# SimApi Guide

September 5, 2024

SARTORIUS

# Contents

# 1 Introduction to SimApis

A SimApi is a software interface between the Umetrics® Suite software and a data source. The primary purpose of a SimApi is to provide data to SIMCA®-online or SIMCA®.

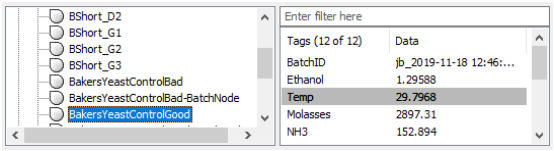
Sartorius Stedim Data Analytics AB develops SimApis for many different data sources, such as process historians and general-purpose databases.

This document shows what a SimApi is, and how it is used in Umetrics Suite products. You'll learn how to plan for, and install a SimApi, how to troubleshoot and how to test your installation. The final chapter contains technical details of SimApis aimed at developers.

## 1.1 SimApi purpose: provide data to Umetrics Suite products

The primary purpose of a SimApi is to provide data to SIMCA-online or SIMCA from a data source. The data source is not part of SIMCA-online but can be a process historian or other system that keeps and manages the data.

A SimApi exposes a hierarchy of nodes, corresponding to folders in a file system. Each node can contain other nodes, or tags. A tag corresponds to a variable. For these tags, data can be obtained. The picture shows a tag, **Temp**, selected in the node **BakersYeastControlGood** in a data source in SIMCA-online. It also shows the latest values taken from the data source.



## 1.2 SimApi usage in the Umetrics Suite

The desktop software **SIMCA** can use a SimApi to retrieve data for project creation and model building as the following picture illustrates.

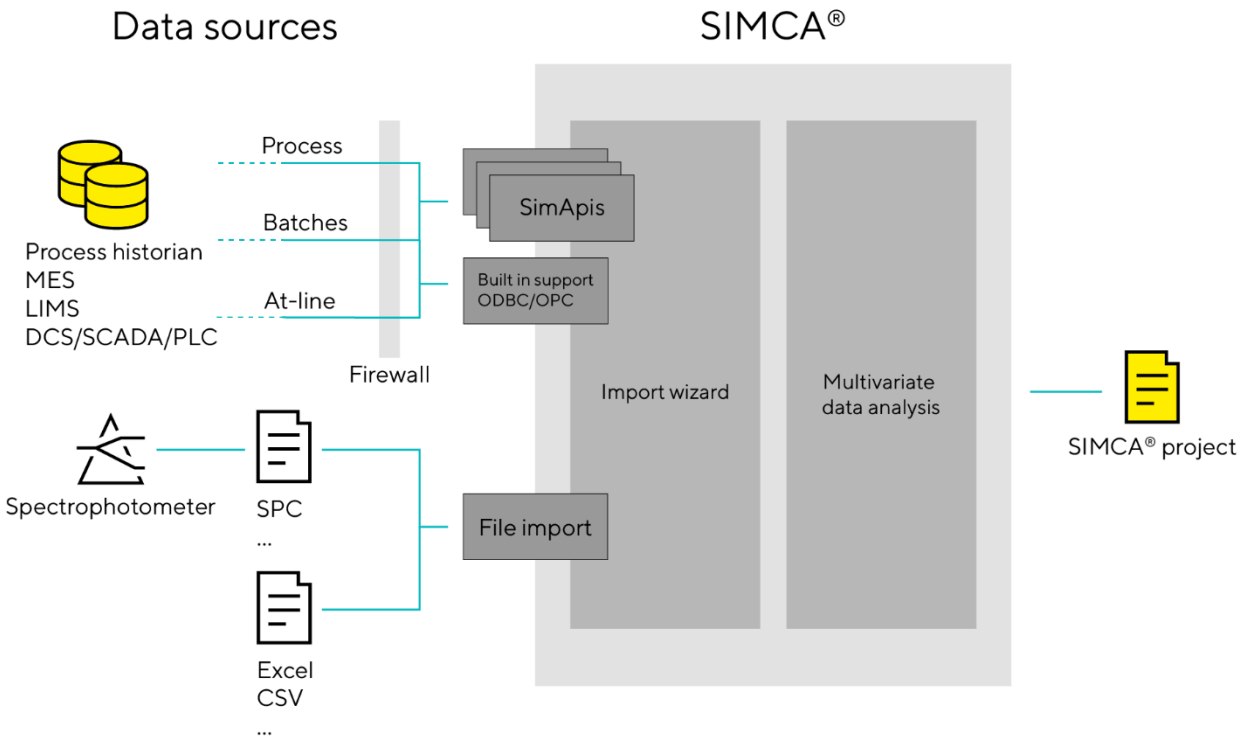



Figure 1. SIMCA used to obtain data from a data source through a SimApi.

**SIMCA-online** uses SimApis to obtain data in real-time for monitoring and control, as well as write back data to the data source. The following picture shows where the SimApi is in a system consisting of a data source, SIMCA-online server, and clients.

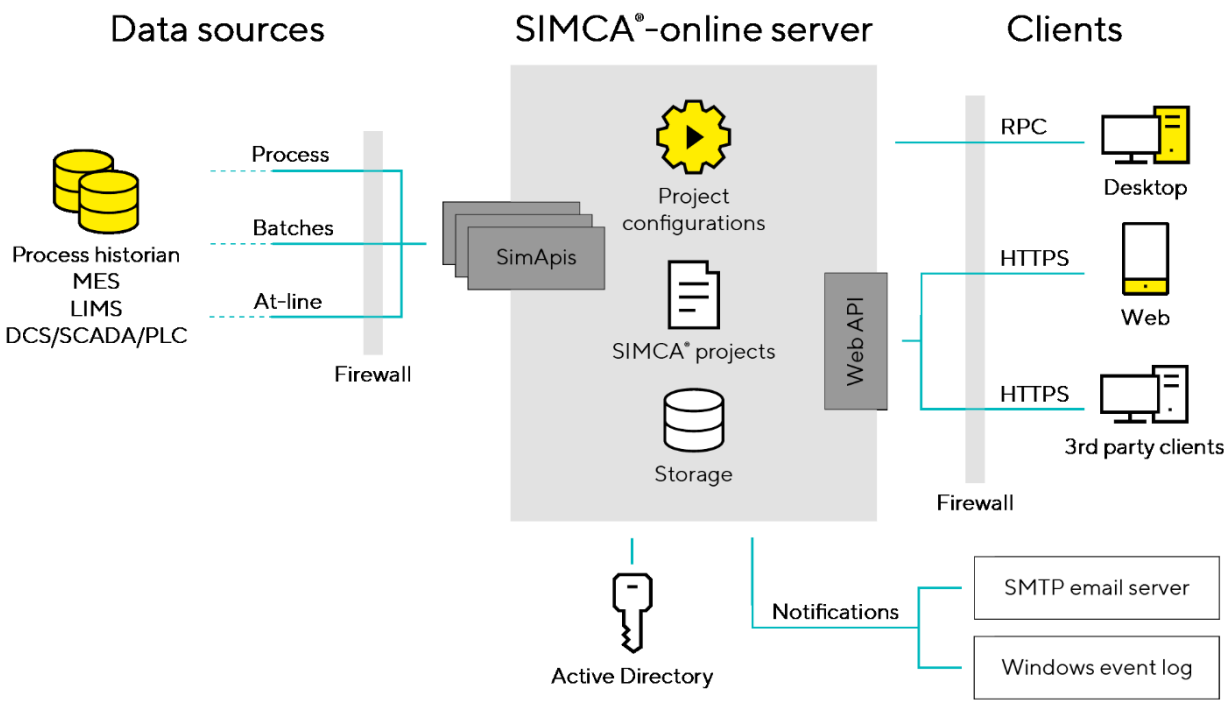



Figure 2. A SIMCA-online installation with a server with SimApis connecting to data sources, and SIMCA-online clients working with data on the SIMCA-online server.

