA-online and SIMCA.

For the SimApi to work the database needs to have a particular structure or design as described below.

Here are general requirements:

- All tables used has a unique primary key.
- Most tables in the database require a date/time columns so that the SimApi can identify the timestamp for each row of data. This column should be indexed in the database so that performance 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 (for example varchar in the database). Missing values (nulls) are also allowed for data columns.
- The term view and table are used interchangeably in this document to mean the same thing.
- If you create a view in the data source and want to use that in the SimApi, be careful to not introduce performance issues: the view must be quick to access by the SimApi. This means you typically cannot create a complex view that aggregates data from many other tables and performs complicated business logic to compute the results. Such a view won't fulfil the performance requirements of the users of the SimApi such as SIMCA-online used for real-time monitoring of a process.

The SimApi has many features, and each feature has various required settings that needs to be made, as described in chapter 3.

2.2 Database performance considerations

To use a SimApi in SIMCA-online which is used for real-time monitoring, it is important that the data source behaves as a good process data historian.

- There must be no data acquisition delays in the data source: current data in the data source should reflect the current state of the process to work well in SIMCA-online.

- The data source must also respond quickly to requests for data so that the server will be able to keep up with real-time execution of projects.

This has several implications:

- Data warehouses or databases that perform data aggregation may not work well with SIMCA-online for real-time project execution, because they might introduce data acquisition delays.
- Database views that perform complicated and time-consuming queries risk being too slow for use in SIMCA-online.

2.3 Networking considerations

You should locate the SIMCA-online server close to the data source in the network. This ensures a fast connection between SIMCA-online and its data source.

Networking equipment, such as firewalls, may interfere with the connection between SIMCA-online and the data source. If this is the case firewall rules may have to be modified.

Verify connectivity from the PC running the SimApi to the data source using for example the Test Connection button in ODBC Data sources.

2.4 ODBC Drivers

The SimApi requires ODBC drivers for your data source to be installed on the PC where the SimApi is installed.

ODBC drivers are obtained from the manufacturer of the database. Download and install the latest available version. Drivers for Microsoft SQL Server are often already installed on most Windows computers.

You need the drivers that match the platform of the SimApi. Typically, this means 64-bit x64 drivers to be used with 64-bit Windows and SIMCA and SIMCA-online. For old 32-bit SIMCA versions, the 32-bit ODBC drivers are required.

2.5 Database authentication

Databases require authentication for the SimApi to be able to access data so that only the data that is needed by the SimApi can be accessed by it. Database administrators limit the access in the database to a specific user used by the SimApi.

User authentication can be done in one of three ways with the ODBC SimApi:

1. Specifying the username and password in the SimApi Configuration dialog. The credentials are stored in encrypted format on the PC.
2. Specifying the username and password in the ODBC connection created in ODBC Data sources (see below). This stores the credentials in Windows.
3. Not specifying credentials explicitly, but instead using Windows Authentication and the user account that runs the SimApi. For desktop SIMCA, this means the user running SIMCA, and for SIMCA-online this means the SIMCA-online service account configured in Windows services.

Chapter 3 shows how to configure the SimApi.

2.6 Visual C++ Redistributable

To use the SimApi on a computer, it must have the following software installed:

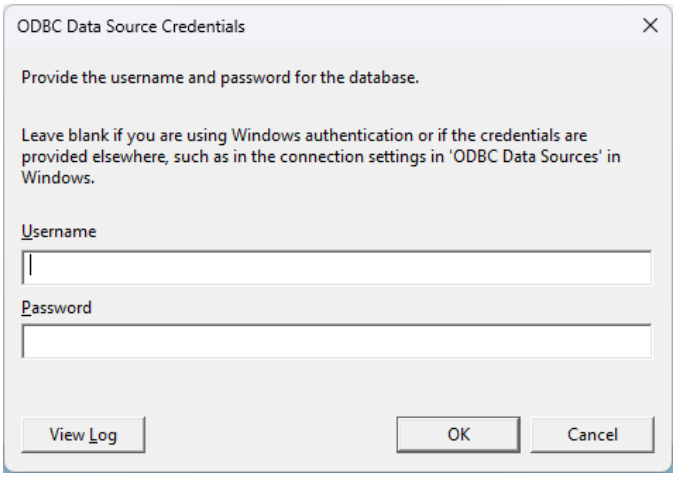
- The **Microsoft Visual C++ Redistributable for Visual Studio** 2015**-2022**. This is already available on all computers with recent versions of SIMCA or SIMCA-online. To run the SimApi in other contexts, the latest version is found at https://learn.microsoft.com/en-us/cpp/windows/latest-supported-vc-redist?view=msvc-170#visual-studio-2015-2017-2019-and-2022

3 Installation and setup

The **SimApi Guide** downloadable from sartorius.com/umetrics-simapi contains good background information on SimApis how install, configure, troubleshoot, and test a SimApi. You may want to refer to this document to learn more about various topics.

