September 2022, V 2.0

Quick Start Guide OntoStudio-X

Introduction

OntoStudio-X (OSX) combines the advantages of two well-established technical worlds: the table organization of Microsoft Excel and the inference engine OntoBroker. By combining OntoBroker's function and data structures with Excel's internal object model (C and COM API), all of OntoBroker's functions can be accessed within the Excel cell structure WITHOUT using Excel VBA. Excel files that have OntoBroker ontologies and instruction structures can thus also be passed on macro-free (as a normal .xlsx file).

A significant contribution to the smooth integration of OntoBroker in Excel is made by Excel's new dynamic array functions. Thus, the result of a query in a cell can fill an entire field of cells. If required, this result can then be further processed using Excel functions.

Since the coupling between Excel and OntoBroker runs via the Java API, individual OSX functions can also return objects that are only intermediate results for further processing and cannot be displayed in a cell. These are then displayed as a named reference in the cell and used correctly in further processing (e.g. =OB.CreateManager(...) returns the value: ClientOntologyManagerImpl@1).

In addition to the integration of the OntoBroker Java API, there is another functional interface Python (currently Python 3.10). In addition to all kinds of data processing that cannot be performed natively by Excel or OntoBroker, such as neural networks, many special functions that would conventionally be left to the Excel macro language VBA are performed via Python. For example, the OS-X editor is controlled via Python (F2 key), which, among other things, also functionally color-codes ObjectLogic constructs. In addition, many OntoBroker extensions are provided on the basis of Python, which are not generically part of the OntoBroker reasoning functionality (e.g. in the case of the axioms declaration: in the OntoBroker Command Mode "insert ...", in the direct mapping of the Java API command "=OB.DeclareAxioms(...)" and in the extended Python implementation "=OSX_DeclareCmplxAxiom(...)".

The Java and Python functions are fixed in the basic version of OSX ("baked" versions) and cannot be extended. For creating user-defined functions in Java or Python, additional development packages are available on request.

OSX differs from earlier, Eclipse-based versions of OntoStudio by its high flexibility, which in the realization by means of MS Excel appeals to a much larger user group compared to the developer community that is familiar with Eclipse. New functions or extensions/improvements are created at least weekly, since the development team of semafora also works with OSX and has a high interest in a constantly growing range of functions to make work easier.

Systemvoraussetzungen und Komponenten

Excel	Current (2022 update) Office 365 or Office 2021+, each in the 64-bit version. The use of online based Excel versions is not possible.
OntoBroker	From version 6.3 - of OntoBroker only the address / port and the credentials must be known. Several OntoBroker servers can be addressed simultaneously. The location of the servers is irrelevant.
OSX Package	Zip to unpack in the user directory. The zip file contains all components incl. OpenJDK and Python, which are necessary for the operation on OSX ("semafora OSX Vxx_yy.zip").
Windows	Windows 10/11, 64-Bit

It is recommended to perform at least the annual updates of Microsoft products. Installation instructions can be found in Appendix 2.

Usage

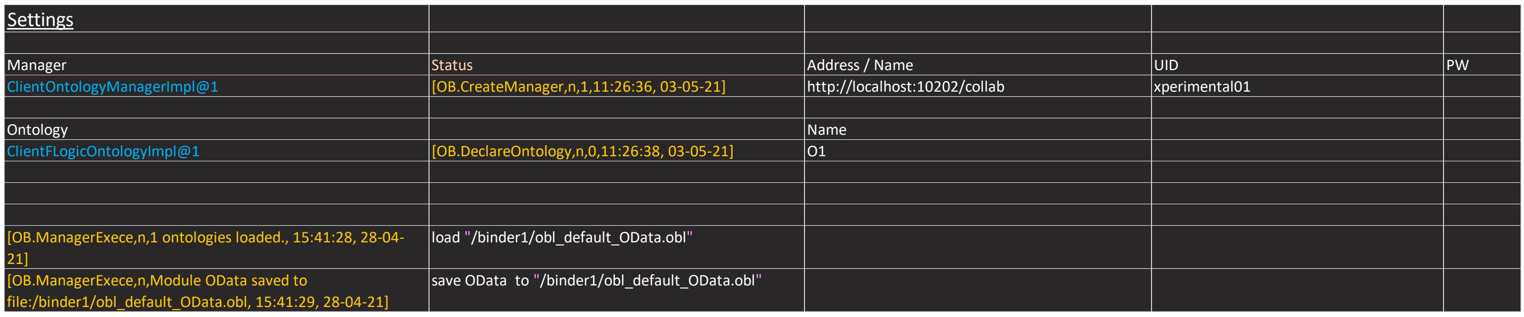
The basis of the usage instructions is the file "OntoStudio-X Demo.xlsx". The following worksheets are included in it:

- Settings (initialization of the OntoBroker server(s) and ontologies)
- SysLib (definition of static and dynamic axioms)
- Query (execution of queries)

However, the division of the worksheets is only exemplary and can be changed as desired.

1. *Initialization*

In the Settings sheet:



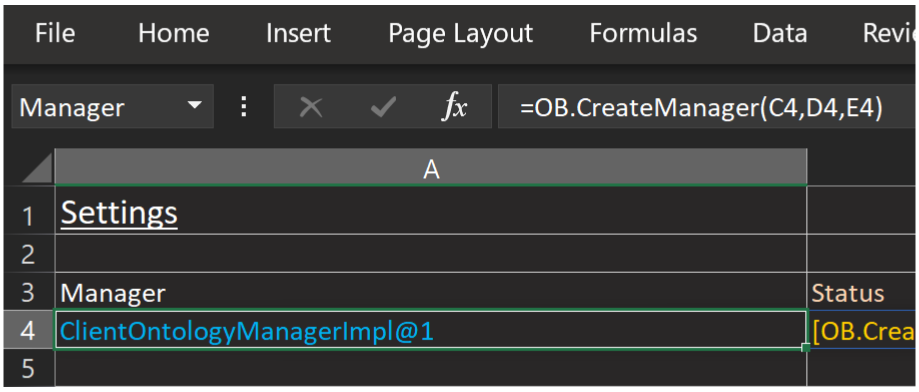
Settings				
Manager	Status	Address / Name	UID	PW
ClientOntologyManagerImpl@1	[OB.CreateManager,n,1,11:26:36, 03-05-21]	http://localhost:10202/collab	xperimental01	
Ontology		Name		
ClientFLogicOntologyImpl@1	[OB.DeclareOntology,n,0,11:26:38, 03-05-21]	O1		
[OB.ManagerExece,n,1 ontologies loaded., 15:41:28, 28-04-21]	load "/binder1/obl_default_OData.obl"			
[OB.ManagerExece,n,Module OData saved to file:/binder1/obl_default_OData.obl, 15:41:29, 28-04-21]	save OData to "/binder1/obl_default_OData.obl"			

- Update the address, UID and password of the OntoBroker server
- Go to the cell A4 with the entry ClientOntologyManagerImpl@xxx and press Ctrl-Enter. In cell B4 should appear the confirmation with time and date of the created OntoBroker Manager. In A4, the reference to the manager object appears. Per OntoBroker instance you usually need only one Manager object.

  ATTENTION: If you use a local OntoBroker version, it must be started first. Open a PowerShell window in the OntoBroker Directory and open the local OntoBroker with .\start-ontobroker.cmd. Please note that the address is as follows: http://localhost:8267/collab . The OntoBroker can be terminated in the PowerShell with Ctr-c.

- Next, enter the ontology name to use in cell B6, then go to cell A5 with the ClientFLogicOntologyImpl@xxx entry and press Ctrl-Enter. Confirmation with time and date of the referenced OntoBroker ontology should appear in cell B5. In A5 the reference to the ontology object appears. It may make sense to reference multiple ontologies. Line 5 would then have to be replicated with a different ontology name.

- Both the Manager and the Ontology object are referenced Cell name reference in the Excel Workbook:

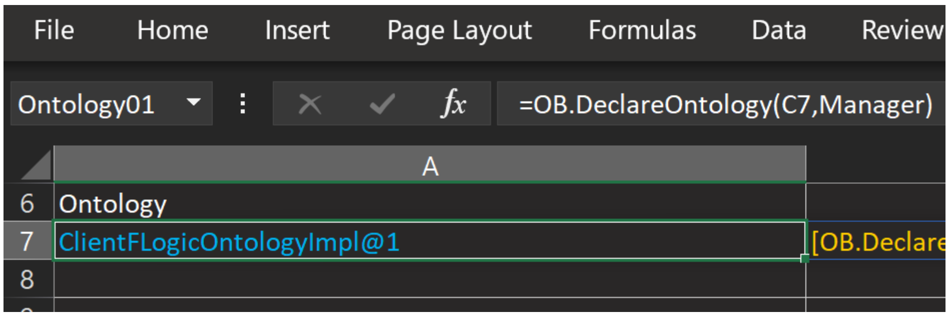


File Home Insert Page Layout Formulas Data Revi

Manager =OB.CreateManager(C4,D4,E4)

	A	
1	Settings	
2		
3	Manager	Status
4	ClientOntologyManagerImpl@1	[OB.Crea
5		

here with Manager and for the ontology with Ontology01:



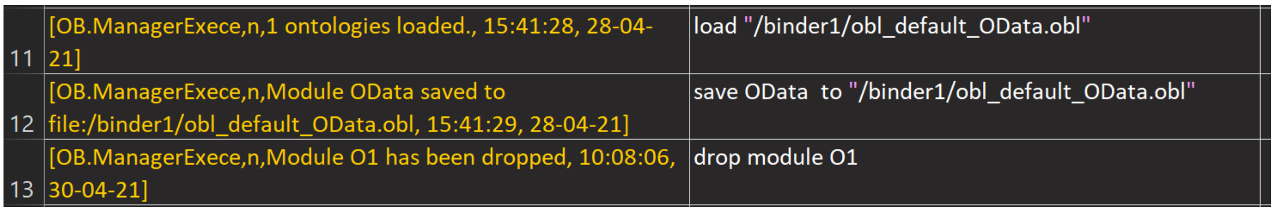
File Home Insert Page Layout Formulas Data Review

Ontology01 =OB.DeclareOntology(C7,Manager)

	A	
6	Ontology	
7	ClientFLogicOntologyImpl@1	[OB.Declare
8		

The references are important because functions based on the manager or ontology object always refer to the objects, so they can be easily referenced in the respective functions.