## 1.3 Commonly used SimApis

The most widely used SimApis are:

- The **PI AF** SimApi for connecting to Aveva (formerly OSIsoft) PI Systems.
- The **OPC UA** SimApi
- The **ODBC SimApi** – for general access to databases such as SQL Server or Oracle

All available SimApis are listed together with their features in paragraph 3.

## 1.4 The DBMaker SimApi for simulation data

DBMaker is an application provided with the SIMCA-online server installation. It simulates a data source, such as a process historian, by using a preloaded data table where observations are provided one by one to SIMCA-online through the DBMaker SimApi.

DBMaker is only used for demonstration purposes and cannot be used in production with live data from a data source.

See the built-in help to learn more about DBMaker.

## 1.5 Additional documentation

This document is one of a set of related documents, each with different focus and target audience:

| Source | What | Where |
|---|---|---|
| SIMCA-online web page | Introductory information and downloads | sartorius.com/umetrics-simca-online |

| | | |
|---|---|---|
| SIMCA-online ReadMe and Installation.pdf | Installation and how to get started with SIMCA-online demo data | In the installation zip file |
| SIMCA-online Implementation Guide | Outlines SIMCA-online functionality, puts it in context with other Umetrics Suite software, describes requirements and best practices for successful deployment, and step-by-step installation instructions. | sartorius.com/umetrics-simca-online |
| SimApi Guide | Preparing for and performing SimApi installations, including troubleshooting. Also contains technical details on SimApis for developers. | sartorius.com/umetrics-simapi |
| SimApi User Guides | Documentation for each published SimApi with features, installation instructions, and configuration specifics. | sartorius.com/umetrics-simapi |
| SIMCA-online Technical Guide | Technical reference for SIMCA-online server installation planning, troubleshooting, and in-depth how SIMCA-online works. | sartorius.com/umetrics-simca-online |
| SIMCA-online help | Web-based help how to use SIMCA-online and how SIMCA-online works. | In the software itself, and on sartorius.com/umetrics-simca-online |
| SIMCA-online Web Client Installation Guide | Describes the installation of the SIMCA-online Web Client. | sartorius.com/umetrics-simca-online |
| Umetrics knowledge base | Searchable database with articles about each released software version, technical articles, and known issues in Umetrics Suite products. | sartorius.com/umetrics-kb |
| SIMCA help / user guide | How to use desktop SIMCA for creating projects and modelling data. | In SIMCA and on sartorius.com/umetrics-simca |
| Support web page | How to obtain support technical support. | sartorius.com/umetrics-support |

## 1.6 Technical support

Sartorius **online support team** answers technical questions about SimApis and can also forward requests for enhancement of SimApis to the appropriate people. Learn more at sartorius.com/umetrics-support.

# 2 Obtaining SimApis

We provide documentation for available SimApis and links to installation programs at sartorius.com/umetrics-simapi.

Each SimApi is documented in its own User Guide.

The SimApi Guide which you are reading now complements that information with SimApi complementing information when it comes to SimApi planning, installation, and troubleshooting.

# 3 SimApi features

Not all data sources are alike. A SimApi need not implement all functions in the specification. For these reasons, different SimApis offer different functionality. The following matrix lists available SimApis and their features.

| | BUCHI NIR-Online | CSV | Discoverant | Inmation | IP21 | IP21 Batch | JUNES | MFCS Access | MFCS4 | ODBC | OPC DA | OPC HDA | OPC UA | PI AF | SIPAT | Wonderware |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Current data** | o | o | o | o | o | | o | o | o | o | o | o | o | o | o | o |
| **Historical continuous data** | o | | o | o | o | | o | o | o | o | | o | o | o | o | o |
| **Discrete data** | o | | o | | | | | | | o | | | | | | |
| **Batch data** | o | | o | | | o | | | | o | | | | o | o | |
| **Batch context node** | o | | o | o | | o | | o | o | o | | | o | o | o | |
| **Write back - continuous data** | o | | | o | o | | | | | o | o | o | | o | | o |
| **Write back - discrete data** | o | | | | | | | | | o | | | | | | |
| **Write back - batch data** | o | | | | | | | | | o | | | | o | | |
| **Node hierarchy** | | | | o | | o | o | o | o | | o | o | o | o | | |
| **Array tag expansion** | | | | | o | | | | | | o | o | o | | | |
| **Multiple data sources** | o | | | o | o | o | o | o | o | o | | | o | o | | o |
| **Connection resiliency** | o | o | | o | | | o | o | o | o | | | o | o | | o |
| **Developed in-house** | o | o | o | o | o | o | o | o | o | o | o | o | o | o | | o |

The features are explained below. Notice that the table has separate columns to show which features are available in SIMCA-online and SIMCA respectively.

| Feature | Purpose | SIMCA-online usage | SIMCA usage |
|---|---|---|---|
| Current data | Read a single observation with the most recent value from the data source. | Real-time normal execution | - |
| Historical data | Read many observations at once with historical data from the data source. | Catch-up and predict of past data, create projects using File > New | Database Import Wizard to import process data for model creation. |
| Discrete data | Read laboratory/IPC data from the data source. Many observations per batch. | For batch projects with phases or batch conditions configured for discrete data retrieval. | - |
| Batch data | Read batch conditions and final quality attributes (or | Batch conditions or local centering. | Database Import Wizard to read batch conditions for |

| Feature | Purpose | SIMCA-online usage | SIMCA usage |
|---|---|---|---|
| | other MES type data). One observation per batch. | | batch level model creation. |
| Batch node | Specify the start time and the end time (empty for an active batch) for a specific batch. Enumerate all batches that existed in a time range. | Required for execution of batch configurations. | Database Import Wizard to select batches to import. |
| Write back – continuous data | Write continuous data, such as predictions, back to the data source. | Write back data from the batch evolution level, for Control Advisor or for continuous configurations | - |
| Write back – discrete | Write discrete data, such as predictions, back to the data source. | Write back for batch configurations at the batch evolution level for phases configured for discrete data retrieval | - |
| Write back - batch data | Write back batch level data, such as predictions or final quality attributes, to the data source. | Write back for batch configuration at the batch level | - |
| Node hierarchy | The SimApi supports a hierarchy of nodes, similarly to a file system. Each node can contain tags and other nodes. The hierarchy makes is easier to manage a large number of nodes and tags. | Supported in all places where tags are used. | |
| Array tag expansion | An array tag stores multiple values. The SimApi expands the array tag to many individual tags, one for each element in the array. | Supported where tags are used for continuous data. Each expanded tag must be mapped to a variable in the SIMCA project. | |
| Multiple data sources | The SimApi can connect to more than a single data source or supports multiple instances of itself with individual settings and log files for each instance. | Connect to several different data sources of the same kind. | - |
| Connection resiliency | If the SimApi becomes disconnected from the data source, it will try to reestablish the connection automatically. | The SimApi doesn't have to be restarted to reestablish connections to the data source. | - |
| Developed in-house | The SimApi is developed, provided and supported by | | |

| Feature | Purpose | SIMCA-online usage | SIMCA usage |
|---|---|---|---|
| | Sartorius Stedim Data Analytics AB. | | |

## 3.1 Only current data, without historical data, is not recommended

Some SimApis, notably OPC DA, only supports reading current data, and not historical data.