To install and configure this SimApi, perform these steps:

1. Install ODBC drivers for your database engine (see chapter 2).
2. Set up an ODBC database connection in Windows ODBC Data Sources and test it to make sure it works (detailed steps in the next section).
3. Install the SimApi on the PC using its installation program: (for detailed instructions, see chapter 5 in the SimApi Guide):
    a. Start by uninstalling any previous version in Windows Apps and Features before installing the new one.
    b. Unzip the zip file, consult the Version Info.txt file and the user guide (which you're reading now...).
    c. Run the setup exe file to install the SimApi.
4. If the Visual C++ Redistributable on the PC isn't up to date, install the most recent version (see 2.6).
5. Add the SimApi to SIMCA or SIMCA-online, as described in chapter 5 in the SimApi Guide.
    a. When you click the button to configure the SimApi you will get this dialog to provide credentials to use (learn about the options in the previous chapter):



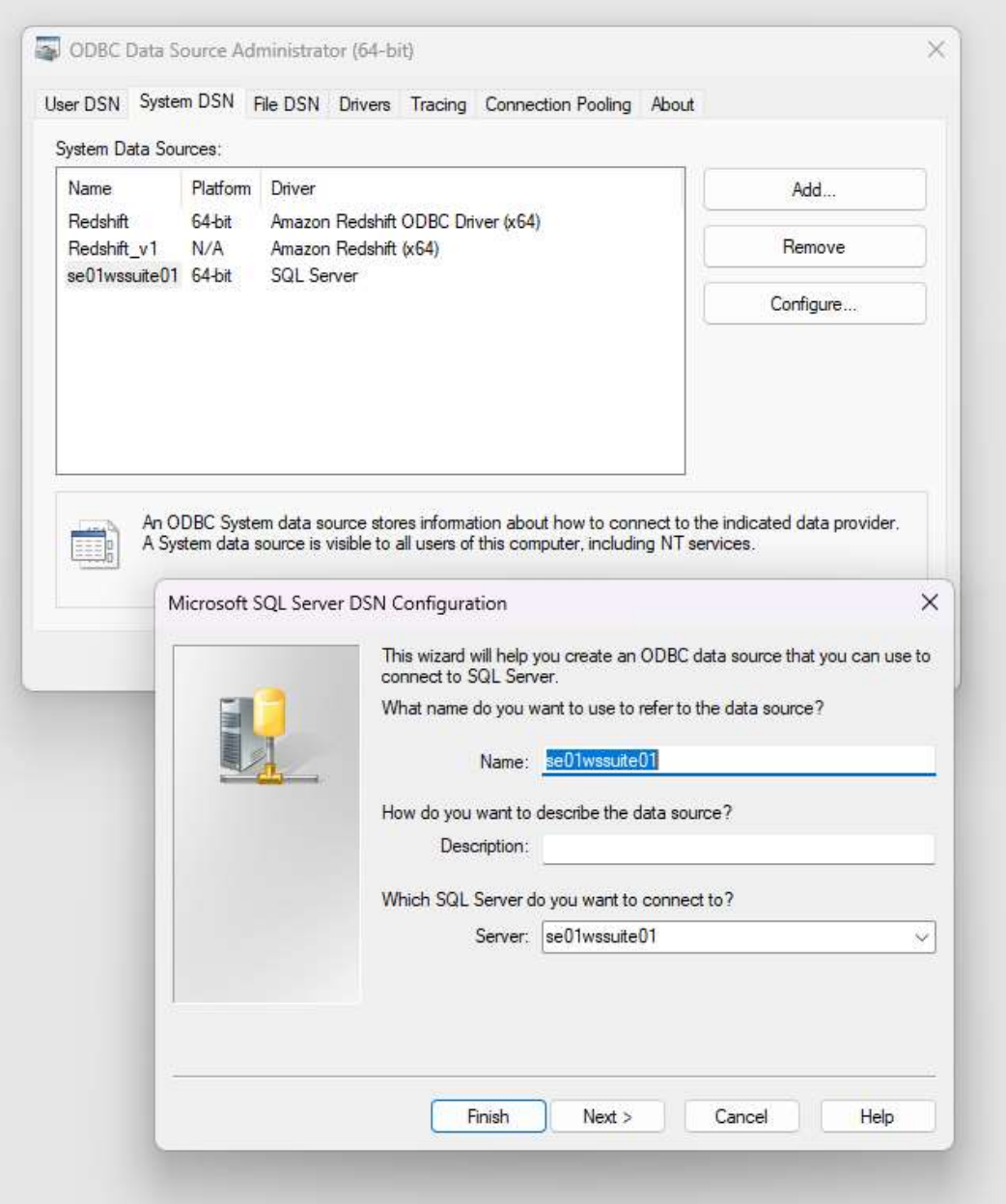
    b. **All other settings for this SimApi are made manually in an XML file using a text editor**. How to configure each feature is described later in this chapter.
6. Test the SimApi. See chapter 6 in the SimApi Guide.

