SIMCA-Q

Embedded solution

Interface Description SIMCA-Q 16

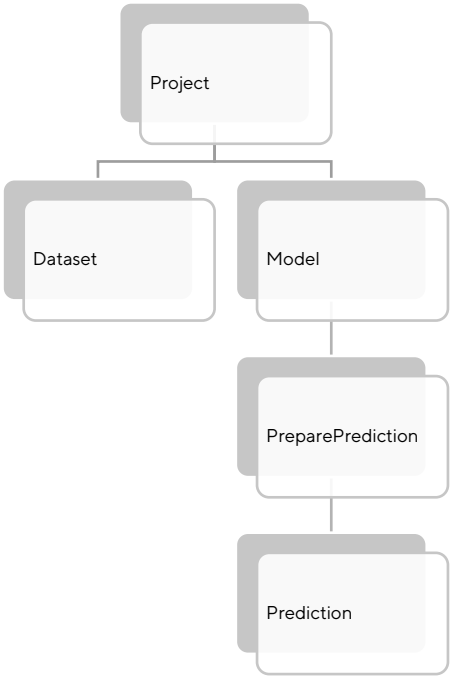
8 June 2020

This document gives an overview of the functions and objects found in the C and COM interfaces of SIMCA®-Q embedded solution. For a general description of SIMCA-Q see 'SIMCA-Q 16 User Guide.pdf'.