A SimApi that only supports current data cannot be used in desktop SIMCA, because it won't be able to read historical data on which to build the models.

For SIMCA-online, we strongly recommend a data source and SimApi that provide not only current data for real-time execution, but also *historical* data to be able to predict and catch-up past data. SIMCA-online automatically switches between real-time data and historical data as needed and this cannot be turned off.

A data source that only provides current data, but not historical data can work for continuous projects in SIMCA-online, but for batch projects historical data is required.

# 4 Preparing for a SimApi installation

This section describes important information for a successful installation of a SimApi.

## 4.1 64-bit or 32-bit SimApis

There are 32-bit and 64-bit versions of each SimApi.

SIMCA-online and SIMCA are 64-bit and require the 64-bit SimApis variants. The legacy 32-bit SimApis are still available for older installations.

## 4.2 Location for log file and settings

A SimApi stores its log files in the hidden **Program Data** folder[1]:

%programdata%\Umetrics\SimApi, where %programdata% maps to the actual folder on your computer. It defaults to C:\ProgramData.

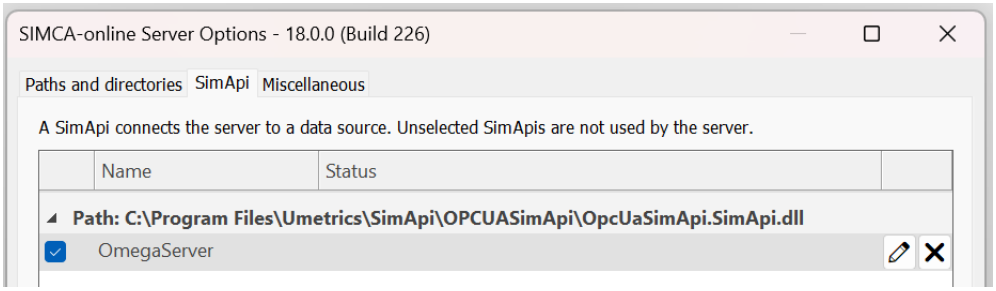
Each SimApi typically uses its own log file, which similarly to the SIMCA-online server log file will contain more or less data depending on a log level setting. This file is useful for troubleshooting. The log file is named <simapi>.log where <simapi> is the SimApi that you are installing, for example PIAFSimApi. Also see the next section for SIMCA-online SimApi instance names.

This folder also contains the SimApi settings in an XML file named <simapi>.xml.

Most SimApis have graphical user interfaces that change the settings in the xml file, but for some you enter the changes directly in the XML file with a text editor, such as Notepad. See the user guide for each SimApi.

## 4.3 File names when named instances are used with SIMCA-online

In SIMCA-online, each SimApi instance get its own configuration file and log file to work with multiple instances of each SimApi. The names of these files are suffixed by the name of the instance as given on the SimApi tab in the SIMCA-online Server Options dialog.




The following example shows the naming of these files, where <simapi> needs to be replaced with the SimApi name.

- Configuration name given when the instance is added: **OmegaServer**
- Configuration file name: <simapi>**OmegaServer**.xml
- Log file name: <simapi>**OmegaServer**.log

Note that the generic file <simapi>.log file is still created. This log file contains entries that for technical reasons cannot be directed to the log file of the instances.

[1] 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.

Note that SIMCA does not support multiple instances of the SimApi, and therefore uses the names without instance name as described above.

## 4.4 Network planning

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 may interfere with the connection between SIMCA-online and the data source.

## 4.5 User accounts and data source permissions

Data sources typically control access to their data. This is usually done with usernames and passwords but IP-address- or DNS-based restrictions can also be used (for example PI Trusts in Aveva PI System).

The username and password can be provided to the data source in different ways:

- A SimApi is run as the Windows user of the user running desktop SIMCA or the SIMCA-online service account on the server computer. The SimApi can connect to the data source using this account. This is how the OPC I, and the PI SimApi work, and ODBC if you don't provide credentials when configuring it.
- For generic ODBC you can use the ODBC Data Sources Administrator application found on Start in Windows.
- Some database providers provide their own drivers and tools for their databases. Oracle databases, for example, use the Oracle Data Access Components (ODAC).
- Some SimApis, such as PI AF and ODBC, have configuration dialogs that store the encrypted credentials in the SimApi XML configuration file.
- PI also has various security options available in the PI System Management Tools on the PI server computer. Read more in the PI AF SimApi User Guide. This guide is helpful even if you use the older OSIsoft PI SimApi.
- OPC DA and HDA use DCOM as the transport between data source and SimApi. DCOM is configured with the Component Services tool (DCOMCNFG.EXE) in Windows and uses Windows authentication.
- For the older OSIsoft PI SimApi (not the newer AF SimApi), the OSIsoft AboutPI-SDK application (PISDKUtility.exe) is used to set up the connection to the PI server.

## 4.6 Verifying data source connectivity

When you want to install a SimApi on a computer it can be useful to verify the connectivity from that computer to the data source with another tool:

- **ODBC Data Sources** in Windows is used to configure and test generic ODBC. Note that there are two versions of this tool on 64-bit Windows: one for 32-bit applications and one for 64-bit. Use the Test Data Source button at the end of the ODBC configuration wizard to verify connectivity to the database. We recommend that you configure your data sources as **System DSNs**.
- **A database specific connection tool** from the provider of the database, such as the Oracle Data Access Components.
- **PI System Explorer can be used to test connectivity to the PI AF server. It is part of the PI AF Client which is a pre-requisite for the PI AF SimApi.**
- **OPC UA Expert** from Unified Automtion - UaExpert is a cross-platform test client for OPC UA servers.
- **AboutPI-SDK application** (PISDKUtility.exe) can be used to test connectivity and to view any error messages that might have been logged when SIMCA-online tries to connect to the PI server. This is only used for the older OSIsoft SimApi, not PIAF.
- **PI System Management Tools** are used on the PI server computer for troubleshooting from that side. For example, to look for security issues preventing access from the SIMCA-online server. Learn more on PI system troubleshooting in this YouTube video.

