SARTORIUS

MODDE®-Q

Design of Experiments Solution

Interface description for MODDE®-Q 13

November 16, 2021

MODDE®-Q exposes automation interfaces. The program id for MODDE®-Q is MODDEQ.Application. The type library, moddeLib, is defined in MODDE-Q.tlb.

With MODDE®-Q it is possible to create new investigations and to view results from any existing investigation. Most functionality available in MODDE® is also available in MODDE®-Q including all designs, constraints, predictions and the optimizer.

MODDE®-Q can also export a HTML report from a template created by the Report in MODDE® 9 and later, with all the plots and lists supported by MODDE's Report.

More information on the MODDE® terminology can be found in the MODDE® user guide.

Activation

After installing MODDE®-Q it must be activated through the COM interface. MODDE®-Q can be run as a demo for 60 days.

When the activation functions are called the computer must be connected to the Internet. If the computer running MODDE®-Q is not connected to Internet, the license file can be downloaded to another computer and transferred. See the document "Manual Product Activation.pdf" on how to download a license file and then use the function in the COM interface to activate the license file. The Host ID referred in the document can be retrieved through a function in the COM interface.

SARTORIUS

Updates in version 13

See the help file for details.

1. One new ResponseRole (Response criteria in GUI):

   ```
   eRRInside
   ```

2. New enum ResponseCondition:

   ```
   typedef enum
   {
         eRCRequired = 0,
         eRCDesired,
         eRCObserved
   } ResponseCondition;
   ```

3. New functions in IOptimizer:

   ```
   LONG GetResponseCondition(BSTR strResponse, LONG* eCondition) const;
   LONG SetResponseCondition(BSTR strResponse, LONG eCondition);
   LONG GetDesirability(IDispatch* piStringVectorFactorSettings, FLOAT* fDesirability);
   ```

4. New function in IModel:

   ```
   LONG GetExpandedNumberOfTerms(LONG* iNumTerms);
   ```

5. New function IAnalysis:

   ```
   LONG GetVIPUnsorted(IDispatch** pInvestigationData);
   ```

Updates in version 13.0.1

1. New function in IApplication:

   ```
   SaveOfflineActivationFile(BSTR szPath, BSTR szActivationKey)
   ```

2. New functions in IFactor:

   ```
   LONG SetNOR(FLOAT fValue)
   LONG GetNOR(FLOAT* pfValue);
   ```

3. New function in IOptimizer:

   ```
   LONG COMOptimizer::SaveResponseSettings()
   ```

Overview

The MODDE®-Q interface consists of several interfaces, with IApplication as the base. The IInvestigation interface can be created from IApplication, and the other interfaces can be created through IInvestigation, see Figure 1

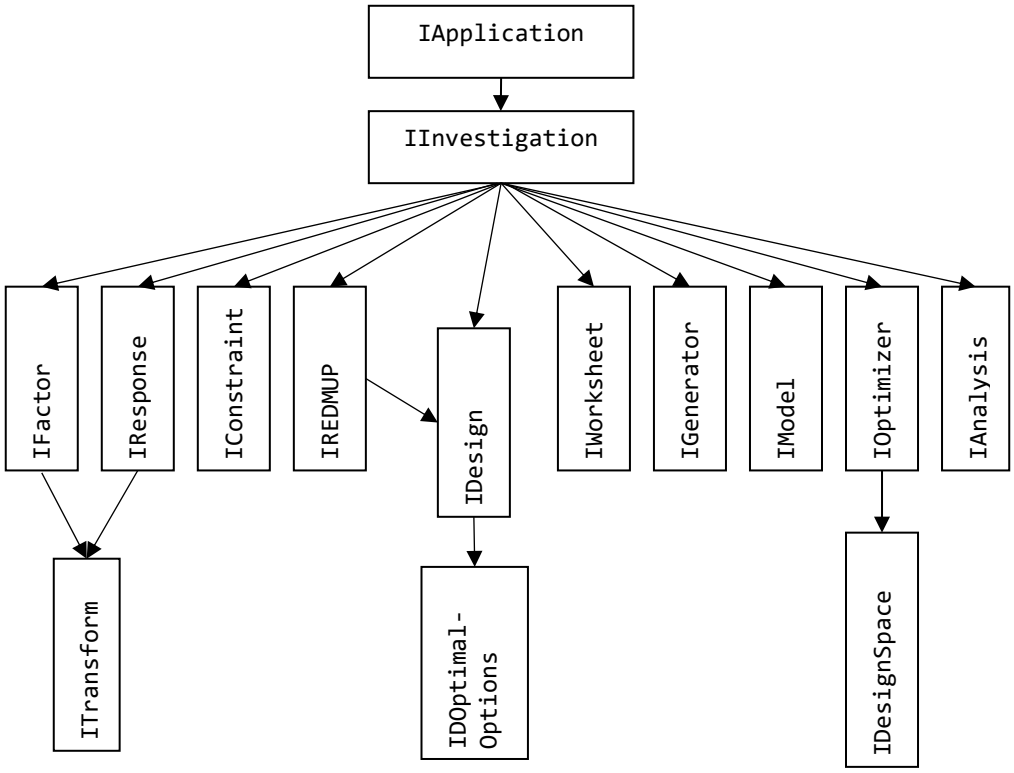

Figure 1 Interface overview

In addition to the interfaces in Figure 1 there are a few data container interfaces which can be created by any interface and trough "MODDEQ.Application". They are IInvestigationData, IIntVector, IIntMatrix, IFloatVector, IFloatMatrix and IStringVector.