3.1 Configuring an ODBC data source connection in Windows for use by the SimApi

The SimApi requires a configured **ODBC data source connection** to work.

You create this in the **ODBC Data Sources control panel in Windows**. There are two versions of this tool on 64-bit Windows: one for 32-bit applications and one for 64-bit. Use the one the matches the SimApi you are using, typically 64-bit since all recent versions of SIMCA-online and SIMCA are 64-bit and require the 64-bit SimApis variants.

This screenshot shows this application in Windows. There are three different database connections in this example. The below dialog shows the first page of the configuration wizard for the SQL Server connection to the server se01wssuite01:



Usage guidelines:

- We recommend that you configure your data sources as **System DSNs as** displayed in the screenshot. This ensures they are available for all users of the PC including services such as SIMCA-online.
- Click Add to add a new connection. Select the driver to use which you installed as described in the previous chapter, matching your database, and click through the wizard and configure settings to connect to your database server.
- At the end of the wizard, you can try the Test Data Source button to verify connectivity to the database. This of course won't work if you provide the username and password in the ODBC itself (see previous chapter).

3.2 Selecting between two ways to access process data

There are two ways to access process data in this SimApi:

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 in 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 specified for each row of data in the lookup view.

Learn more about how to set up these and all other features of the SimApi below. Refer back to chapter 1.1 to read more about features of this SimApi that you're configuring below.

3.3 XML configuration file and log file locations

A SimApi stores its log files in the hidden **Program Data** folder[2]: %programdata%\Umetrics\SimApi, where %programdata% maps to the actual folder on your computer. It defaults to C:\ProgramData.

This contains the SimApi settings in an XML file named ODBCSimApi_<InstanceName>.xml. **You can edit this file manually using a text editor such as Notepad to make changes.**

Each SimApi typically uses its own log file, which similarly to the SIMCA-online server log file will contain data depending on a log level setting. This file is useful for troubleshooting.

The log file is named ODBCSimApi_<InstanceName>.log.

<InstanceName> is the name of the SimApi instance you added in SIMCA-online Server Options or Default for desktop SIMCA. For example, if you added named the instance "MyDB" the log file name will be ODBCSimApi_MyDB.log. Learn more about this in 4.2 – 4.3 in the SimApi Guide.

3.4 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 configured above.
- 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.

These and all other settings also have descriptions in the table at the end of the chapter.

Important: Some databases are case-sensitive when it comes to names of databases, tables, views, columns so make sure you use the correct casing used in your database in the XML file.

3.5 Direct Mode for continuous/process views

Each row in a Direct Mode view is an observation.

The columns in the view represent variables:

- There must be a single column with date/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 date/time column must be of datetime or datetime2 data type.
- The remaining columns will be exposed through the SimApi as tags with the same names as the column names. Values for these columns are process data.

[2] This folder is hidden in Windows by default. To see it in File Explorer you configure it show hidden files. Note that you can navigate to a hidden folder by typing an address in File Explorer's address bar.

- Write back is supported for all tags, however the correct permissions must be set in the ODBC data source. To be able to write back missing values, nulls must be supported for the column.

The configuration in the XML file is straight forward:

```
<!-- Optional: Direct mode settings: -->
<setting key="Tables" value="MyTableName" />
<!-- Time column in PDB/HDB views, and Direct Mode views: -->
<setting key="TimeField" value="MyTimeColumn"/>"/>
```

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 for the SimApi to expose them.