- **Excel** can be used to obtain data from an ODBC connection and most other systems when a suitable plugin is installed.
- **Matrikon OPC Explorer** for Ior HDA (these are separate tools) can be used to test OPC connectivity, and **Matrikon OPC Analyzer** can be used to diagnose the OPC connectivity issues. Download these free tools from https://www.matrikonopc.com/products/opc-desktop-tools/index.aspx
- **OPC Rescue** (for DInd HDA) from the OPC Training Institute's web site "enables users to easily diagnose communication and security problems, and repair them instantly with the push of a button. All this can be done without ever having to learn to configure DCOM"

# 5 Installing a SimApi

Here is how to install a SimApi on a PC:

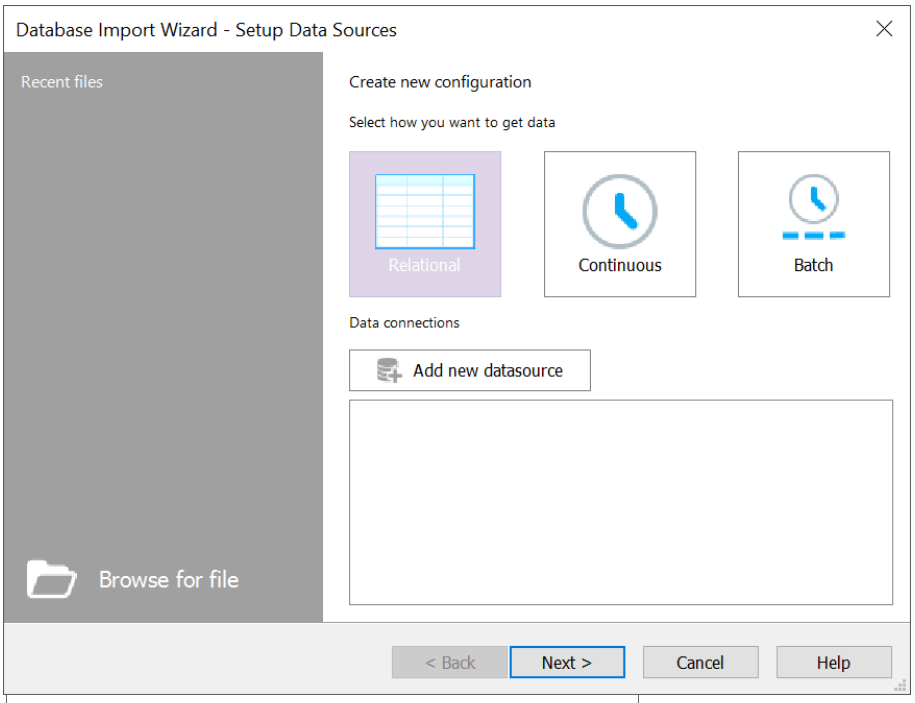
1. Read the **User Guide** for the SimApi you are installing. It contains specifics for that SimApi that complement the general instructions you are reading now.
2. Install and configure any **prerequisites** mentioned in the SimApi User Guide (for example database drivers or SDKs)
3. Run the **setup program to install the SimApi**. Install the 64-bit (x64) or the 32-bit (x86) version that matches the software you will run it in.
4. **Configure the SimApi** in SIMCA-online or SIMCA as described in the following sections and refer to the user guide of the SimApi for descriptions of available settings.
5. **Start the SIMCA-online server**. Note that this can take time, because when the SimApi is initialized, it will enumerate all tags in the data source.
6. Test the SimApi by obtaining some data. For SIMCA-online, you can use **File > Extract** as described in 6.1.
7. If the SimApi fails to work as expected, refer to the **SimApi log files** for troubleshooting, and to the SimApi user guide.

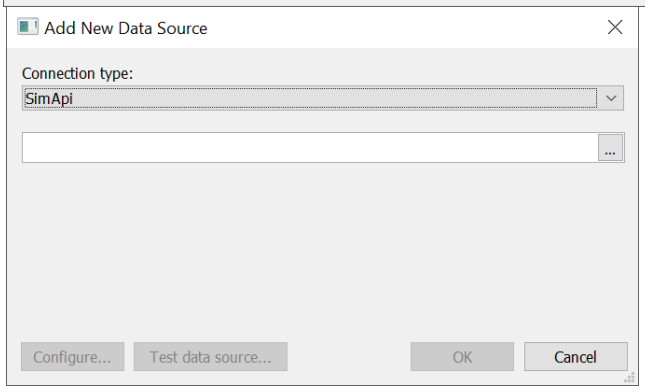
## 5.1 Setting up the SimApi for use in SIMCA

Here's how to use the SimApi in SIMCA:

1. Start the database import in one of the following ways:
    a. **To create a new project in SIMCA: File > New Regular Project** or **New Batch Project**. Select From database on the Home tab.
    b. **To import a data set in an existing project in SIMCA: From dataset** on the Data tab of an open SIMCA project.

2. Click **Add new data source**




3. Select **SimApi** as the connection type, click the **...**-button and locate the <simapi>.dll in the installation folder, and click **Open**.
4. Click **Configure** and refer to the individual SimApi User Guide how to make the settings.
5. Click the **Test data source** connection to verify that you can connect to the database. This can take a long time if there are many tags in the data source.
6. Click **OK** to complete the configuration.
7. Refer to the SIMCA help for how to work with imported data.

## 5.2 Setting up the SimApi for use in SIMCA-online

Important: To be able to use a SimApi, a SIMCA-online server license is required. A demo installation of SIMCA-online does *not* allow SimApis to be used.

To add a SimApi to the system, you run **SIMCA-online Server Options** on the server PC. Learn the details steps in the SICMA-online help topic Add and configure a SimApi on the server.

Tip: If you make changes for a SimApi, you can restart that SimApi separately from Server Options without restarting the entire server.

To configure multiple instances of this SimApi, repeat the above steps and use unique names for each instance. Read more about the different log and configuration files for the instances in 4.2.

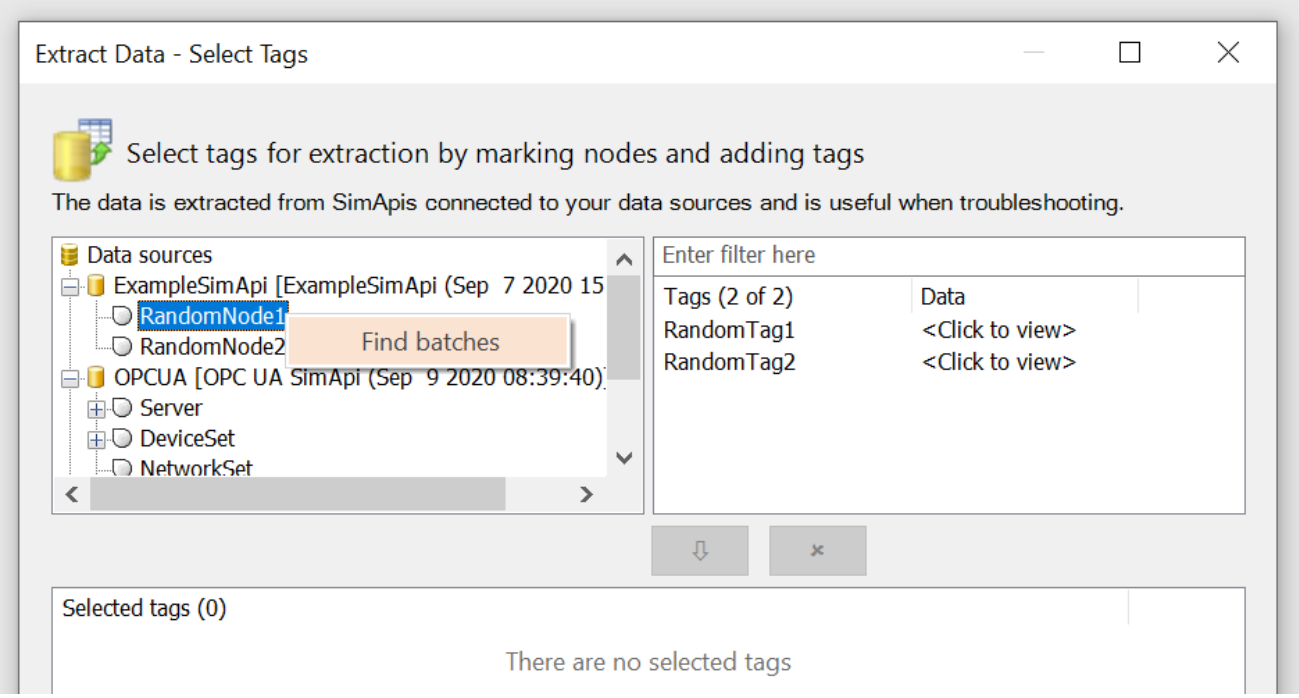
# 6 Testing and troubleshooting a SimApi installation