- On the Settings page there are also examples for initiating instructions of the Command Mode of OntoBroker, exemplarily for load, save and drop (see OntoBroker Manual):



11	[OB.ManagerExece,n,1 ontologies loaded., 15:41:28, 28-04-21]	load "/binder1/obl_default_OData.obl"
12	[OB.ManagerExece,n,Module OData saved to file:/binder1/obl_default_OData.obl, 15:41:29, 28-04-21]	save OData to "/binder1/obl_default_OData.obl"
13	[OB.ManagerExece,n,Module O1 has been dropped, 10:08:06, 30-04-21]	drop module O1

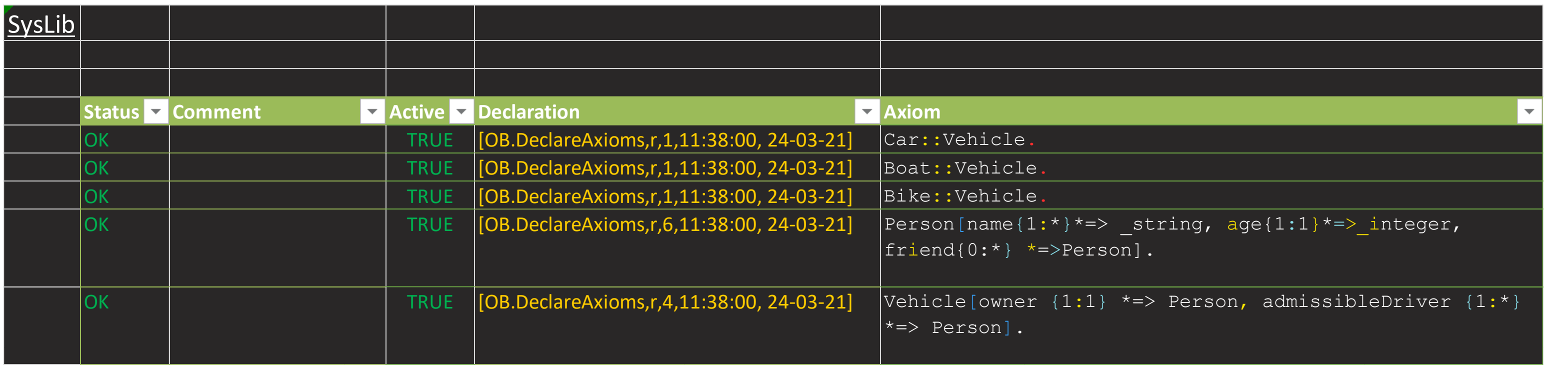
The respective instruction contents are in column B, next to the instruction in column A. This is: =OB.ManagerExec(B11, Manager), which is the function that refers only to the ontology manager (with the cell reference "Manager", see previous remarks on this) (i.e. not to a specific ontology) and executes the instruction in this case in cell B11. The execution status is in the function cell, which in turn is also executed with Ctrl-Enter. Any number of frequently used "commands" can be listed to be executed when needed.

Note the mentioned path in the instruction content. Here the ontology obl_default_OData.obl is loaded from the folder /binder1/. This is the default external (externally bound) path of an OntoBroker Docker installation. The path name can be arbitrarily different in a specific use case.

Attention: when executing the commands load and drop, the reference to the ontology object may be omitted, which is currently not automatically displayed (but the available ontologies can be displayed at any time with =OB.ListOntologies(Manager)). But since the further reference is needed for the OSX functions, it makes sense to create a new ontology object by executing the function =OB.DeclareOntology(C7,Manager) again (execute 2x if necessary - depending on the previous status, there is still no feedback during the first execution).

2. *Declaration of AxiomsAxiomen*

In the sheet SysLib:



SysLib

Status	Comment	Active	Declaration	Axiom
OK		TRUE	[OB.DeclareAxioms,r,1,11:38:00, 24-03-21]	Car::Vehicle.
OK		TRUE	[OB.DeclareAxioms,r,1,11:38:00, 24-03-21]	Boat::Vehicle.
OK		TRUE	[OB.DeclareAxioms,r,1,11:38:00, 24-03-21]	Bike::Vehicle.
OK		TRUE	[OB.DeclareAxioms,r,6,11:38:00, 24-03-21]	Person[name{1:*}*=> _string, age{1:1}*=>_integer, friend{0:*} *=>Person].
OK		TRUE	[OB.DeclareAxioms,r,4,11:38:00, 24-03-21]	Vehicle[owner {1:1} *=> Person, admissibleDriver {1:*} *=> Person].

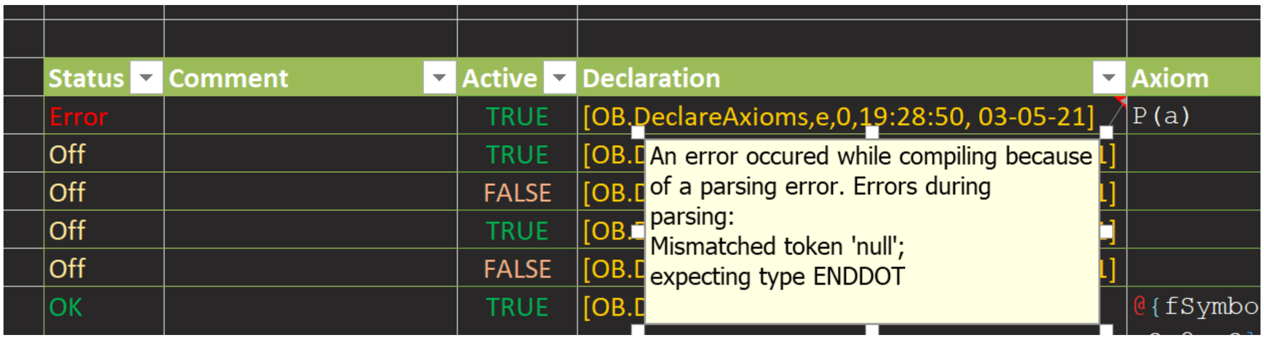
the axioms can be declared in a table form. The following columns are used in the simple usage:

Column name	*Function*
Status	Status feedback if axiom is on, off, or has an error
Comment	Location to make comments related to the axion
Active	Switch axiom on or off (TRUE/FALSE)
Declaration	The actual axiom declaration function: =OB.DeclareAxioms([@Axiom],Ontology01,[@Active])
Axiom	The actual OBL statement as described in the OntoBroker manual. Meta instructions like insert into, modify, etc. are not allowed.

The table contains additional columns on the right side reserved for functional tests (described separately).

All axioms are managed separately within OSX (cache function). I.e. if an axiom is switched off, it will be explicitly deleted in OntoBroker. Also, it is detected if an axiom is declared 2x, which is not allowed in order not to endanger the consistency of the declarations between OSX and OntoBroker.