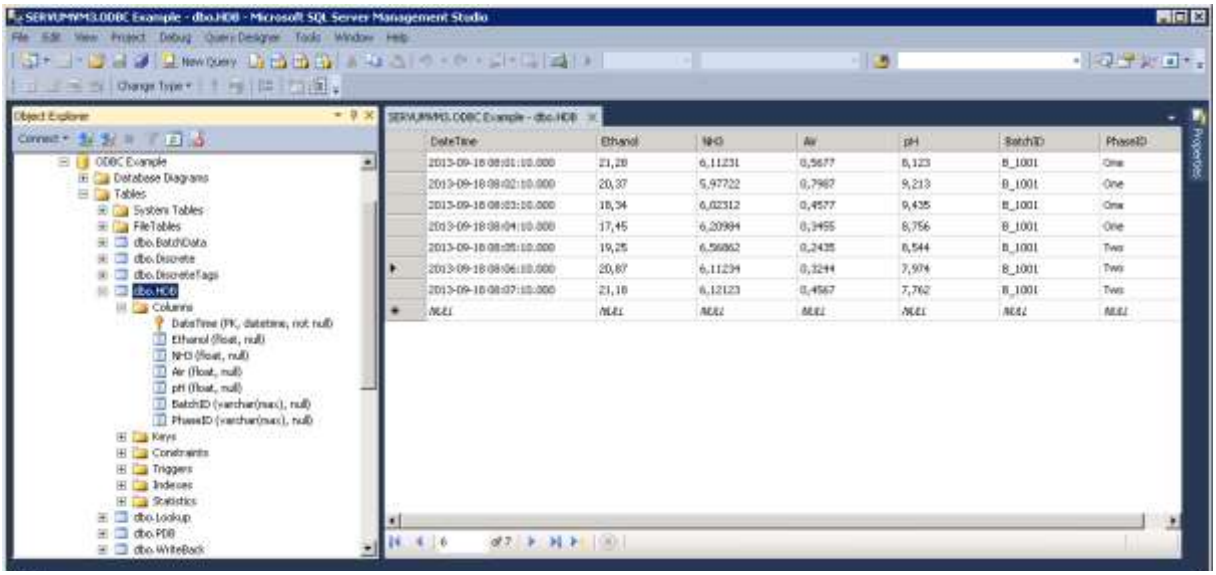


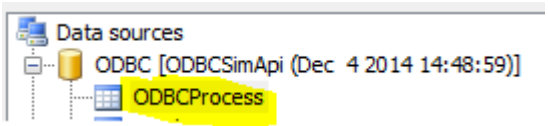
Figure 1. Direct Mode table example.

3.6 Lookup View Mode for continuous/process views

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

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.

The data exposed by the Lookup View is presented by the SimApi as a node that is always 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 must 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 are arbitrary since the names are specified in the configuration file.

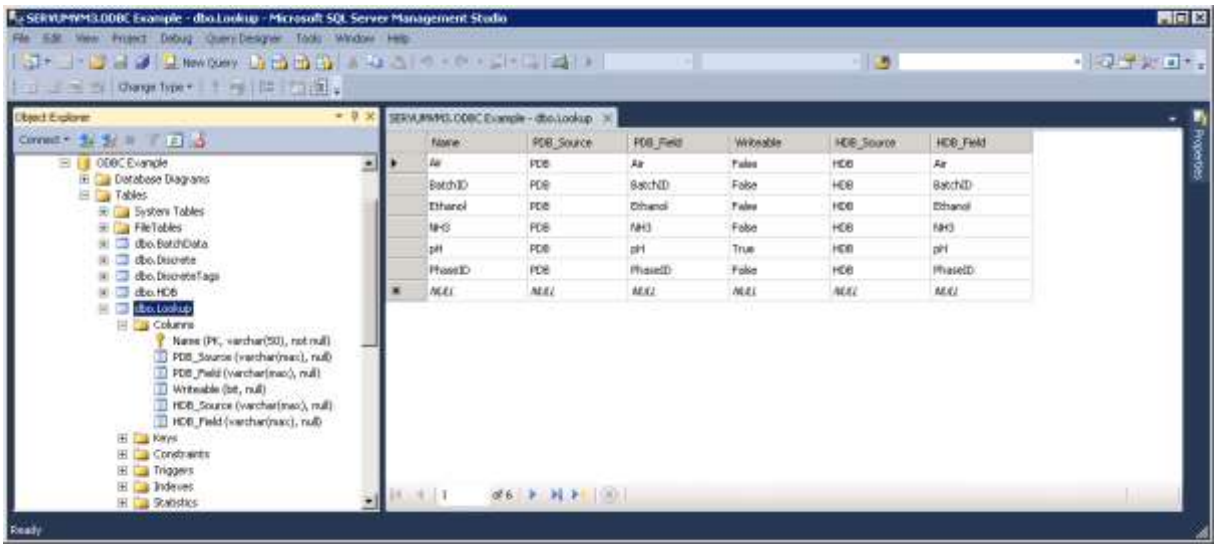


Figure 2. Example of a Lookup view in the form of a database table. In the screenshot you'll see that one PDB source and one HDB source are used. The Name column determines the tag names that the SimApi will use, and in this case the _Field columns use the same column names. Only one tag is writable.