This chapter is about testing and troubleshooting a SimApi installation.

## 6.1 Testing a SimApi from SIMCA-online

Once the SIMCA-online server has been started successfully you can test your SimApi in SIMCA-online (if the server does not start, see 6.2):

Log in to the server in the SIMCA-online client, and navigate to **Extract** on the **File tab**. Extract helps you test the SimApi by obtaining data through it:




- The **nodes** ("folders") of the SimApi are displayed in the left box. **Tags** for the selected node are displayed top-right.
- **Current data** can be tested quickly simply by clicking **<Click to view>** on tags that provide continuous process data (see the screenshot)
- Right-click on a node to **Find batches** within a time range. The node must be a batch node that knows about batches.
- Select tags in Extract and click Next and finish the wizard to obtain data using the different modes of data retrieval: **current**-, **historical**-, **batch**- and **discrete data**.

Compare the extracted data with what you see in your data source using its tools. Learn more on testing and validating all features of a SimApi in 7.13.

## 6.2 Troubleshoot SimApi problems using the SimApi log file

If the server does not start, the SimApi doesn't work as expected or extract fails, you need to consult the SimApi log file which tells you what the problem is. Enable Debug-level logging in the SimApi log to get full details. See 4.2.

Note: the SIMCA-online server logs are not so useful here. They will show how the SimApi was loaded and initialized by the server, but the *SimApi* specific details are in *its* log file.

## 6.3 Use the right SIMCA-online service account

When you are testing access to the data source, remember that you are logged in as a specific user on the server computer (typically your own user account in a Windows domain), but that the SIMCA-online server **service account** is a different account, by default LocalSystem, which has **different** access rights compared to your user account.

For this reason, it is not uncommon that tests work when run as your account, but that SIMCA-online fails to connect to the data source.

To solve this issue, access must be granted for the account used by the SIMCA-online server service. Typically, you change LocalSystem to a specific domain service account, and grant rights to this account. Note that this does not apply if the SimApi uses credentials that are set in the SimApi configuration because these credentials take precedence.

# 7 Technical details on SimApis

This chapter gives technical details on how a SimApi works. It is mainly aimed at developers that want to understand SimApis for the purpose of implementing a SimApi for a data source.

Developers should also read the earlier parts of this document for an introduction to SimApis and to the high-level descriptions of features.

## 7.1 When to consider developing a SimApi and when not to?

Before considering developing a SimApi for a data source:

1. Investigate if there already is a SimApi that you can use. Perhaps you can enable some feature in your data source to use one of the existing SimApis, such as OPC UA.
2. Carefully go through this document and its references and investigate if your data source fulfills necessary requirements: for example, it needs to be fast enough, provide not just current data, but also historical data.

For these reasons, we don't recommend developing a SimApi that connects to low level hardware or instruments. It is better to connect those instruments to a process historian such as Aveva PI System, and let it obtain data from the instrument, and historize it. Then the PIAF SimApi can be used to obtain data from PI to the Umetrics product.

## 7.2 SimApi development and the SimApi specification

The SimApi specification, **SimApi-v2**, contains documentation for all C-functions in the SimApi that a SimApi DLL needs to implement as well as some guidance for how to develop a SimApi.

Implementing a SimApi using C or C++ is in most cases at an unnecessarily low level.

The recommended, and easier, way to implement a SimApi is to base it on the **ExampleSimApi** source code that we provide. It is an example SimApi implementation that handles the C-interface and translates it into.NET Framework where the actual implementation is made. It also has framework code for logging, settings, configuration GUI and other framework code.

To develop a SimApi, the team of developers need experience in Windows development, .NET Framework, C, or C++. Good knowledge of the data source that the SimApi should connect to is also required, because the purpose of a SimApi is to translate data requests from SIMCA-online or SIMCA to the API of the data source. A SimApi implementation is never a one-off project, but typically needs ongoing support and occasional maintenance.

## 7.3 Reading or writing data

A SimApi has the main task of providing data from a data source. This is referred to as **reading** data.

Most SimApi implementations also support **writing** data. This means writing back data through the SimApi to the data source. Writing data is an optional feature in SIMCA-online.

## 7.4 Tags and Nodes

A **tag** is an identifier of a column or "variable" in a data source. A tag's name is used to identity the tag. Names within node must be unique. SIMCA-online 18 is the first version to support a node which contains a sub node and tag with the same name. For example: the node Parent might have a sub node called Batch and a tag called Batch.

A **node** is a container of tags. A node can also contain other nodes, similarly to how a file system has folders in folders.

Like in a file system, the node and tag names can be combined to a full path that uniquely identifies a tag. The tag paths are used in SIMCA-online or SIMCA when selecting tags to use. A tag path starts with a SimApi instance name followed by the node-structure, and ending with the tag name, each item separated with a colon (:). For example ":ODBCSQLServer:Node:SensorTag1".

### 7.4.1 The SimApi enumerates tags and nodes at startup

A SimApi implementation browses the server for nodes and tags in the data source when the SimApi is initialized and keeps track of them so that the various SimApi functions that are used for enumerating tags and node can be implemented.

SimApi initialization does not happen just at startup of the server but can also be re-triggered by a user in SIMCA-online with the Refresh SimApi functionality.

### 7.4.2 Case sensitivity of tag- and node names

Tag names and node names are **case sensitive**.

Thus, a tag called "tag1" is **not** the same as "Tag1" because of the different case of the "T". We recommend against using tags or node names that differ only in case.

### 7.4.3 Continuous process node

When a node contains tags with continuous process data, it can be referred to as a **process node**. The following two screenshots show a tabular representation of a process node with data followed by a picture showing how the node looks when selecting tags in SIMCA-online.