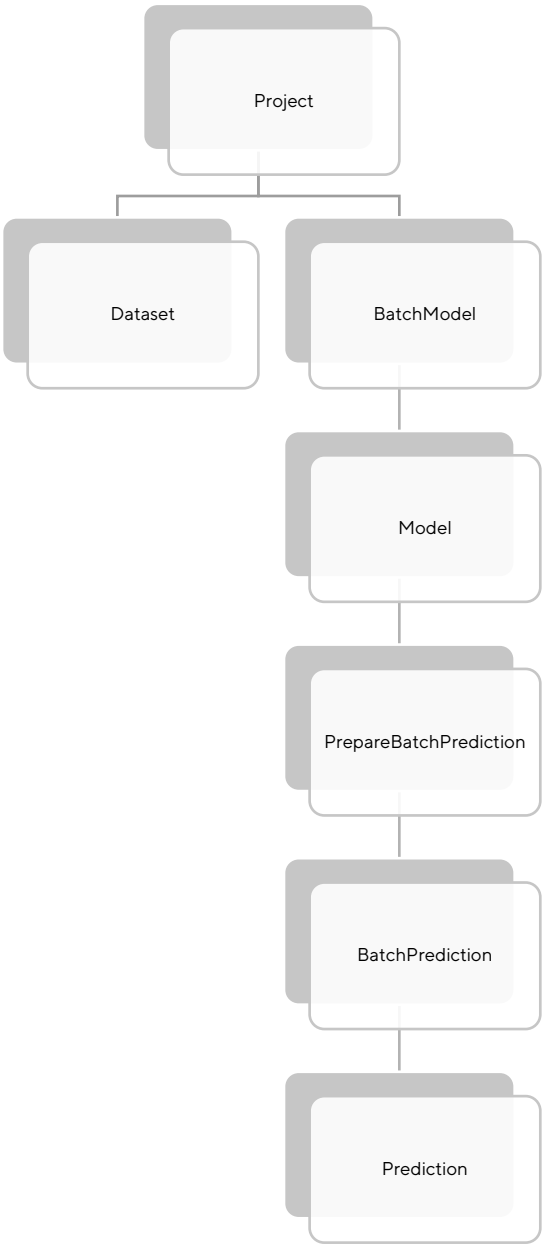
The C interface

Structure when doing predictions

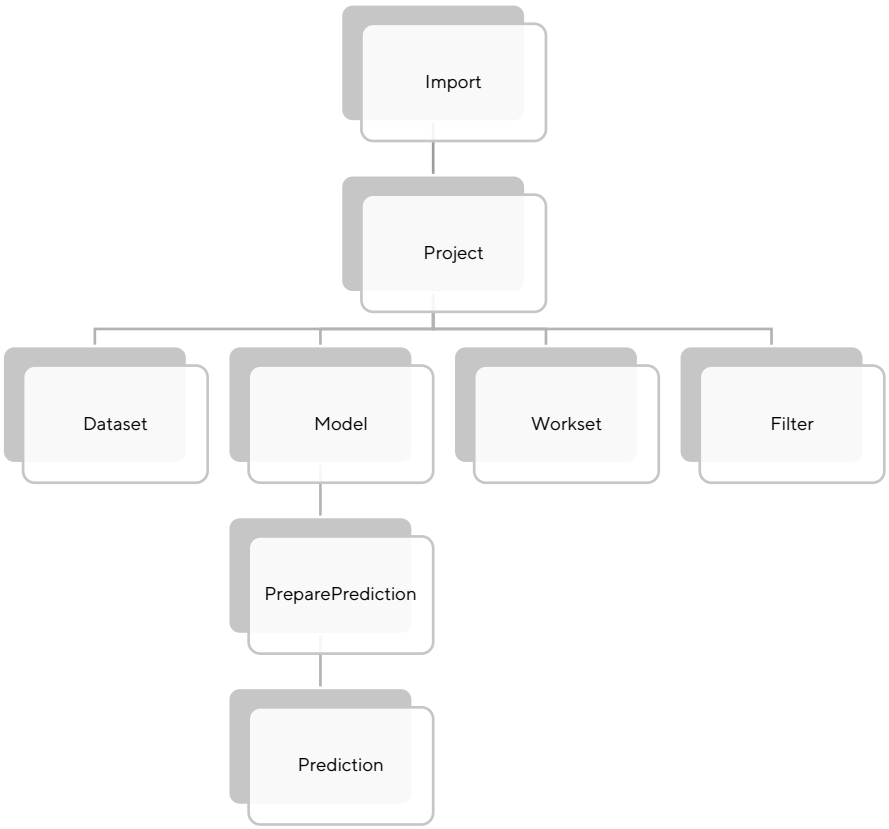
The simple structure of SIMCA-Q for doing predictions can be illustrated as follows:



Structure when doing batch predictions



Structure when creating models



Functions

The functions found in the C Interface can be divided into different categories where one category corresponds to one or several objects.

General

General objects and functions are functionality that is not necessary connected with a specific project, but apply to SIMCA-Q. Some of the more common ones are:

- `BoolVector`
- `ComponentVector`
- `FloatMatrix`
- `FloatVector`
- `IntVector`
- `StringMatrix`
- `StringVector`
- `VariableVector`
- `Variable`
- `VectorData`

Project

The Project object represents a SIMCA-Q project. From this object, such things as getting project options, reading project-, data- and model information are retrieved.

Dataset
The Dataset object represents a dataset, a spreadsheet-like object in a project. The functions in Dataset enable retrieval of dataset information, like variable- and observation IDs and observation data.

BatchModel
The BatchModel object contains the BEM (Batch Evolution Model) and the BLM (Batch Level Model) built on datasets created from that BEM.

Model
The Model object represents a model in a SIMCA-Q project. From the Model object, all model specific data are retrieved. Functions like getting the scores, DModX, the name of the model, number of components and starting a prediction is contained in this object. Also functions to modify the model are available from the Model object in SIMCA-Q, for example, auto fit, changing model type, model name etc.

Filter
From the Filter object creation of a spectral or time series filter is performed. This includes specifying the filter type and all available filter settings.

Import
The Import object is used to import new data to a new or existing project. Specifying observations, variables and data is done with the Import object.

Workset
Before creating a model, a Workset object must be retrieved were scaling, transform and lagged variables are specified.

PreparePrediction
The PreparePrediction object represents a collection of data that is used in a prediction. It has functions for getting the variables required for the prediction, setting quantitative and qualitative data/lags that should be used for prediction.

PrepareBatchPrediction
The PrepareBatchPrediction object contains similar functions as PreparePrediction, but is constructed for a batch prediction. Data for each phase and batch conditions are set through this object.

BatchPrediction
The Batch Prediction object holds the predictions for all batch evolution and batch level models.