3.6.1 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 it 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 or datetime2, not null).
  – For an HDB: The historical time for the tag values (Primary key, datetime or datetime2, 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 unit in the Execution conditions page 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 execution condition** field in configuration of this project in SIMCA-online.

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

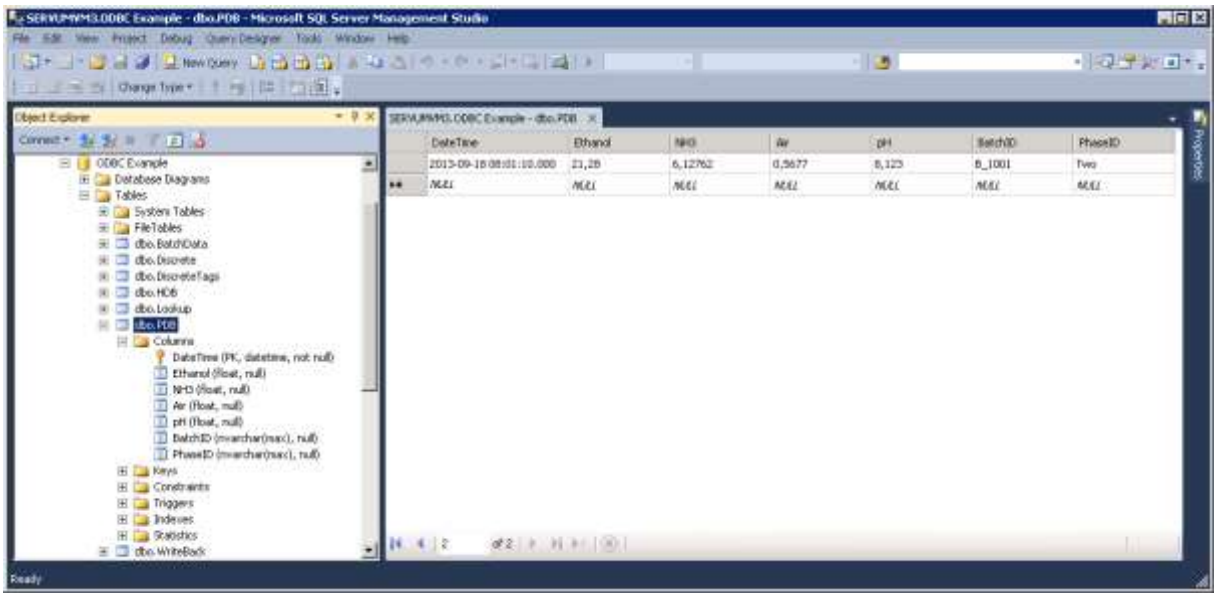


Figure 4. PDB example.

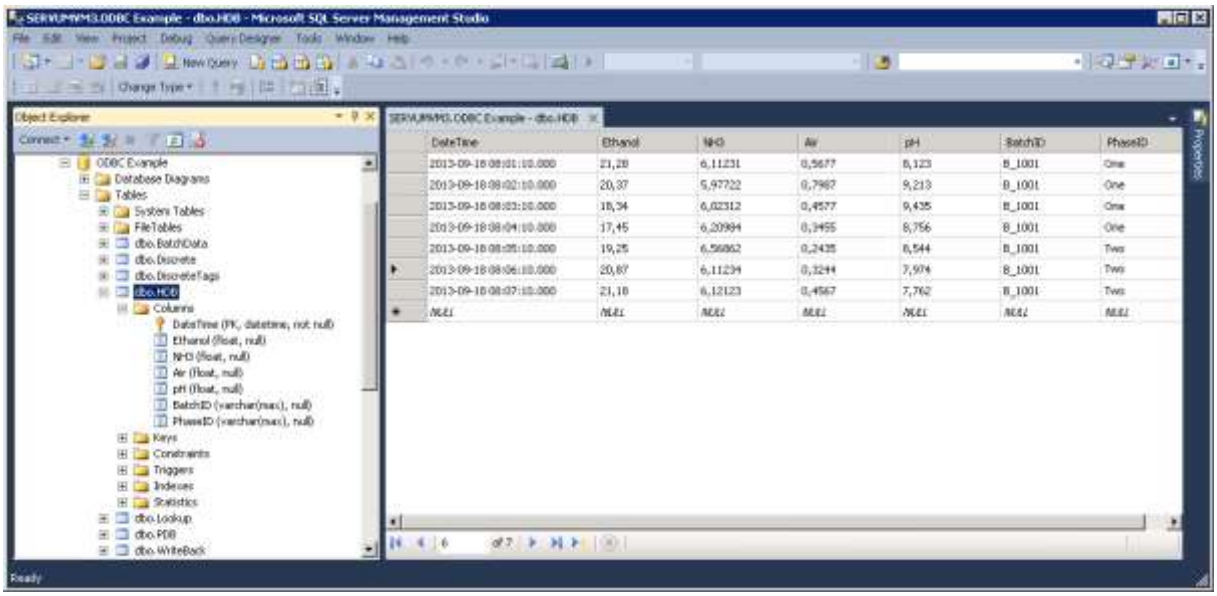
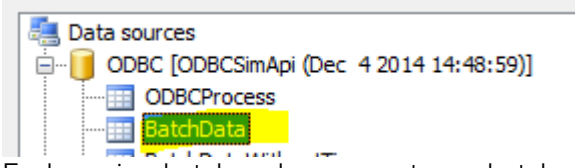