Sorting_missing [Continuous Node]

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 1 | Observation | $BatchID | Ethanol | Temp | Molasses | NH3 | |
| 5484 | 2015-05-07 09:22:50 | gb_2015-05-07 09:09:20 | 0,05726 | 33,134 | -1493 | 0,09287 | 6704, |
| 5484 | 2015-05-07 09:23:00 | gb_2015-05-07 09:09:20 | 0,055 | 33,1032 | -1494 | 0,11787 | 6704, |
| 5484 | 2015-05-07 09:23:10 | hb_2015-05-07 09:23:10 | 0,0405 | 30,5541 | -1495 | 0,09286 | 750,2 |
| 5484 | 2015-05-07 09:23:20 | hb_2015-05-07 09:23:10 | 0,0253 | 30,8894 | -1496 | 35,3357 | 2015, |
| 5484 | 2015-05-07 09:23:30 | hb_2015-05-07 09:23:10 | 0,09403 | 31,1833 | -1497 | 78,7428 | 2257 |
| 5485 | 2015-05-07 09:23:40 | hb_2015-05-07 09:23:10 | 0,2566 | 30,8559 | -1498 | 84,7357 | 2443, |
| 5485 | 2015-05-07 09:23:50 | hb_2015-05-07 09:23:10 | 0,40119 | 30,6198 | -1499 | 85,8143 | 2605, |
| 5485 | 2015-05-07 09:24:00 | hb_2015-05-07 09:23:10 | 0,5167 | 30,5541 | -1500 | 88,3714 | 2790, |

Figure 3. An example of a node with data. The node name is **Sorting_missing**. The first column contains the time stamps of the observations. The remaining column headers are tags. Each column contains measurements of the tags. Each row constitutes one observation.

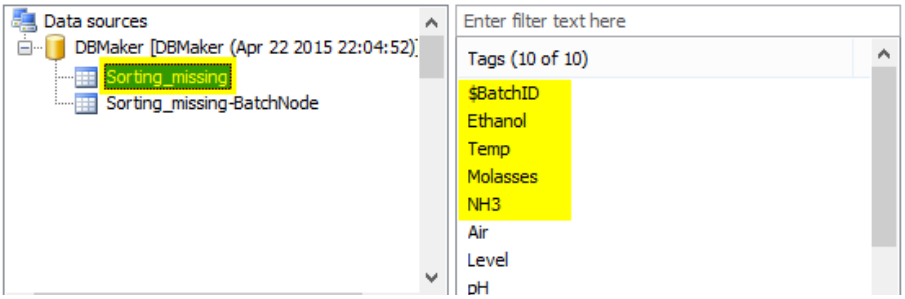



Figure 4: This is how the same node looks from SIMCA-online. You see the node name to the left, and the tags in that node listed to the right. The full tag path to the Ethanol tag in this screenshot is **:DBMaker:Sorting_missing:Ethanol**.

### 7.4.4 Continuous process nodes must be independent of batches, runs or time

To work well in a SimApi a node must be **independent** of batches, runs, or time. Having a node that contains data for a specific batch or time range would not work well in SIMCA-online because the project configuration then could only read data for that batch and not be used for other batches.

Instead, a node should be mapped to one or more physical units in the process where measurements are performed.

### 7.4.5 Batch id tag required in continuous process nodes for batch project execution

Each continuous process must have a tag (variable) holding the batch identifier **for each observation**. This batch identifier is used by SIMCA or SIMCA-online to know to which batch each observation belongs.

The **$BatchID** tag in the screenshots in 7.4.3 is such an example.

While not required, it is recommended to have a tag in the process node that shows the current phase or step of the process. This tag can then be used in phase execution conditions in SIMCA-online or in SIMCA when importing data. Values for this tag can be for example "phase1", “cleaning”, “phase2”.

### 7.4.6 Batch context node

A **batch node is a node that keeps track of batches**; their batch identifiers, start times, and end times. It is a requirement for batch project execution in SIMCA-online. A data source can have more than one batch node that exposes batches in different ways. The user selects the batch node that applies to his or her application. This example exposes batches that span two different units:

- /Factory1 –batches with aggregated lifetimes over both Unit1 and Unit2.
    - /Factory1/Unit1 – batches with lifetimes in Unit1 only
    - /Factory1/Unit2 – batches with lifetimes in Unit2 only

If you don’t have a batch node in your data source, you can use the Batch Context Generator in SIMCA-online. See the built-in help.

### 7.4.7 Optional batch data

A batch node can also contain **batch data**; data for which there is only one observation for the whole batch. Note that tags with batch data need not be in a node that has the full functionality of a batch node. It is enough that the SimApi supports reading batch data for the tags. Learn more on batch data in 7.6.

Here is an example of a batch node:

Sorting_missing-BatchNode [Batch Node]

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Observation | B | Start Time | Stop Time | QP1 | QP2 | Innoc | Amount | Yield |
| 667 | Ta_2015-05-07 06:51:00 | T | 2015-05-07 06:51:00 (1430974260) | 2015-05-07 07:04:40 (1430975080) | c | d | | 6960 | 0,505668 |
| 668 | Va_2015-05-07 07:04:50 | V | 2015-05-07 07:04:50 (1430975090) | 2015-05-07 07:18:30 (1430975910) | a | b | | | |
| 669 | Xa_2015-05-07 07:18:40 | X | 2015-05-07 07:18:40 (1430975920) | 2015-05-07 07:32:20 (1430976740) | c | d | 993,28 | 6597 | 0,504218 |
| 670 | Za_2015-05-07 07:32:30 | Z | 2015-05-07 07:32:30 (1430976750) | 2015-05-07 07:46:10 (1430977570) | a | b | 952,32 | 5541 | 0,45427 |
| 671 | ab_2015-05-07 07:46:20 | a | 2015-05-07 07:46:20 (1430977580) | 2015-05-07 08:00:00 (1430978400) | c | d | 967 | 5749 | 0,492269 |
| 672 | bb_2015-05-07 08:00:10 | b | 2015-05-07 08:00:10 (1430978410) | 2015-05-07 08:13:50 (1430979230) | a | b | 914 | 5365 | 0,482205 |
| 673 | cb_2015-05-07 08:14:00 | c | 2015-05-07 08:14:00 (1430979240) | 2015-05-07 08:27:40 (1430980060) | c | d | 952 | 5875 | 0,500618 |
| 674 | db_2015-05-07 08:27:50 | d | 2015-05-07 08:27:50 (1430980070) | 2015-05-07 08:41:30 (1430980890) | a | b | 952 | 5835 | 0,505364 |
| 675 | eb_2015-05-07 08:41:40 | e | 2015-05-07 08:41:40 (1430980900) | 2015-05-07 08:55:20 (1430981720) | c | d | 952 | 5973 | 0,4816 |
| 676 | fb_2015-05-07 08:55:30 | f | 2015-05-07 08:55:30 (1430981730) | 2015-05-07 09:09:10 (1430982550) | a | b | 960 | 5402 | 0,46254 |
| 677 | gb_2015-05-07 09:09:20 | g | 2015-05-07 09:09:20 (1430982560) | 2015-05-07 09:23:00 (1430983380) | c | d | 950 | 5982 | 0,47437 |
| 678 | hb_2015-05-07 09:23:10 | h | 2015-05-07 09:23:10 (1430983390) | 2015-05-07 15:28:23 (1431005303) | a | b | 943 | 5977 | 0,470986 |

Figure 5. A sample batch node named **Sorting_missing-BatchNode**. The first column contains batch identifiers, followed by the start time and end time of the batch. The last five columns are tags with batch data (batch conditions). This example also shows the different data types which are defined in 7.5: QP1 and QP2 are qualitative (text) tags the remaining tags contain numerical data. Also notice the missing data highlighted in yellow in some cells.