Error messages are signaled in the Notes box (as well as in the status) and shown on mouseover:

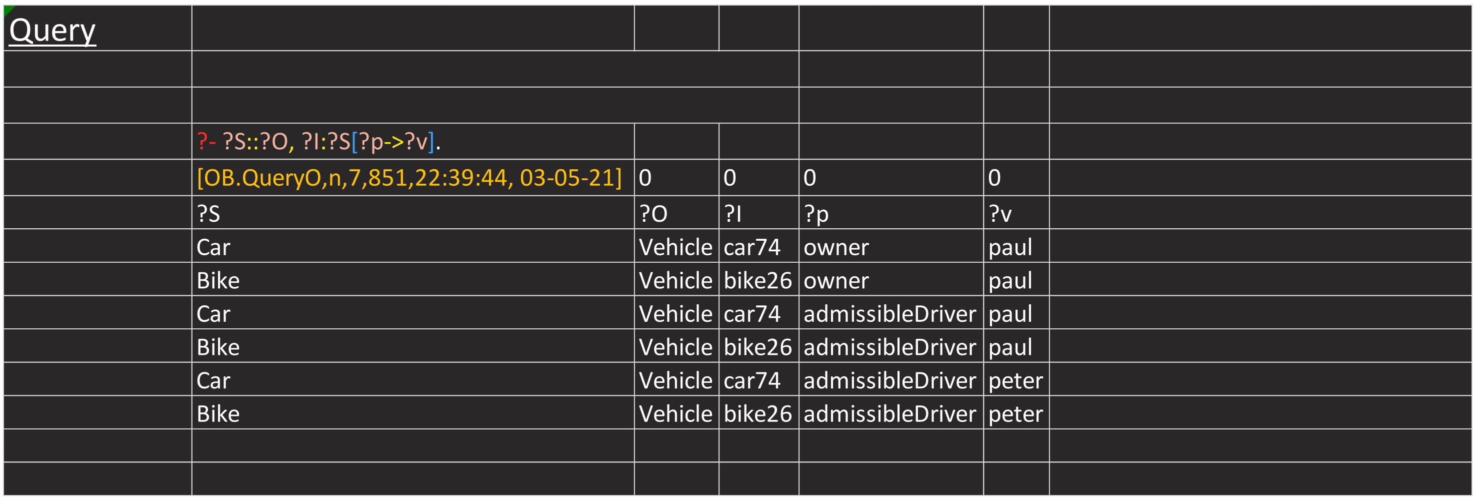


Status	Comment	Active	Declaration	Axiom
Error		TRUE	[OB.DeclareAxioms,e,0,19:28:50, 03-05-21]	P(a)
Off		TRUE	[OB.D	
Off		FALSE	[OB.D	
Off		TRUE	[OB.	
Off		FALSE	[OB.D	
OK		TRUE	[OB.D	@{fSymbo

An error occured while compiling because of a parsing error. Errors during parsing:
Mismatched token 'null'; expecting type ENDDOT

3. Performing Queries

In the Query sheet:



Query

?- ?S::?O, ?I:?S[?p->?v].				
[OB.QueryO,n,7,851,22:39:44, 03-05-21]	0	0	0	0
?S	?O	?I	?p	?v
Car	Vehicle	car74	owner	paul
Bike	Vehicle	bike26	owner	paul
Car	Vehicle	car74	admissibleDriver	paul
Bike	Vehicle	bike26	admissibleDriver	paul
Car	Vehicle	car74	admissibleDriver	peter
Bike	Vehicle	bike26	admissibleDriver	peter

the query command =OB.QueryO(B4,,Ontology01,,,,,,TRUE) will execute the query string from cell B4. The status of the query is as follows (the columns next to the query status are unoccupied array element and will still be represented as empty cell in an upcoming version) :

- n	new query
- 7	query-key
- 851	execution time measured on OSX level until display in ms
- 22:39:44	time
- 03-05-21	date (dd-mm-yy)

Appendix 1 - Reference Functions

Name	Function	Arguments (order as show below)

OB.CreateManager	create an OB manager object, which is the basis for all further operations	value = "URL", description = "location of the OB server",
		
		value = "User", description = "UID",
		
		value = "PW", description = "PW",
		
		value = "onto", description = "the ontology language to be used (optional, default = OBJECLOGIC)",
		
		value = "dummyRef", description = "for automatic updates (optional)",
		
		value = "clearOnto", description = "remove all non-default ontologies (optional, default = false)"

OB.DeclareAxioms	declare one ore more single/composite axioms, several single or composite can be comma separated	value = "axiom", description = "legal obl axiom (separated by \",\" for multiple statements)"
		
		value = "ontology", description = "ontology object",
		
		value = "addAx", description = "boolean: turn axiom on/off (optional)",
		
		value = "dummyRef", description = "dependency ref (optional)",
		
		value = "cell reference ", description = "cell reference if called by macro (optional)"

OB.DeclareOntology	obtain the reference to an onotology based on a given name or create one the ontology doesnt exist	value = "ontoStr", description = "ontology string",
		
		value = "manager", description = "manager object",
		
		value = "addOnto", description = "boolean: turn ontology on/off (optional, default = true)",
		
		value = "defaultURI", description = "boolean: prefix default URI (optional, default = true)",
		
		value = "dummyRef", description = "dependency ref (optional)",

		value = "macroMode", description = "error messages piped through result (optional, default false)"

OB.ListAxioms	return all EDB axioms	value = "managerObj", description = "reference to the OB server"
		
		value = "dummyRef", description = "dependency ref (optional)"

OB.ListFormulas	return all EDB formulas	value = "managerObj", description = "reference to the OB server"),
		
		value = "dummyRef", description = "dependency ref (optional)")

OB.ListOntologies	list all ontologies currently available	value = "manager", description = "ontology manager object"

OB.ListPredicates	return all EDB Predicates	value = "managerObj", description = "reference to the OB server",
		
		value = "dummyRef", description = "dependency ref (optional)"

OB.ListRules	return all EDB rules	value = "managerObj", description = "reference to the OB server"
		
		value = "dummyRef", description = "dependency ref (optional)"

OB.LoadOntology	load a named ontology	value = "manager", description = "manager object",
		
		value = "fileStr", description = "valid path of the ontology incl. filename",
		
		value = "ontoStr", description = "name of the ontology to be loaded",
		
		value = "dummyRef", description = "dependency ref (optional)")

OB.ManagerExec	execute the commands at level ontology manager	value = "execStr", description = "string to be executed", value = "manager", description = "ontology manager object",

OB.QueryO	execute query string	value = "queryStr", description = "query string to be executed",
		
		value = "nameSpaceStr", description = "name space (optional, default: obl:default)",
		
		value = "onto", description = "ontology object",
		
		value = "headL", description = "(headline output (optional, default: true)",

		value = "transP", description = "transposed output (optional, default: false)",
		value = "noStrQuote", description = "remove \\\" double quotes (optional, default: false)",
		value = "resultOntology", description = "ontology object where the result should be stored (optional)",
		value = "dummyRef", description = "dependency ref (optional)",
		value = "macroMode", description = "for calls from VBA macros(optional)",
		value = "rawMode", description = "(optional, default: false)"

OB.SaveOntology	Save the ontology to a file	value = "fileStr", description = "name of the file",
		value = "onto", description = "ontology bolean: turn axiom on/off (optional)",
		value = "format", description = "ontology format (optional - default is OBJECTLOGIC")
		value = "pathConvention", description = "optional, default is true = MSDOS")

OB.sysGetClassName	return the full qualified name of the referred class	value = "objectParm", description = "reference to the class object", value = "dummyRef", description = "dependency ref (optional)"
OB.sysGetOntoName	return the full qualified name of the referred ontology	value = "objectParm", description = "reference to the ontology object", value = "dummyRef", description = "dependency ref (optional)"
OB.sysListAxiomsCache	return the axiom cache array	-
OB.sysListOntologyCache	return the axiom cache array	-

OSX_AddSheetAxiom	Turns complete sheet (first line headers, organized in record manner) into axioms	:param name: name of sheet :param onto: ontology object reference :param active: active flag :param dummy: dummy reference
OSX_GetHome	Returns home path	-
OSX_GetJINXHome	Returns home path of JINX directory	-
OSX_GetPyXLLHome	Returns home path of PyXLL directory	-
OSX_LatestInPath	Returns full path to log file	:param path: path part of full path filename :param base: type of log file
OSX_ListSheets	List all workbook sheets but not special ones with $ prefix	-
OSX_MySheet	Returns sheet name	-
OSX_OBLStatusCheck	Evaluates DeclareAxioms status and returns result	:param val: string with return status
OSX_RangeCalc	Calculates selection	-
OSX_ReadOBKeyXML()	Returns the OB license key data	-
OSX_RevFileRead	Reads text file and writs it in reverse order line be line into the sheet (un-formatted, faster)	:param filename: name of text file to read

OSX_RevFileReadB	Reads text file and writs it in reverse order line be line into the sheet (formatted)	:param filename: name of text file to read
OSX_RunOS	Executes operating system command, returns output as string	:param cmd: OS command
OSX_RunOS_CSV	Executes operating system command, returns output as csv	:param cmd: OS command
OSX_RunOSbg	Executes operating system command in background, returns PID	:param cmd: OS command
OSX_SheetCalc	Calculates sheet	-
OSX_TermOSbg	Terminates process based on PID, returns result	:param val: process ID