Figure 3. HDB example.

3.7 Batch node

A batch node contains meta-information about batches such as start time, stop time, and optionally batch conditions. A batch node is required by SIMCA-online to analyze batch data but can be omitted for a continuous (non-batch) project.

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 or datetime2, not null).
- BatchStop – The time when the whole batch is completed in the system (not in a unit), null if not completed (datetime or datetime2).

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.

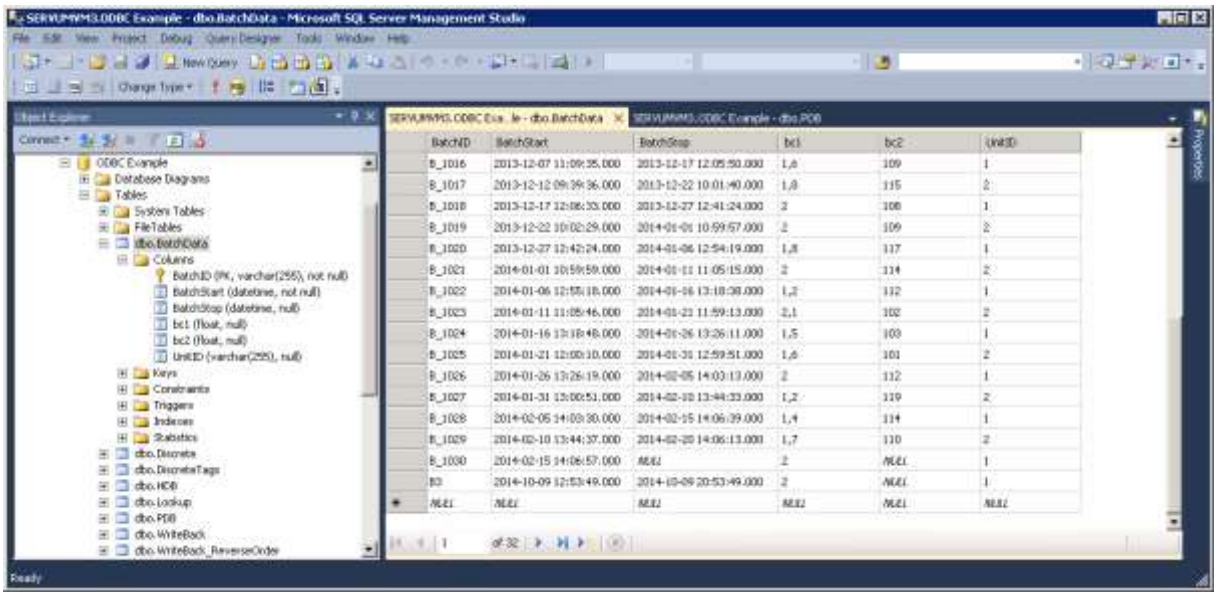


Figure 5. Batch node example with two batch conditions (bc1 and bc2) and a UnitID column.

3.8 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 float[3]).