Note: The above screenshot is taken from DBMaker, bundled with SIMCA-online. To see this yourself in DBMaker, click the View Data button on the Bakers Yeast database to display two windows, one of which is the batch node, and the other the process data.

## 7.5 Data types: numerical data, text data and missing data

For each tag, a SimApi can support three types of data: numerical, text and missing:

- **Numerical** data are typically real values of process parameters, for example 6.5123. The SimApi can only handle 32-bit single precision floating point values. Single-precision floating-point format - Wikipedia. All other numerical data types in a data source should be converted to float. As such they can deal with both large and small values but with only about 6 or 7 significant digits. Learn more in the Technical Guide.
- This can lead to loss of precision for large integers or for real numbers that are both large and have decimals. For more information, see the Technical Guide.
- **Text/string** data are used for batch IDs, phase execution conditions or for qualitative variables. The values for text tag data are case sensitive. This means that the value "running" is not the same as "RUNNING". Datetime variables are not supported directly by the SimApi, but they can be returned as a string formatted as YY-MM-DD HH:MM (for example "2020-09-07 13:45").
- **Missing values** means there is no value to return, i.e., no data.

What type is **returned** is up to the SimApi implementation. A SimApi knows about the data in the data source and should return the data type that fits best.

## 7.6 Three modes of data retrieval: Continuous, Batch and Discrete

The SimApi specification defines three modes of retrieval for data, i.e. three different ways the SimApi can provide data from tags in a data source (or in the other direction: write data to tags in a data source).

- **Continuous data retrieval** – this refers to data read continuously, and sequentially, observation per observation as the batch or process evolves. Data is read for the current time, or for a specific range, at a regular interval between observations. For example, all data between 09:00:00 and 10:00:00 sampled every 60 seconds resulting in 61 observations when end points are inclusive.
- **Batch data retrieval** – this refers to a single observation with data for an entire batch (**not** associated with a specific maturity or time point). Batch attributes and local centering data are read as batch data in SIMCA-online. Batch conditions are normally read as batch data too (unless they are configured for discrete data retrieval).
- **Discrete data retrieval** – discrete data can consist of several observations for many maturities. But unlike continuous data, discrete data is not read sequentially but rather all data at once for a specific phase of a batch. Data need not be spaced with regular intervals of the maturity variable. *All* data is re-read each time the data is requested, at the configured interval.

For any given tag data can be requested in any of the three modes, but typically a SimApi will only support one of these modes for an individual tag. Likewise, it is allowed to mix tags within a node, but typically all tags within a specific node support the same mode of data retrieval.

For continuous data (but not for batch- or discrete data[2]), requests can be made for current data or historical data which is the topic of the next section.

Not all SimApis support all modes. See the feature matrix above and the SimApi web page for details.

[2] As you saw above, all discrete data is read each time data is requested, and thus it doesn't make sense talking about current or historical data here.

## 7.7 Current and Historical continuous data through a SimApi

Continuous data refers to process data that changes over time.

### 7.7.1 Current data

Reading **current data** means asking the data source for the latest values of tags at the time of asking. Notice that the time of the external data source is **not** used here.

The data read as current data is what SIMCA-online will show as live data. For this reason, it is important that there are **no unnecessary delays in the data source**. Current data should be as recent as possible to work well in SIMCA-online.

The data source may use its knowledge of data and how long values are valid and decide to return missing data when the raw data for a time point is too old. For example: data is requested at 15:00:00 but the most recent data point in the data source is from 03:00:00. In this case the data is 12 hours old so the SimApi may decide to return missing value (no data).

### 7.7.2 Historical data

Reading **historical data** means asking the data source for values of one or more tags for a specific time range with a specific interval between observations. Notice that here it **is** the data source's local time that is used to find the data. Therefore, time synchronization between data source and servers are important.

Historical data consists of a matrix of data. It is up to the SimApi implementation to request the data from the data source, and sample it at the specified interval and construct the matrix of data to return:

- Sometimes the data source itself has aggregation functions to return processed data, or sampling functions, that can be used to return the right data.
- For other data sources the SimApi must request all data in the time range and then sample the right observations to construct the matrix.
- Data must be returned for a time range, even though there might not be raw data in the time range, but only just before the start time. For example: data exists in the data source at time points 10 and 20. The SimApi requests data for time 15 and 17. In this case the values for timepoint 10 should be returned by the SimApi but timestamped as time 15 and 17 since these were the most recent data points at those times. The values for tags at time 10 are referred to as **bounds values** for the requested range. For a deeper explanation of bounds values, see for example the documentation for **returnBounds** at UA Part 11: Historical Access - 6.4.3 ReadRawModifiedDetails structure (opcfoundation.org)
- Interpolation should never used to calculate values for future time points, because data will not match what is read in real time as current data. For the example from the previous bullet: if data for 15 and 17 were to be interpolated using the values for item 10 and 20, they would effectively use values from the future which is not allowed.

The data source may use its knowledge of data and how long values are valid and decide to return missing data when the raw data for a time point is to old. For example: data is requested for 15:00:00 but the most recent data point in the data source is from 03:00:00. In this case the data is 12 hours old so the SimApi may decide to return missing value (no data).

Note: SIMCA-online typically does not request more than one hundred observations in one call during normal project execution. When doing extract in SIMCA-online, or when running desktop SIMCA, larger requests of data can be made. These can take a long time which is to be expected.

### 7.7.3 Current data and historical data must match

Sometimes there can be **differences when data is read as real-time current data or historical data**. This causes problems in SIMCA-online because the server automatically switches between current and historical data as needed.

### 7.7.4 Low latency data acquisition

When a data source is used by SIMCA-online in real-time, it is important that the data in the data source is current. There should be no unnecessary delays in the data acquisition in the data source. Continuous process data for **all** variables must be available at the same time for every observation. Data that come in late for some variables will not be picked up by SIMCA-online.

## 7.8 Data can be read for any time t

When SIMCA-online asks for a value of a tag for time t it will receive the value from the data source from time t, **or** the latest observation in the data source before time t, **or** an interpolated value for time t. Thus, the server will always get a value at each time it asks for, even though an observation for this exact time point might not exist in the data source.

Timestamps in the SimApi are always UTC. SIMCA-online clients and SIMCA present the time as local time.

## 7.9 Threading

The SimApi is, by default, called by a single thread by the user of the SimApi. This is true for all SIMCA versions and SIMCA-online up until version 17.

SIMCA-online 18 supports a feature flag to turn on multi-threaded access through SimApi. Read more in the help topic Concurrent SimApi access.

This means that SimApis should prepare for multi-threading, if possible, by making the SimApi implementation thread safe, and document this and any considerations for users of the SimApi.

## 7.10 Log file

A SimApi should log actions, error messages and warnings to its log file to help troubleshooting. Use the different log levels to signify the importance of the logging.

It is recommended to log “Not implemented” for features that have not been implemented in a SimApi.