Shortcuts / Funktionstasten

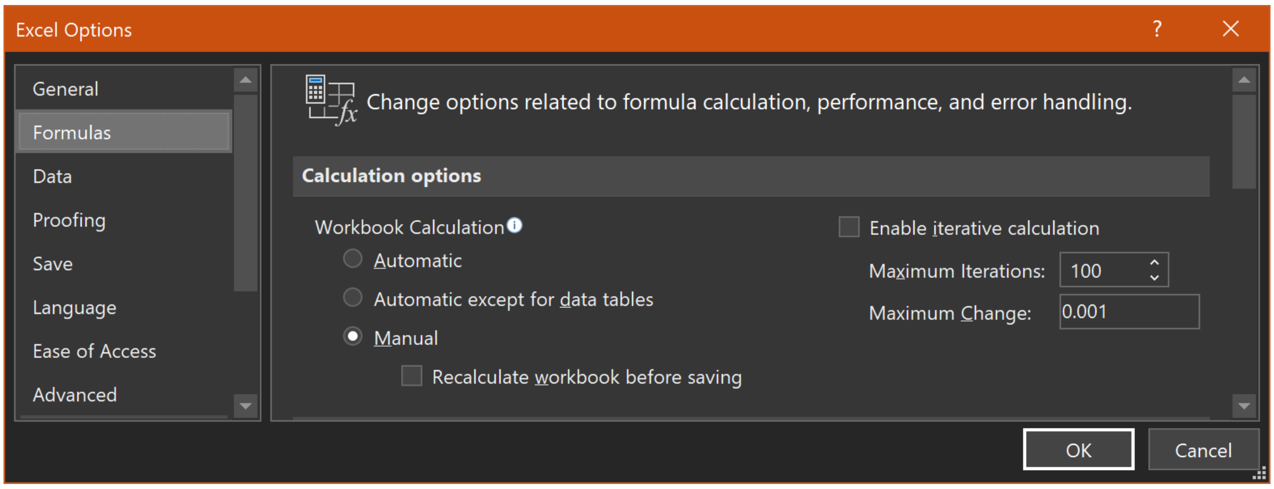
Shortcut	Description	Comment
F1	Colorize OBL syntax	Not yet applicable for extended syntax
F2	Start OSX editor in colorizing mode	Both editors automatically save all changes after escape, Ctrl-s or exit window click.

		If the editor has been started with the query text cell AND the query function selected, queries can be triggered by Shift-F9 to check immediately the query formula.
F3	Start OSX editor in NONE-colorizing mode	
Ctrl-Enter	Calculate cell	
Alt-Enter	Calculate sheet	

Appendix 2 - Installation

1. make sure that all system requirements are met.
2. Please unzip the content of the "semafora OSX Vxx_xx.zip" file into your user directory.
3. Make the following recommended Excel Options settings:

   3.1. Switch on the automatic calculation of the worksheets and the calculation before saving:


Excel Options

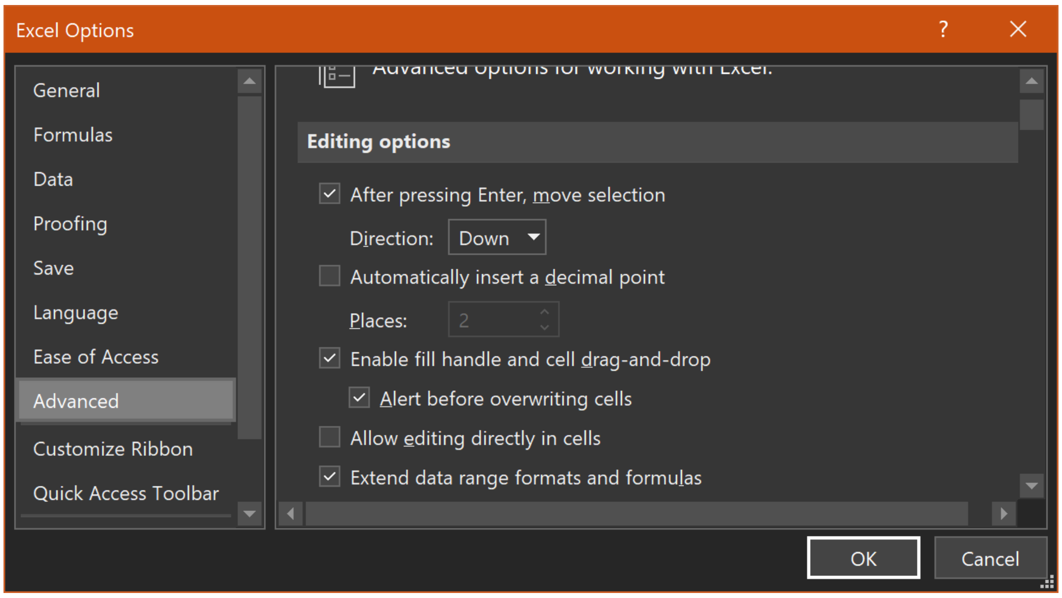
Change options related to formula calculation, performance, and error handling.

Calculation options

Workbook Calculation
- ○ Automatic
- ○ Automatic except for data tables
- ● Manual
  - ☐ Recalculate workbook before saving

☐ Enable iterative calculation
Maximum Iterations: 100
Maximum Change: 0.001

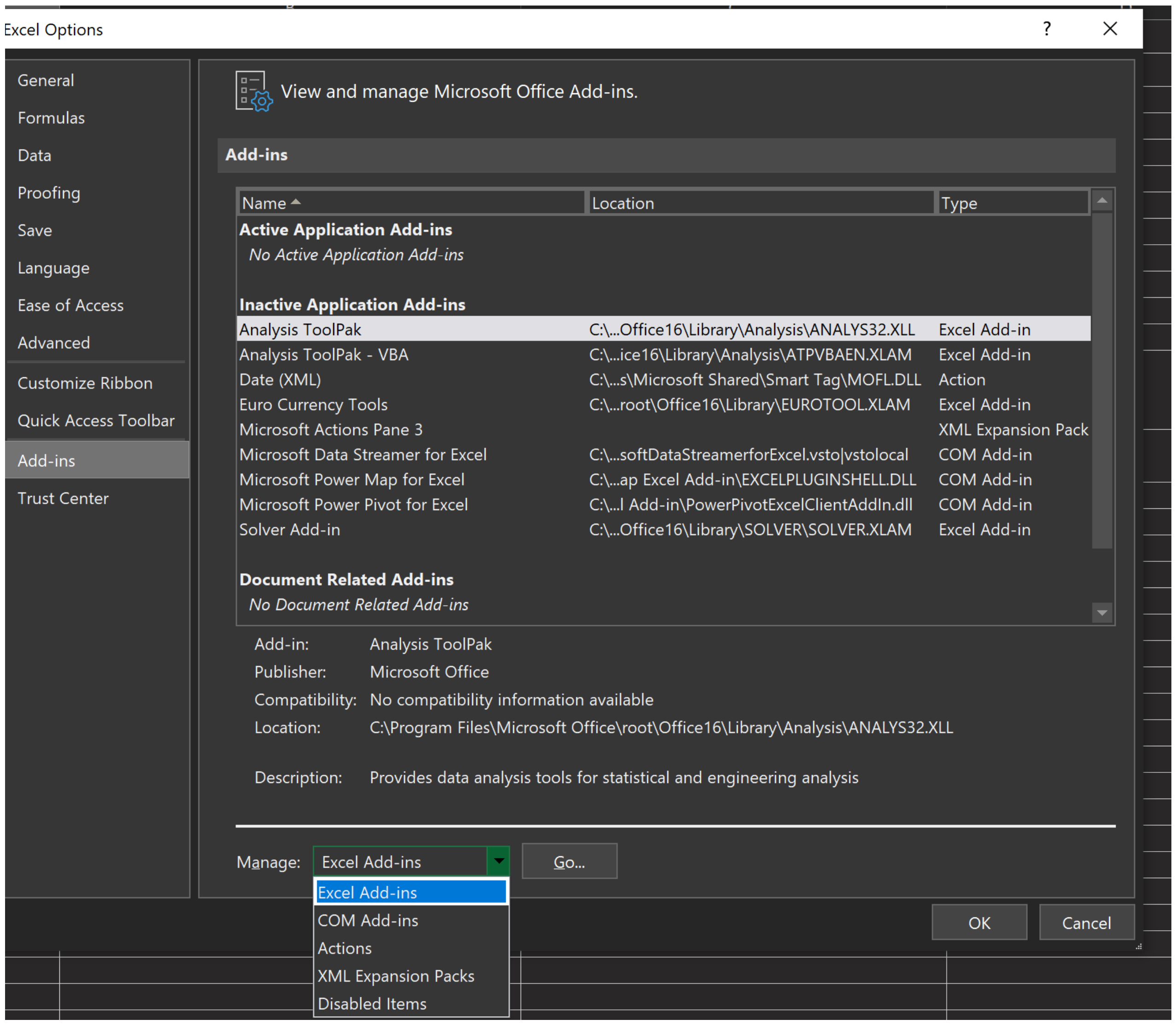
   3.2. S Turn off in-cell editing.


Excel Options

Advanced options for working with Excel.

Editing options
- ☑ After pressing Enter, move selection
  - Direction: Down
- ☐ Automatically insert a decimal point
  - Places: 2
- ☑ Enable fill handle and cell drag-and-drop
  - ☑ Alert before overwriting cells
- ☐ Allow editing directly in cells
- ☑ Extend data range formats and formulas

4. Copy the demo file "OntoStudio-X Demo Vxx_yy.xlsx" into your working directory.
5. Open the file and add the necessary Excel add-ins:


Excel Options

View and manage Microsoft Office Add-ins.