Prediction
The Prediction object represents a prediction made on some data and a model. The functions in this object retrieve predicted data from the prediction.

Prefix
All functions in the C-interface files have a SQ_ prefix.

EzQ
This is an example of how to write a high-level C++ interface of the prediction part of SIMCA-Q providing a class to use SIMCA-Q in just a few lines of code. This can easily be extended and modified according to user needs. This class can be found in the C++ sample directory, that can be downloaded from the website, and the sample displays how to use this class.

The COM interface
SIMCA-Q has beside the C-interface a COM-interface. The COM-interface is a dual interface, i.e. it is both a raw COM interface and a Dispinterface. The version independent program ID for SIMCA-Q is `Umetrics.SIMCAQ`, the program ID for SIMCA-Q 15 is `Umetrics.SIMCAQ.4` and the interface is `ISIMCAQ`. The type library, with namespace `SIMCAQLib`, is defined in SIMCAQ.tlb.

Interfaces

The COM interface in SIMCA-Q consists of several interfaces, with ISIMCAQ as the base interface. From the ISIMCAQ interface/coclass some other interfaces/coclasses can be constructed. The only way to create another coclass/interface pointer other than ISIMCAQ is by going through ISIMCAQ or some other interface in SIMCA-Q. If an instance, other than an ISIMCAQ like IProject or IModel, is created directly it will be an invalid instance and SIMCA-Q will probably behave very strange.

When one interface derives/inherits another interface, the first interface is called the base interface and the other interface is called the derived or extended interface. All functions in a base interface also exist and are valid in the extended interface.

When one interface dispatches another interface it means the other interface is returned from some function call in the first interface. The first interface creates the other interface (e.g. retrieving an IProject object from the ISIMCAQ object). In the description of the interfaces below, the data structure interfaces are left out from the Dispatches section.

- BOOL (b as in boolean; like bIsX).
- long (n as in number; like nIndex).
- BSTR (str as in string; like strName).
- float (f as in float; like fDeltaX).
- A SIMCA-Q definition (e as in enum; like ePCA_X).
- An IDispatch pointer which is an Interface pointer (I as in interface; IProject).
- A pointer to any of the above types (p as in pointer; like pfMax).
  A pointer is always used on the **out** parameter and when there is an interface parameter. When there is an out IDispatch/interface pointer there will be pp (pointer to pointer), like ppIProject.

How all these pointers and pointer to pointers affect you when working with the COM interface depends on the programming language you are using. Many modern and easy to use programming languages marshals the pointer arguments automatically when calling the interface functions, which means that you don't need to care about how to format your arguments to suite the function. The pointer can in these cases be referred as a reference to the data type it points at, a reference many programming languages automatically generate.

ISIMCAQ

ISIMCAQ is the base program interface for SIMCA-Q. This interface has common functions like setting log file, handling license file, plug-in path, library version and so on.

Inherits

The ISIMCAQ interface does not extend any other interface and is not extended by any another interface.

Dispatches

The ISIMCAQ dispatches the following interfaces.

The data structures:

- `IStringVector`
- `IFloatVector`
- `IIntVector`
- `IStringMatrix`
- `IFloatMatrix`
- `IBoolVector`
- `IComponentVector`

The data structures are straight forward and will not be described in this document. They all have their starting index at 1, not 0 as some programming languages. For example; an IFloatVector of size 4 has the positions 1, 2, 3 and 4 as valid positions. An IFloatMatrix with 3 rows and 2 columns has the positions 1, 2 and 3 as valid positions for the row, 1 and 2 as valid positions for the column.

Other main interfaces

- `IProject`
- `IFileReader`

IImport

The IImport interface represents a spreadsheet-like import object that will result in dataset, and maybe a new project, when the import is done. From this interface such things as adding quantitative and qualitative data, setting names for observations, variables and pointing out which ones are Y can be done.

Inherits

The IImport interface does not extend any interface. No interfaces derive from it.

Dispatches

The IImport interface dispatches the following interfaces.

- `IProject`
- `IDataset`

IWorkset

The IWorkset interface represents a workset, which is a specification for a model about how to use the data in the dataset. Among the interface functions are inclusion/exclusion of variables/observations, transformation/scaling, handling of classes and creating/updating a model.