Every function returns S_OK if the call succeeded and S_FALSE or an error code if an error occurred. If an error occurred more information can be found by calling **GetLastError**().

Typical scenarios

Get the results from an existing investigation

To open an existing investigation, create an instance of MODDEQ.Application and call the function **OpenInvestigation()**, this will return an IInvestigation interface. Call **GetAnalysis()** in IInvestigation get the Analysis interface and from there most of the results are available. The Optimizer and the Report generator are also accessed through the IInvestigation Interface.

Create a new investigation

To create a new investigation, create an instance of MODDEQ.Application and call the function **NewInvestigation()**, this will return an IInvestigation interface. Add the factors by first calling **NewFactor()** in IInvestigation to get an IFactor Interface. Setup the factor trough that interface and finish by calling **AddFactor()**. Repeat this for every factor. For responses call **NewResponse()** and **AddResponse()**.

Select design by calling **NewDesign()** to get the IDesign interface. Use the IDesign interface to setup the design, and possibly also **IsDesignValid()** and **GetRecommendedDesign()** in IInvestigation. Finish by calling **GenerateDesign()**.

Get the IWorksheet interface trough **GetWorksheet()** in IInvestigation and fill in the response values.

Analyze the results as above.

Interface description

IApplication

The IApplication interface is the base interface that is used when opening and creating new investigations. It is through the IApplication interface MODDE®-Q can be activated.

IInvestigation

The IInvestigation interface holds the information about the open investigation.

It is used to create a new investigation and to modify an existing one. The interface can be retrieved from the IApplication interface.

IWorksheet

The worksheet can be read and rewritten. Both factor settings and response values can be changed, as well as the experiment name and run order. If the setting for a factor is changed to a value higher or lower than the limit of a factor, the setting will still be changed.

Most functions take an experiment number as argument, that number is the same as shown in the worksheet in MODDE®, from 1 to the number of the last experiment.

All functions in the IWorksheet interface returns a LONG, which corresponds to a HRESULT. S_OK is returned if the call succeeded and one of the error codes defined below if an error occurred.

IFactor

The IFactor interface holds information about a factor and is also used for creating new factors. The interface can be retrieved from the IInvestigation interface.

IResponse

The IResponse interface holds information about a response and is also used for creating new responses. The interface can be retrieved from the IInvestigation interface.

IAnalysis

The IAnalysis interface holds most of the information found under the Analysis menu in MODDE®. The interface can be retrieved from the IInvestigation interface.

IConstraints

The IConstraints interface holds the information about a constraint. The interface can be retrieved from the IInvestigation interface.

IDesign

The IDesign interface holds information about a design and is used for creating new designs.

IDesignSpace

The IDesignSpace interface holds information on how to estimate design space by Monte Carlo simulations.

IDOptimalOptions

The IDOptimalOptions interface holds information about the possible options when creating a new D-Optimal design.

IGeneralizedSubsetDesign

The IGeneralizedSubsetDesign interface holds information about a Generalized subset design and is used for creating new designs.

IGeneralizedSubsetDesignSummary

The IGeneralizedSubsetDesignSummary interface holds information about a design subset in a Generalized subset design.

IGenerator

The IGenerator interface holds the information about a generator.

IModel

The IModel interface holds the information about the model.

IOptimizer

The IOptimizer interface holds the information about the optimizer.

IREDMUP

The IREDMUP interface holds the information about and creates a RED-MUP design.

ITransform

The ITransform interface holds information about the transformation for a factor or response. The ITransform interface can be created from the program ID MODDEQ.Transform.

Container interfaces

IInvestigationData
The IInvestigationData interface holds information about a data table.

IFloatMatrix
The IFloatMatrix interface holds a floating point matrix of a specified size. The IFloatMatrix interface can be created from the program ID MODDEQ.FloatMatrix.

IFloatVector
The IFloatVector interface holds a floating point vector of a specified size. The IFloatVector interface can be created from the program ID MODDEQ.FloatVector.

IIntMatrix
The IIntMatrix interface holds an integer matrix of a specified size. The IIntMatrix interface can be created from the program ID MODDEQ.IntMatrix.

IIntVector
The IIntVector interface holds an integer vector of a specified size. The IIntVector interface can be created from the program ID MODDEQ.IntVector.

IStringVector
The IStringVector interface holds a vector of strings of a specified size. The IStringVector interface can be created from the program ID MODDEQ.StringVector.