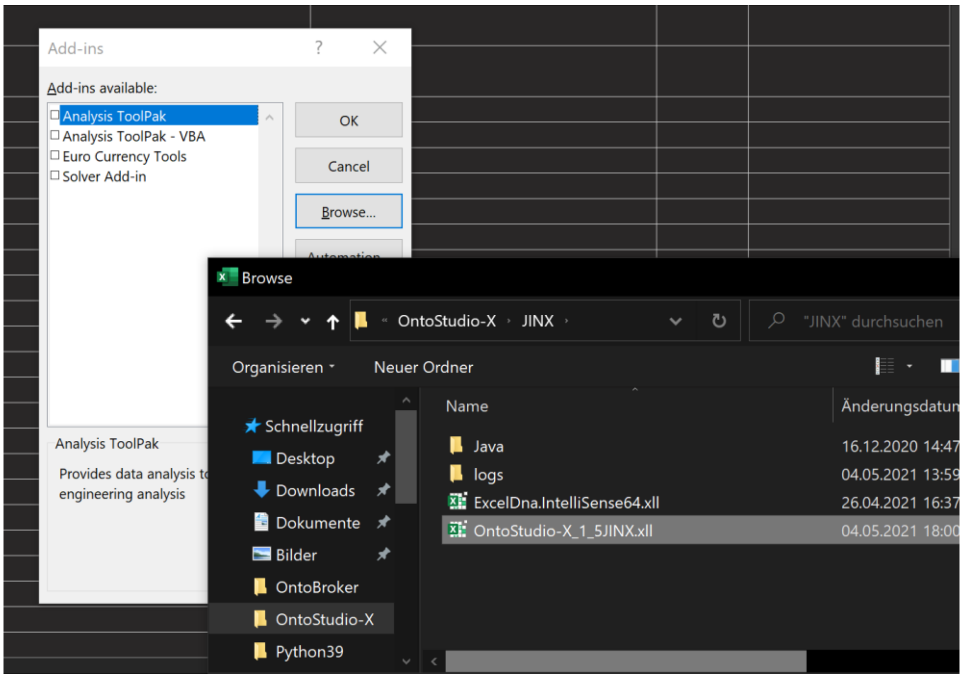
Add-ins

Name	Location	Type
Active Application Add-ins		
No Active Application Add-ins		
Inactive Application Add-ins		
Analysis ToolPak	C:\...Office16\Library\Analysis\ANALYS32.XLL	Excel Add-in
Analysis ToolPak - VBA	C:\...ice16\Library\Analysis\ATPVBAEN.XLAM	Excel Add-in
Date (XML)	C:\...s\Microsoft Shared\Smart Tag\MOFL.DLL	Action
Euro Currency Tools	C:\...root\Office16\Library\EUROTOOL.XLAM	Excel Add-in
Microsoft Actions Pane 3		XML Expansion Pack
Microsoft Data Streamer for Excel	C:\...softDataStreamerforExcel.vsto|vstolocal	COM Add-in
Microsoft Power Map for Excel	C:\...ap Excel Add-in\EXCELPLUGINSHELL.DLL	COM Add-in
Microsoft Power Pivot for Excel	C:\...l Add-in\PowerPivotExcelClientAddIn.dll	COM Add-in
Solver Add-in	C:\...Office16\Library\SOLVER\SOLVER.XLAM	Excel Add-in
Document Related Add-ins		
No Document Related Add-ins		

Add-in:	Analysis ToolPak
Publisher:	Microsoft Office
Compatibility:	No compatibility information available
Location:	C:\Program Files\Microsoft Office\root\Office16\Library\Analysis\ANALYS32.XLL
Description:	Provides data analysis tools for statistical and engineering analysis

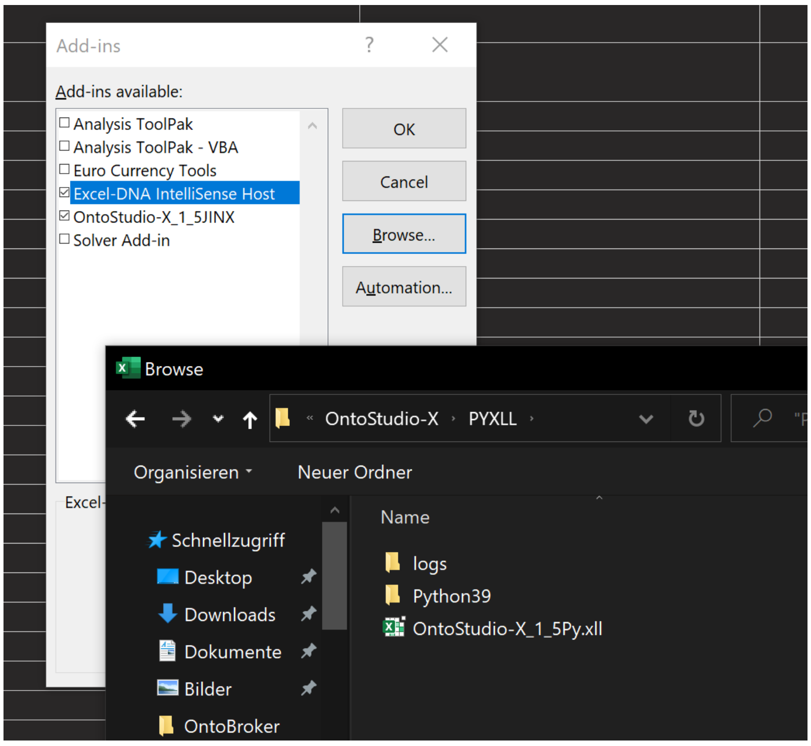
Manage: Excel Add-ins

Click Go and load a total of 3 .xll files, first in the JINX folder:

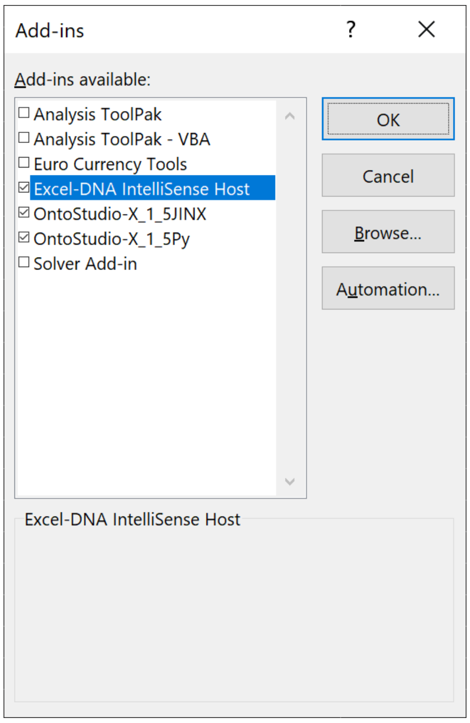


Name	Änderungsdatun
Java	16.12.2020 14:47
logs	04.05.2021 13:59
ExcelDna.IntelliSense64.xll	26.04.2021 16:37
OntoStudio-X_1_5JINX.xll	04.05.2021 18:00

and then in the PyXLL folder:



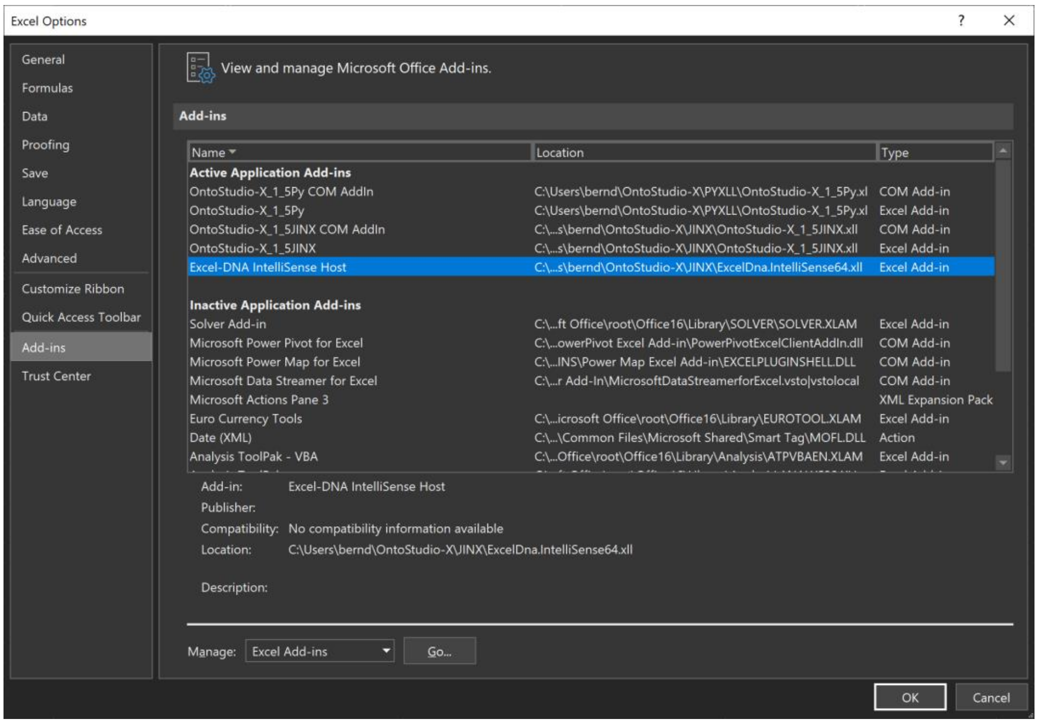
The add-ins loading window will look like this:



- ☐ Analysis ToolPak
- ☐ Analysis ToolPak - VBA
- ☐ Euro Currency Tools
- ☑ Excel-DNA IntelliSense Host
- ☑ OntoStudio-X_1_5JINX
- ☑ OntoStudio-X_1_5Py
- ☐ Solver Add-in

Excel-DNA IntelliSense Host

Click OK and the overview window should look like this:



View and manage Microsoft Office Add-ins.