Inherits

The IWorkset interface does not extend any interface. No interfaces derive from it.

Dispatches

The IWorkset interface dispatches the following interface.

- `IModel`

IFilter

The IFilter interface is used to create a spectral or time series filtered dataset. This includes specifying the filter type and all available filter settings.

Inherits

The IFilter interface does not extend any interface. No interface derives from it.

Dispatches

The IFilter interface dispatches no other interface.

IProject

The IProject interface represents a SIMCA-Q project. From this interface, such things as setting project options, reading project, data and model information are retrieved.

Inherits

The IProject interface does not extend any other interface. No interface derives from it.

Dispatches

The IProject interface dispatches the following interfaces.

- `IModel`
- `IProjectOptions`
- `IDataset`
- `IShewhartControlChart`
- `IEWMAControlChart`

- `ICusumControlChart`

IModel

The IModel interface represents a model in a SIMCA-Q project. From the IModel interface all model specific data are retrieved. Functions like getting the scores, DModX, the name of the model, number of components and starting a prediction are contained in this interface.

Inherits

The IModel interface does not extend any other interface, and in the batch prediction version it is extended by IBatchLevelModel and IBatchEvolutionModel.

Dispatches

The IModel interface dispatches the following interfaces.

- `IVectorData`
- `IPreparePrediction`
- `IModelStatistics`
- `IModelOptions`
- `IIntVector`
- `IStringVector`

IDataset

The IDataset interface represents a dataset, a spreadsheet-like object in a project. The functions in IDataset enable retrieval of dataset information, like variable- and observation IDs and observation data.

Inherits

The IDataset interface does not extend any other interface. No interface derives from it.

Dispatches

The IDataset interface dispatches the following interfaces.

- `IVariableVector`
- `IDatasetTypeVector`
- `IStringVector`
- `IVectorData`

IVectorData

The IVectorData interface represents the output from a vector. It consists of a float matrix with corresponding row and column names.

Inherits

The IVectorData interface does not extend any other interface. No interface derives from it.

Dispatches

The IVectorData interface dispatches the following interfaces.

- `IFloatMatrix`
- `IStringVector`

IVariableVector

The IVariableVector interface represents a vector with variables.

Inherits

The IVariableVector interface does not extend any other interface. No interface derives from it.

Dispatches

The IVariableVector interface dispatches the following interface.

- `IVariable`

IVariable

The IVariable interface represents a variable in the model or in the dataset. If has functions to get the ID of the variable, information whether the variable is X, Y, qualitative, lagged etc.

Inherits

The IVariable interface does not extend any other interface. No interface derives from it.

Dispatches

The IVariable interface dispatches the following interface.

- `IIntVector`
- `IStringVector`

IBatchModel

The IBatchModel interface represents a BatchModel(BM) which holds the Batch Evolution Model(BEM) and the Batch Level Model(BLM)-models built on datasets created from that BEM.

Inherits

The IBatchModel interface does not extend any interface. No interface derives from it.

Dispatches

The IBatchModel interface dispatches following interfaces:

- `IBatchEvolutionModel`
- `IBatchLevelModel`

IBatchPrediction

The IBatchPrediction interface represents a batch prediction made from a batch project. This interface is only a dispatcher for the IBatchEvolutionPrediction and IPrediction interfaces.

Inherits

The IBatchPrediction interface does not extend any interface. No interface derives from it.

Dispatches

The IBatchPrediction interface dispatches following interfaces:

- `IBatchEvolutionPrediction`
- `IPrediction`

IBatchEvolutionModel

The IBatchEvolutionModel interface represents a batch evolution model. Besides the base interface functions there exist functions for retrieving model aligned data and batch information.

Inherits

The IBatchEvolutionModel interface extends the IModel interface. No interfaces derive from it.

Dispatches

The IBatchEvolutionModel interface dispatches no other interface.

IBatchLevelModel

The IBatchLevelModel interface represents a batch level model. Besides the base interface functions there exist functions for Batch VIP and retrieval of the IPrepareBatchPredictions interface.