## 7.11 Error handling

When a SimApi cannot fulfill a request from the data source it can handle this problem in one of two ways; by returning missing values (no data) or by signaling a SimApi error:

1. Returning missing values to the caller and signaling success allows the caller to continue as normal (but of course without any data). This is a recommended practice for partial errors such as when data could be obtained for some, but not all, tags in a request.
2. Signaling a SimApi error allows the caller (for example the SIMCA-online server) to see this immediately and to act. This is a recommended practice for requests that fails completely and cannot return any data at all.

SIMCA-online handles missing values or error codes differently, as is described in the SIMCA-online Technical Guide.

## 7.12 SimApi performance requirements

The functions in the SimApi are used to obtain data.

If data access is slow, the SimApi will not work well which this example shows: If SIMCA-online requests data every second, but it takes two seconds to obtain, the SIMCA-online server will never be able to keep up in real-time but progressively fall further and further behind.

In the sub sections we'll show how SIMCA and SIMCA-online uses the data access SimApi functions and how frequently the SimApi functions will be called. This can help in setting performance requirements for a SimApi implementation.

### 7.12.1 SIMCA's usage of SimApi functions

When desktop SIMCA or other offline products use a SimApi to obtain data, these requests will be for batches and process data for a set of variables in a certain time range.

Since these requests are initiated manually by a user, they don't happen very frequently and don't cause a significant load to a data source.

These SimApi functions are used to obtain the data:

- simapi2_nodeGetActiveBatches
- simapi2_nodeGetBatchTimes
- simapi2_connectionReadHistoricalDataEx

### 7.12.2 SIMCA-online's usage of SimApi functions

SIMCA-online is used for real-time monitoring of a process, and hence it requests data through the SimApi at regular intervals. The shortest execution interval that can be used is 1 second. Some real-world examples of execution intervals are 10 s, 1 minute, or 10 minutes.

A server can have many projects running at the same time.

To reduce the number of API calls through the SimApi, the server optimizes data requests by grouping many concurrent smaller requests into a single larger request for all variables at the same time (learn more in the help topic 'Optimized reading from data sources improves performance').

The server's execution algorithm works like this when it requests data using the SimApi functions listed below:

- All phases that execute at the *same interval* are *grouped into a single SimApi* call to reduce the number of calls. The server reads the latest data for **all** variables used by all models that share the interval, i.e., this call will result in a wide data row which then is used by all projects.
    - `simapi2_connectionReadCurrentData`
- For each batch project the server also needs to know which batches are active. This also needs to happen each time a project executes:
    - `simapi2_nodeGetActiveBatches`
    - `simapi2_nodeGetBatchTimes is called less frequently.`
- In addition, SIMCA-online also requires historical data. These requests happen only when needed, such as catch up the beginning of a batch that started before SIMCA-online was started, or when the server is falling behind and needs to read a block of data:
    - `simapi2_connectionReadHistoricalDataEx`
- Optionally, some project configuration use features that use batch data or discrete data which results in SimApi calls to:
    - `simapi2_connectionReadBatchData`
    - `simapi2_connectionReadDiscreteEx`
- Optionally, some project configuration use write-back to push data back to the data source:
    - `simapi2_connectionWriteHistoricalDataEx (and corresponding functions for batch data, discrete data)`

It is important that each call to the core functions for getting data, readCurrentData, getActiveBatches/getBatchTimes, is fast and that is not computationally hard for the data source itself, given how often SIMCA-online may call those functions.

## 7.13 Testing and validating SimApi data

This section is about testing a SimApi to verify that the data returned from it matches data in the data source itself. Running tests like this is important after creating or changing SimApi implementation, or when the API of a data source changes.

In practice, data validation is done using **SIMCA-online** and its **Extract** functionality to pull data from the data source through the SimApi and then comparing with the raw data in the data source. Desktop SIMCA cannot be used to test the real-time aspects of a SimApi.

### 7.13.1 Preparations and requirements

Some items are optional but can be performed if the scope of your testing includes it:

1. Install SIMCA-online as described in ReadMe and Installation Guide.pdf that comes in the product zip.
2. Obtain a license for the SIMCA-online server it and install it. The SimApi won't work without a license. The knowledge base article for SIMCA-online shows how to license the product. For example: SIMCA-online 18 (sartorius.com)
3. Install and configure the SimApi you want to test. Refer to chapters 4 – 5 in this document and the user guide of the specific SimApi.
    a. Optional: make sure the user guide is up-to-date and correct.
4. Make sure you have a tool for your data source which you can use to compare the SimApi data with.
5. In the SIMCA-online desktop client, log in to your SIMCA-online server and use **File > Extract** to obtain data through the SimApi.
6. Optional if your testing scope includes it: after finishing testing, uninstall the SimApi and verify its files are removed.

### 7.13.2 What to test

The feature matrix in chapter 3 lists all possible features, but a given SimApi implementation may support only a subset. You should test all features that are implemented by the given SimApi.

The following tests are common to most SimApi implementations:

- **Authentication** with usernames and passwords
- **Test the various settings in the configuration of the SimApi**
- **Node hierarchy**: The nodes and tags exposed by the SimApi are correct.
    - There must be a tag exposed for all "variables" that should be available through the SimApi. Examples: process measurements, computed values, constants.
- **Connection resiliency**: if the data source is unavailable this results in warnings or errors in the log file, but that the connection to the data source is re-established automatically when the data source is available.
- **Multiple instances**: that two instances can be configured and used independently and simultaneously, with separate logs files.
- **Current data**: extract current data for tags. Make sure data is the last known values from the data source, or missing for bad quality or when data is too old.
    - Extract data every 10 seconds (or so) for a minute.
- **Historical continuous data**: extract historical data for tags.
    - Use the time range that matches when you extracted current data. Verify that current data matches historical data, and the raw data in the data source.
    - Try different time ranges and sampling intervals, verify the data matches the data source.
    - Try extracting data every 1 s, which is the shortest possible sampling interval.
    - Try various type of tags in the data source (process variables, etc.), making sure data matches.
    - Note: SIMCA-online may split a single large historical data request into several smaller chunks. This will be visible in the SimApi log.

- Verify that the SimApi works with text data, numeric data, and missing data.
- **SimApi log file.** Verify that the log contains reasonable entries.
- **Batch node**: right-click a node and do Find batches.
    - Verify batch names, start times, end times for batches.
    - Try an active batch that is running in data source. It should not have an end time through the SimApi.
- **Process node batch identifier tag**. If the SimApi has batch node functionality (see previous bullet), it must also have a batch identifier tag in the matching process data node. Data for this tag should be the batch identifier (batch name). This data is required for batch projects to identify to which batch a row of data belongs.

Depending on if the SimApi supports it, you may also want to test:

- **Batch data** using File > Extract.
- **Discrete data** using File > Extract. Note: to test discrete data with File > Extract the node, the batch node and discrete data node must be in the same SimApi (when SIMCA-online executes projects, they can be from different SimApis).
- **Write back** – pushing data batch to the data source. To test this, you must configure a project configuration in SIMCA-online to write back data vectors to the data source. Then execute the project in SIMCA-online and check the data that is written back in the data source.
    - Continuous data is configured on the **Evolution Write Back page** in the project configuration.
    - Discrete data is configured on the same page, but only for a phase configured for discrete data retrieval.
    - Batch data from the **Batch Write back page**.

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.