Name	Location	Type
Active Application Add-ins		
OntoStudio-X_1_5Py COM AddIn	C:\Users\bernd\OntoStudio-X\PYXLL\OntoStudio-X_1_5Py.xl	COM Add-in
OntoStudio-X_1_5Py	C:\Users\bernd\OntoStudio-X\PYXLL\OntoStudio-X_1_5Py.xl	Excel Add-in
OntoStudio-X_1_5JINX COM AddIn	C:\...s\bernd\OntoStudio-X\JINX\OntoStudio-X_1_5JINX.xll	COM Add-in
OntoStudio-X_1_5JINX	C:\...s\bernd\OntoStudio-X\JINX\OntoStudio-X_1_5JINX.xll	Excel Add-in
Excel-DNA IntelliSense Host	C:\...s\bernd\OntoStudio-X\JINX\ExcelDna.IntelliSense64.xll	Excel Add-in
Inactive Application Add-ins		
Solver Add-in	C:\...ft Office\root\Office16\Library\SOLVER\SOLVER.XLAM	Excel Add-in
Microsoft Power Pivot for Excel	C:\...owerPivot Excel Add-in\PowerPivotExcelClientAddIn.dll	COM Add-in
Microsoft Power Map for Excel	C:\...INS\Power Map Excel Add-in\EXCELPLUGINSHELL.DLL	COM Add-in
Microsoft Data Streamer for Excel	C:\...r Add-In\MicrosoftDataStreamerforExcel.vsto|vstolocal	COM Add-in
Microsoft Actions Pane 3		XML Expansion Pack
Euro Currency Tools	C:\...icrosoft Office\root\Office16\Library\EUROTOOL.XLAM	Excel Add-in
Date (XML)	C:\...\Common Files\Microsoft Shared\Smart Tag\MOFL.DLL	Action
Analysis ToolPak - VBA	C:\...Office\root\Office16\Library\Analysis\ATPVBAEN.XLAM	Excel Add-in

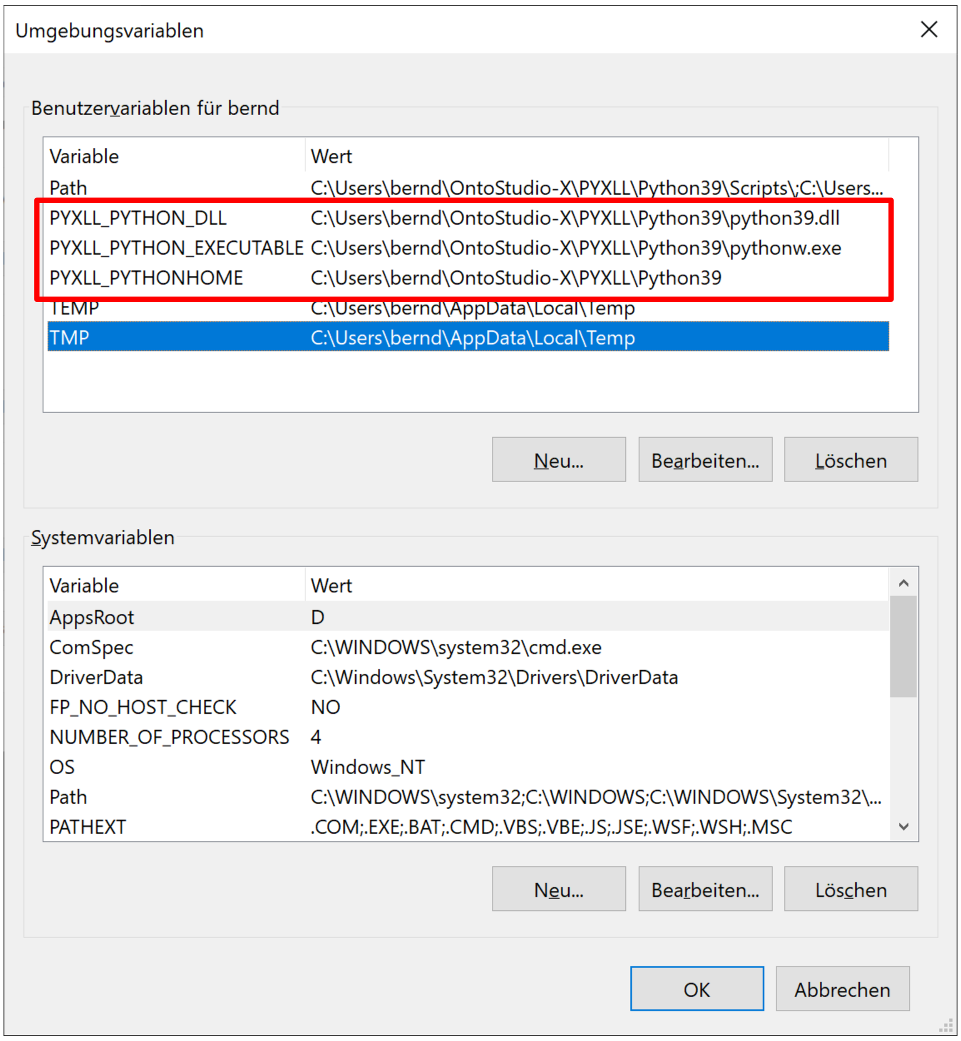
Add-in: Excel-DNA IntelliSense Host
Publisher:
Compatibility: No compatibility information available
Location: C:\Users\bernd\OntoStudio-X\JINX\ExcelDna.IntelliSense64.xll

Description:

Manage: Excel Add-ins

Click OK and then it may take a few seconds to load the add-ins.

6. Click on the Windows taskbar and type "env". Click the offered program Control Panel and select the menu item "Environment variables" and make the following setting:



Umgebungsvariablen

Benutzervariablen für bernd

Variable	Wert
Path	C:\Users\bernd\OntoStudio-X\PYXLL\Python39\Scripts\;C:\Users...
PYXLL_PYTHON_DLL	C:\Users\bernd\OntoStudio-X\PYXLL\Python39\python39.dll
PYXLL_PYTHON_EXECUTABLE	C:\Users\bernd\OntoStudio-X\PYXLL\Python39\pythonw.exe
PYXLL_PYTHONHOME	C:\Users\bernd\OntoStudio-X\PYXLL\Python39
TEMP	C:\Users\bernd\AppData\Local\Temp
TMP	C:\Users\bernd\AppData\Local\Temp

Systemvariablen

Variable	Wert
AppsRoot	D
ComSpec	C:\WINDOWS\system32\cmd.exe
DriverData	C:\Windows\System32\Drivers\DriverData
FP_NO_HOST_CHECK	NO
NUMBER_OF_PROCESSORS	4
OS	Windows_NT
Path	C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\System32\...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC

Confirm all settings with OK.

After that you can start the Excel file OntoStudio-X Demo Vxx_yy.xlsx to test the installation.