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:

[3] 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.

```
<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.

3.8.1 Synthetic batch data instance tags

To configure the optional batch data instance tags, you 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 is 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.

3.8.2 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 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.

3.9 Discrete nodes

Optional discrete nodes contain discrete data measurements. Each row in a discrete node represents one measurement for a batch and tag at a given time.

A discrete node must have the following columns:

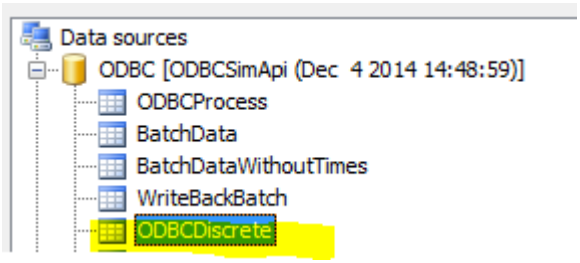
- BatchID – The name of the batch (varchar, not null).
- TagName – The name of the tag (varchar, not null).
- Time – The time when the sample was taken (datetime or datetime2, not null).
- Value – The measurement value (float). Discrete data cannot be string data.

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

Each discrete node configured is exposed as a node by the SimApi. The name of the node is the view name in the database, unless it is overridden. The node name, view name, lookup view, and the column names are configured in the XML configuration file using the attributes of a DiscreteNode element like this:

```
<DiscreteNode NodeName="ODBCDiscrete" ViewName="Discrete" TagLookupView="DiscreteTags"
BatchIDField="BatchID" TimeField="Time" TagNameField="TagName" ValueField="Value"/>
```

The above example exposes a node named ODBCDiscrete by the SimApi, it enumerates the tags using the DiscreteTags table/view. Data is read from the Discrete table/view using the configured columns.



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

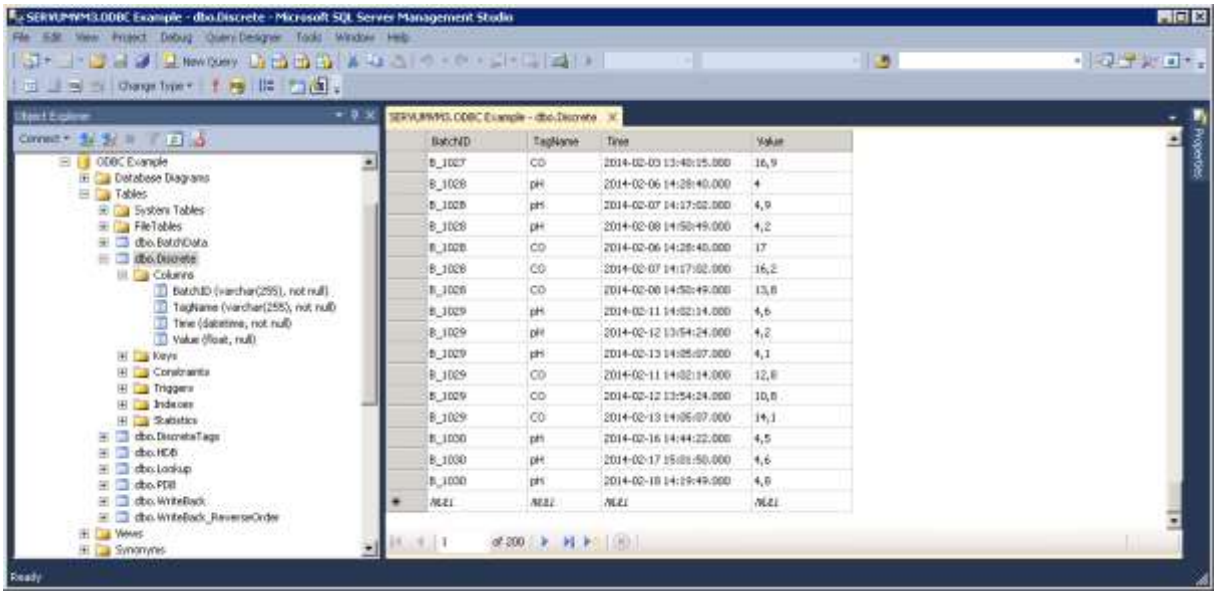
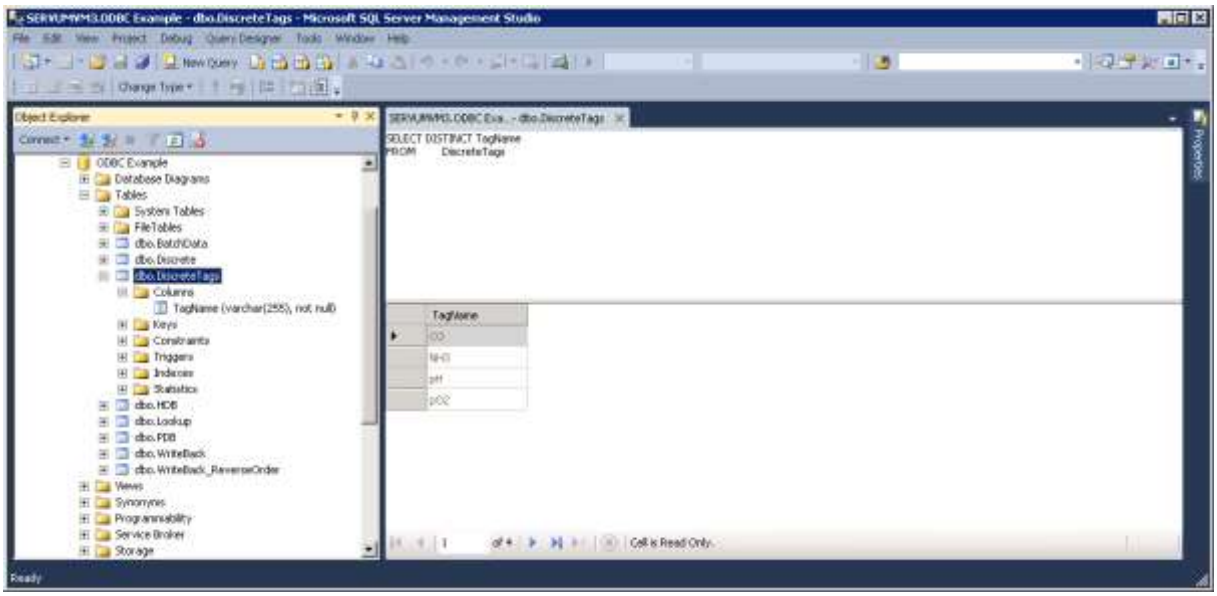


Figure 6. Discrete node example with two tags sampled three times (at roughly 24 hour intervals) per batch.

3.9.1 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.



3.9.2 Discrete data as seen by SIMCA-online

The following is how the discrete data in Figure 6. Discrete node example with two tags sampled three times (at roughly 24 hour intervals) per batch.Figure 6 Figure 2**Error! Reference source not found.**will look in SIMCA-online when combined with the batches in Figure 5.

Discrete Extracted Data (at time 2014-05-27 11:39:48)

	1	2	3	4	5	6	7	8
1	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.

3.10 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.

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.
- If missing values are written back those will be written as null, so the database must support null values for this to work.

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.
- The columns must allow nulls to support writing back missing values

3.11 XML Configuration File

This table lists all settings in the XML configuration file. Use this info and the descriptions above to configure the SimApi.

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 username and password in an encrypted form. Use the Configure button to specify the username 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 ";".
DatabaseConnectionPoolSize	The maximum number of concurrent connections to the database that are allowed. The default is 10. This setting can improve performance by allowing more than one thread to simultaneously access the database. Learn more in chapter 1.

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 must be identical in all views.
Lookup 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 must 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	One <BatchNode> element with the following attributes that control the settings for the synthetic filtered batch node.

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	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. This is an example of how it could look like in the configuration file: <BatchNode ViewName="BatchNode" BatchIDColumn="BatchID" StartTimeColumn="StartTime" StopTimeColumn="StopTime" FilterColumn="Unit FilterValues="FilterValue1| FilterValue2" /> Multiple <BatchNode> elements are supported. Note that you can have different names of the columns between batch nodes.
Batch data view settings	Stored in one or more *<BatchDataView>* elements. See 3.8 Batch Data Views.
Discrete node specific settings	An <DiscreteNode> element for each discrete node with the following attributes. See 3.9 Discrete .
NodeName	The name of the node as it will be exposed by the SimApi. If left blank the value of the ViewName will be used.
ViewName	The name of the view/table that contains the discrete data. If left blank the node is disabled.
TagLookupView	The name of the view/table that defines the discrete tags to use. If left blank the ViewName will be analyzed at startup to enumerate all tags there.
BatchIDField	The column name of the identity of the batch that was measured.
TimeField	The column name of the time of the measurement.
TagNameField	The column name of the tag that was measured.
ValueField	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

This SimApi is developed by Sartorius Data Analytics. For support, please visit sartorius.com/umetrics-support.

Sartorius Stedim Data Analytics AB
Östra Strandgatan 24
903 33 Umeå
Sweden

Phone: +46 90-18 48 00
www.sartorius.com

The information and figures contained in these instructions correspond to the version date specified below.
Sartorius reserves the right to make changes to the technology, features, specifications and design of the equipment without notice.
Masculine or feminine forms are used to facilitate legibility in these instructions and always simultaneously denote all genders.