Inherits

The IBatchLevelModel interface extends the IModel interface. No interfaces derive from it.

Dispatches
The IBatchLevelModel interface dispatches following interface:

- `IPrepareBatchPrediction`

IPreparePrediction
The IPreparePrediction interface represents a collection of data that is used for a prediction. It has functions for getting the variables required for the prediction, setting quantitative and qualitative data/lags that should be used for prediction.

Inherits
The IPreparePrediction interface does not extend any other interface. No interface derives from it.

Dispatches
The IPreparePrediction interface dispatches the following interfaces.

- `IVariableVector`
- `IPrediction`

IPrediction
The IPrediction interface represents a prediction made on some data and a model. The functions in this interface retrieve predicted data from the prediction.

Inherits
The IPrediction interface does not extend any other interface, and in the batch prediction version it is extended by IBatchEvolutionPrediction.

Dispatches
The IPreparePrediction interface dispatches the following interface.

- `IVectorData`

IPrepareBatchPrediction
The IPrepareBatchPrediction interface contains functions to specify data for a batch prediction. These include specifying data for each phase, local centering and batch conditions.

Inherits
The IPrepareBatchPrediction interface does not extend any interface. No interface derives from it.

Dispatches
The IPrepareBatchPrediction interface dispatches following interfaces:

- `IBatchPrediction`

IVariableOrigin
The IVariableOrigin interface represents the origin of a batch level variable.

Inherits
The IVariableOrigin interface does not extend any interface. No interface derives from it.

Dispatches
The IVariableOrigin interface dispatches no other interface.

IShewhartControlChart
The IShewhartControlChart interface represents a Shewhart Control Chart.

Functions like getting the Mean Upper/Lower Control limits and the standard deviation process are contained in this interface.

Inherits
The IShewhartControlChart interface does not extend any interface. No interface derives from it.

Dispatches
The IShewhartControlChart interface dispatches no other interface.

IEWMAControlChart
The IEWMAControlChart interface represents a EWMA Control Chart. The functions in IEWMAControlChart enable retrieval of the EWMA Upper/Lower Control limits and the EWMA standard deviation.

Inherits
The IEWMAControlChart interface does not extend any interface. No interface derives from it.

Dispatches
The IEWMAControlChart interface dispatches no other interface.

ICusumControlChart
The ICusumControlChart interface represents a Cusum Control Chart. It has functions for retrieving the cumulative sum on the high side and the low side difference, the action limit and deadband among others.

Inherits
The ICusumControlChart interface does not extend any interface. No interface derives from it.

Dispatches
The ICusumControlChart interface dispatches no other interface.

IProjectOptions
The IProjectOptions interface contains all options about a project, confidence level, significance level etc.

Inherits
The IProjectOptions interface does not extend any interface. No interfaces derive from it.

Dispatches
The IProjectOptions interface dispatches no other interface.

IModelStatistics
The IModelStatistics interface represents different statistics for a workset or a model.

Inherits
The IModelStatistics interface does not extend any interface. No interfaces derive from it.

Dispatches
The IModelStatistics interface dispatches no other interface.

IModelOptions
The IModelOptions interface contains all options about a model.

Inherits
The IModelOptions interface does not extend any interface. No interfaces derive from it.

Dispatches
The IModelOptions interface dispatches no other interface.

IFileReader
The IFileReader interface represents the import of a prediction set from a file. The IFileReader interface is only available if allowed by the license.

Inherits
The IFileReader interface does not extend any interface. No interfaces derive from it.

Dispatches
The IFileReader interface dispatches the following interfaces.

- `IFileReaderSpecification`
- `IPrediction`

IFileReaderOptions
The IFileReaderOptions interface represents the settings available for different file types.

Inherits
The IFileReaderOptions interface does not extend any interface. No interfaces derive from it.

Dispatches
The IFileReaderOptions interface dispatches no other interfaces.

IFileReaderSpecification
The IFileReaderSpecification interface represents the structure of the file to import, were the data starts in the file, etc.

Inherits
The IFileReaderSpecification interface does not extend any interface. No interfaces derive from it.

Dispatches
The IFileReaderSpecification interface dispatches no other interfaces.