Appendix 3 - Visualization with OSXVis

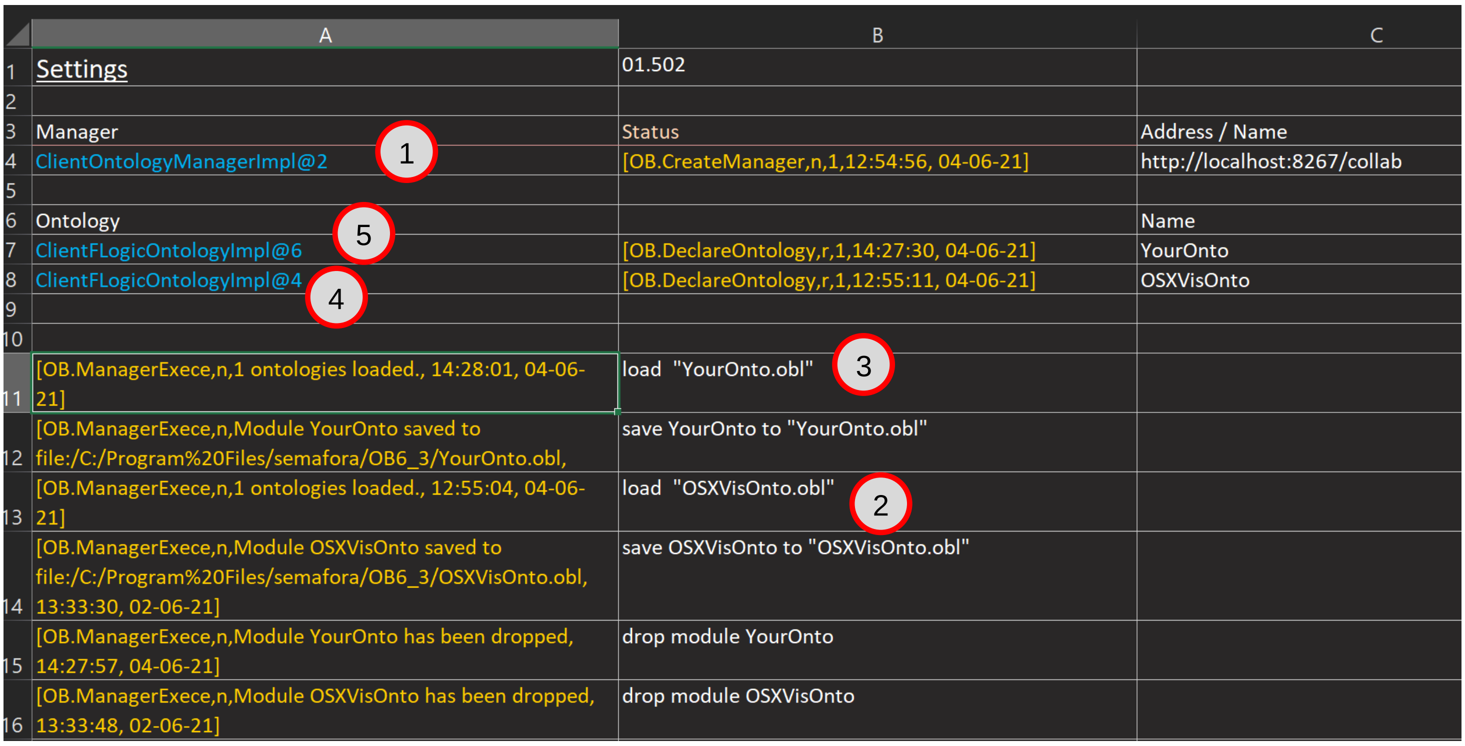
A 3.1 Foreword

- OSXVis runs in its own ontology, the OSXVisONto ensure that all visualization operations are performed separately from the data. Data to be visualized must therefore also always be explicitly specified with the respective ontology (using the ontology parameter).
- The visualization itself is performed by the web server OSXVis.exe.
- The data flow to the visualization is as follows:
    - A query for generating the data is defined.
    - The nodes to be used are declared.
    - The SPO relations to be displayed are declared.
    - The query is executed, the data is assigned to the node and link representation, the nodes and links are provided with color information and everything is stored in an OBL map structure.
    - The visualization map is passed via a defined socket to the web server, which should already have a connection with a web client.
    - The visualization data is displayed on the web client (Chrome works well). The display can be modified in viewing angle and distance.
    - To display further data again, the display must be deleted beforehand.

A 3.2 Preparation

- Start the manager (1) (see Figure A3.1).
- Loading the OSXVisOnto (2) and, if necessary, your own data ontology (3).

Assignment of OSX ontology references (4) and (5), which can then be used to declare axioms and perform queries in the respective ontologies:



	A	B	C
1	Settings	01.502	
2			
3	Manager	Status	Address / Name
4	ClientOntologyManagerImpl@2	[OB.CreateManager,n,1,12:54:56, 04-06-21]	http://localhost:8267/collab
5			
6	Ontology		Name
7	ClientFLogicOntologyImpl@6	[OB.DeclareOntology,r,1,14:27:30, 04-06-21]	YourOnto
8	ClientFLogicOntologyImpl@4	[OB.DeclareOntology,r,1,12:55:11, 04-06-21]	OSXVisOnto
9			
10			
11	[OB.ManagerExece,n,1 ontologies loaded., 14:28:01, 04-06-21]	load "YourOnto.obl"	
12	[OB.ManagerExece,n,Module YourOnto saved to file:/C:/Program%20Files/semafora/OB6_3/YourOnto.obl,	save YourOnto to "YourOnto.obl"	
13	[OB.ManagerExece,n,1 ontologies loaded., 12:55:04, 04-06-21]	load "OSXVisOnto.obl"	
14	[OB.ManagerExece,n,Module OSXVisOnto saved to file:/C:/Program%20Files/semafora/OB6_3/OSXVisOnto.obl, 13:33:30, 02-06-21]	save OSXVisOnto to "OSXVisOnto.obl"	
15	[OB.ManagerExece,n,Module YourOnto has been dropped, 14:27:57, 04-06-21]	drop module YourOnto	
16	[OB.ManagerExece,n,Module OSXVisOnto has been dropped, 13:33:48, 02-06-21]	drop module OSXVisOnto	

Figure A3.1: Sheet Settings

- In addition to the data ontology, the OSXVis ontology is now also loaded. This is also available in the SysLibVis sheet for modification and can be used separately, especially for setting the socket port and the semantic color mapper.

- The default settings of the web and socket ports are: localhost/5000 and localhost/6000. For changes the predicate OSXVisCon("127.0.0.1", 6000) has to be adjusted.The default settings of the web server can be changed at startup:

```
@{fOSXVis2_3} OSXVis(start, windows,?x) :-
_sysIO2("""","”powershell.exe Start-Process -FilePath
\"./OSXVis.exe\"""","",?x). "./OSXVis.exe\"
```

- must be customized with the arguments: --webhost, --webport , --sockhost, --sockport.

- The web server is started with (1) and stopped with (2) (see Figure A3.3). In the console window of the web server appears shortly after start that the socket is available.

- With (3) a chrome window for displaying the web server is started.

- With (4) the display content of the web server is deleted.

- To demonstrate the current display capabilities of the web server, the tutorial ontology is included in the OSXVisOnto ontology. This can be called by (5). All connections of the entities with explicit relations as well as with implicit relations are displayed (hasProperty or age - visible on mouse-over):

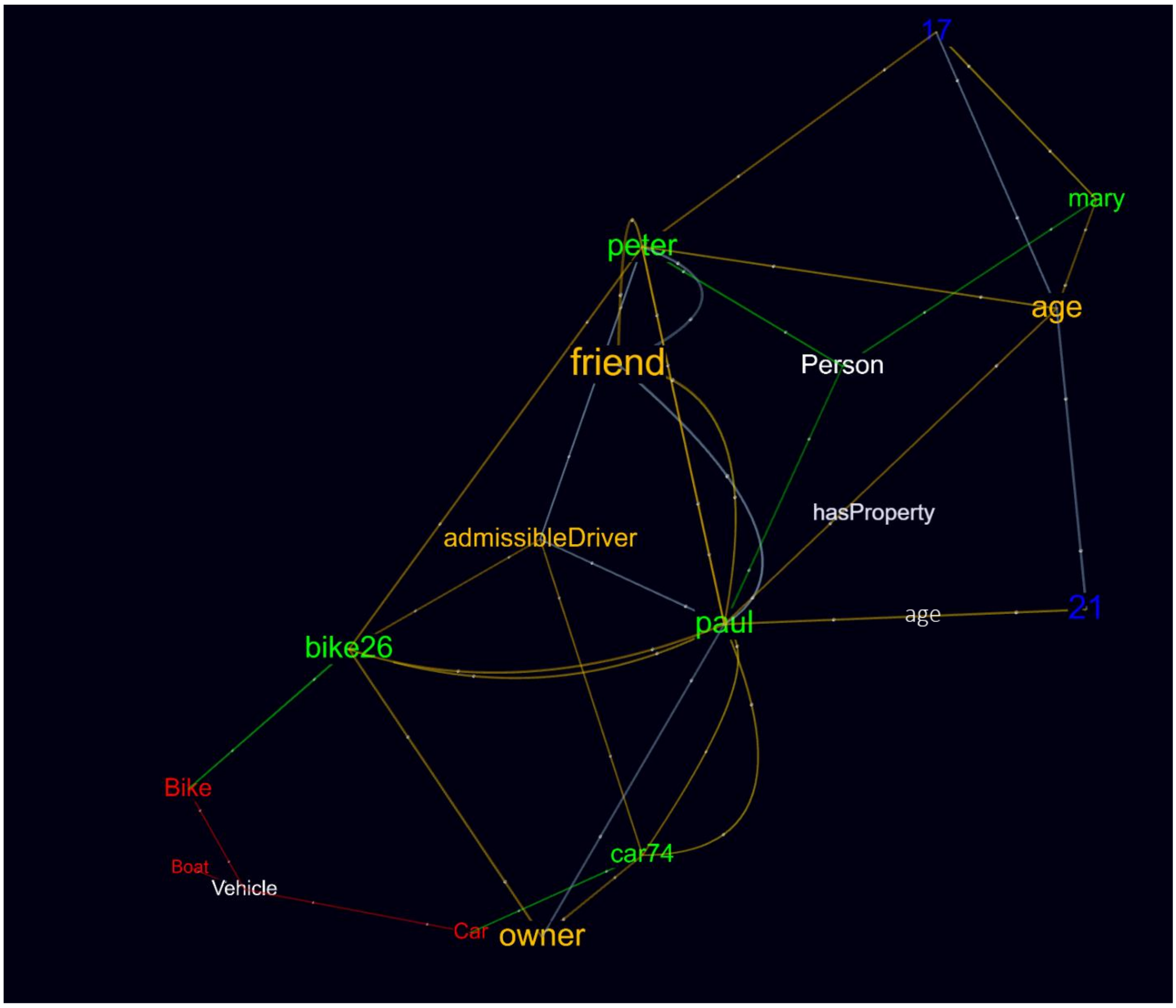


Figure A3.2: Display window of the WebServer OSXVis showing the tutorial ontology

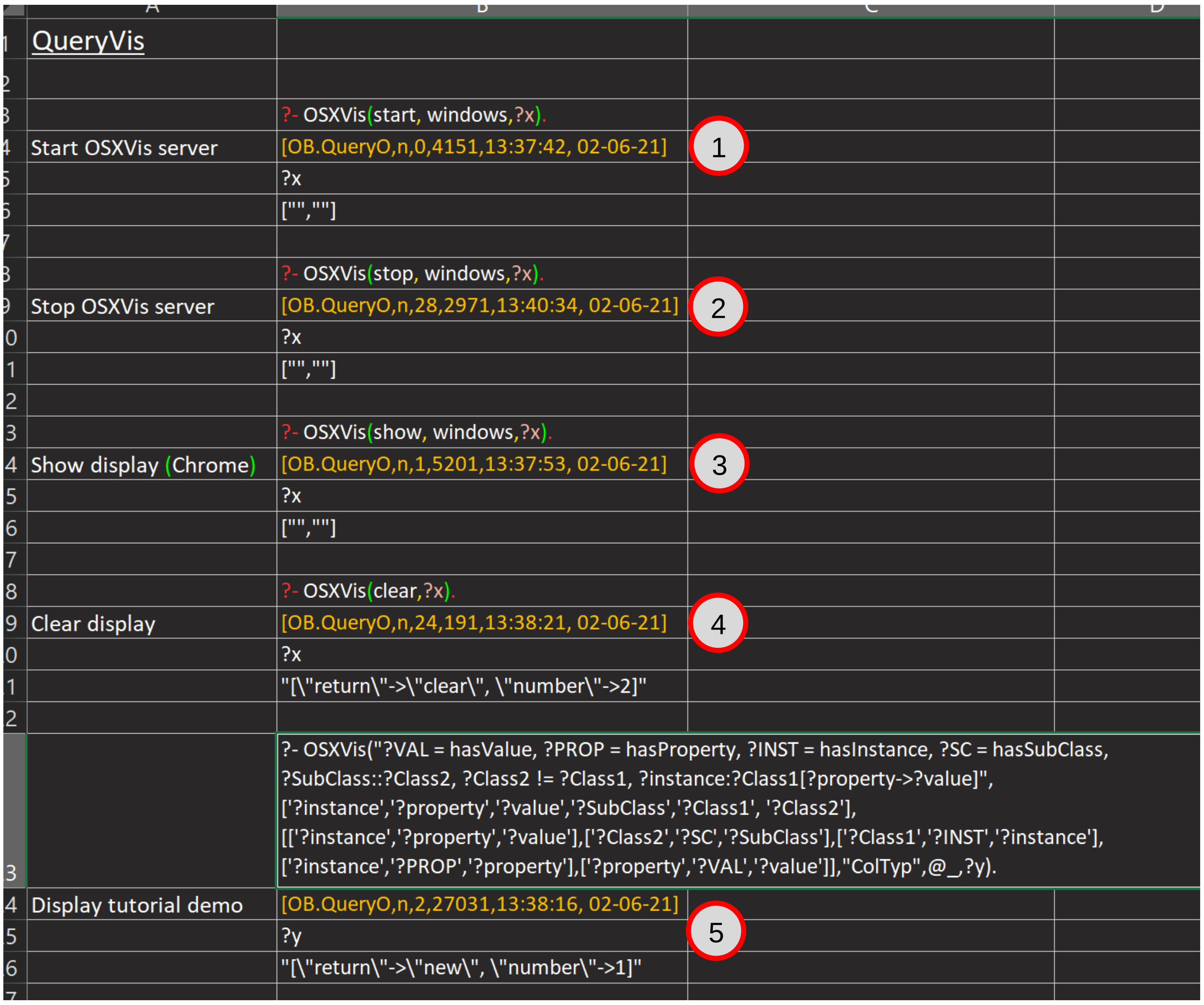
- The moving particles show the direction of the relations. Opposite particles have the same relation, only mirrored. Multiple relations between entities with different contents are curved around the center line for differentiation.
- The tutorial ontology also shows the use of several groups of visualization relations. The parameters at a glance mean:

```
?- OSXVis("?VAL = hasValue, ?PROP = hasProperty, ?INST = hasInstance, ?SC =
hasSubClass, ?SubClass::?Class2, ?Class2 != ?Class1,
?instance:?Class1[?property->?value]",
['?instance','?property','?value','?SubClass','?Class1', '?Class2'],
[['?instance','?property','?value'],['?Class2','?SC','?SubClass'],['?Class1','?
INST','?instance'],
['?instance','?PROP','?property'],['?property','?VAL','?value']],"ColTyp",@_,?y
).
```

- Query Yext
- Nodes to be displayed
- Series of SPO links
- Color mapping operator
- Ontology to which the query text refers
- Return result

  - It is recommended to test the query text in a regular "?-" query before. Possible errors will appear in the OntoBroker console.
  - All nodes must occur as query variables.
  - All nodes should at least appear as S and P parts in the SPO links, otherwise they hang connectionless in the representation space.
  - The P node of the SPO link must also appear as a variable in the query text.
  - In the example, the "self" ontology "@_" is used. Data outside the OSXVisOnto ontology must be referenced accordingly.
  - ?y returns the result and also whether there were any errors. The transmitted display list can be printed in the console window with the web server debug option "--debug True".
  - If the parameter list is shortened by the color mapping operator, an internal cyclic color table is used (see Color predicate in the ontology). Unsorted, the colors of the entities can change from representation to representation, because then no output sequence logic is used.
  - When using a color mapping operator, the colors are semantically assigned. In the used example according to functionality (e.g. whether the entity is a property, but just e.g. no class -concept- and thus get the color Color(3,,). The 3rd entry of the color predicate refers to the color of the dynamic direction particles and is usually gray). Any other color mapping operators can be defined:

```
@{fColTyp0_2} ColTyp(?x,[?y,?z]) :- (
(_rootconcepts(?x,@_), not _properties(?x,@_), not IsInstance(?x), not
IsValue(?x), Color(0,?y,?z)) or
(_concepts(?x,@_), not _rootconcepts(?x,@_), not IsInstance(?x), not
_properties(?x,@_), not IsValue(?x), Color(1,?y,?z)) or
(IsInstance(?x), not _concepts(?x,@_),  Color(2,?y,?z)) or
(_properties(?x,@_), not _concepts(?x,@_), not IsInstance(?x), Color(3,?y,?z))
or
(IsValue(?x), not _properties(?x,@_), not _concepts(?x,@_), not IsInstance(?x),
Color(4,?y,?z)) or
( ?x == hasSubClass, Color(1,?y,?z)) or
( ?x == hasInstance, Color(2,?y,?z)) or
( ?x == hasProperty, Color(3,?y,?z)) or
( ?x == hasValue, Color(12,?y,?z))
).
```



	A	B	C	D
1	QueryVis			
2				
3		?- OSXVis(start, windows,?x).		
4	Start OSXVis server	[OB.QueryO,n,0,4151,13:37:42, 02-06-21]	1	
5		?x		
6		["",""]		
7				
8		?- OSXVis(stop, windows,?x).		
9	Stop OSXVis server	[OB.QueryO,n,28,2971,13:40:34, 02-06-21]	2	
0		?x		
1		["",""]		
2				
3		?- OSXVis(show, windows,?x).		
4	Show display (Chrome)	[OB.QueryO,n,1,5201,13:37:53, 02-06-21]	3	
5		?x		
6		["",""]		
7				
8		?- OSXVis(clear,?x).		
9	Clear display	[OB.QueryO,n,24,191,13:38:21, 02-06-21]	4	
0		?x		
1		"[\"return\"->\"clear\", \"number\"->2]"		
2				
3		?- OSXVis("?VAL = hasValue, ?PROP = hasProperty, ?INST = hasInstance, ?SC = hasSubClass, ?SubClass::?Class2, ?Class2 != ?Class1, ?instance:?Class1[?property->?value]", ['?instance','?property','?value','?SubClass','?Class1', '?Class2'], [['?instance','?property','?value'],['?Class2','?SC','?SubClass'],['?Class1','?INST','?instance'], ['?instance','?PROP','?property'],['?property','?VAL','?value']],"ColTyp",@_,?y).		
4	Display tutorial demo	[OB.QueryO,n,2,27031,13:38:16, 02-06-21]	5	
5		?y		
6		"[\"return\"->\"new\", \"number\"->1]"		
7				

Figure A3.3: Sheet QueryVis

Execution of the commands always happens via the OSX =OB.QueryO(...) function in the cell (typically displayed in dark yellow as "[OB.QueryO,....]").

In case of possible error messages, please contact support@semafora-systems.com.